Agile Principles / Miniature (DIT191 / EDA397)

Eric Knauss
<eric.knauss@cse.gu.se>

Organizational

- Next steps
  - Now: Miniature on Agile
  - Today, 15:00 - 17:00: Feedback on Project proposals
- Exam date
  - Jun-1st, am
  - Guest lectures will provide exam questions
- Mandatory / obligatory meetings
  - Lectures:
    Up to you, but cannot guarantee to have self-contained slides
  - Acceptance tests: You need to be present!
    - You can participate remotely, if your group is organizing that
    - One time missing okay, but need rework (500 words: As an external consultant, I would suggest the following to improve agility of my team)

Agenda today

- Course overview
- Miniature
- Agile Principles revisited

Sprint 1: Getting started

http://commons.wikimedia.org/wiki/File:Sprint_01.jpg

My idea of this course…

- 2 Streams
  - Lectures – *Learn Agile*
  - Project work – *Experience Agile*

- 3 Sprints
  - First sprint – *Getting started*
  - Second sprint – *Focus on Project work*
  - Third sprint – *Advanced Concepts*

Course Objectives

Knowledge and understanding	Skills and ability	Judgement and approach
Compare agile and traditional softw. dev,	Forming a team organically	Explain: people/commun. centric dev.
Relate lean and agile development	Collaborate in small software dev. teams	Apply fact: people drive project success
Contrast different agile methodologies	Interact and show progress continuously	Describe: No single methodology fits all
Use the agile manifest and its accompanying principles	Develop SW using small and frequent iterations	Discuss: methodology needs to adopt to culture
Discuss what is different when leading an agile team	Use test-driven dev. and automated tests	
	Refactor a program/design	
	Be member of agile team	
	Incremental planning using user stories	

Sprint 1

Sprint 2

Sprint 3

What is agility in Software Development?

Agile Development:
A Miniature

http://mediagallery.usatoday.com/New+Flame

Miniatures

- Good to get started in the project
  - Shared ideas / concepts
  - http://c2.com/xp/ExtremeHour.html
  - http://www.massey.ac.nz/~dpparson/agilehour.htm

- Idea: Simulate an agile project within a limited time
  - Agile / Extreme Hour do not scale
  - Lego-Scrum does not scale
  - Thus, falling back to a simulation first presented by Chris Rupp, Sophist

Round 1

- Create teams of 4 to 6 persons
- Assign roles in each team: same number of customers and developers
- Customers and developers sit as far apart as possible
- Customers write instructions for developers
- One of the customers
  - brings written instructions to developers
  - can answer (written) questions with (written) answers
- Talking and drawings between customers and developers are not permitted

- Time for this round: 10 minutes

Retrospective of Applied Strategy

What did work well?

- …

What did not work well?

- Not enough time/suddenly over
- Customers wrote too long → no time for developers
- Communication not fast enough
- Descriptions confusing, full of contradictions

What should we change?

- Time management
- Task management
- Incremental work
- Iterative work
- Define/control language
- Use coordinate-system
- Specify from abstract descriptions to specifics
- Communicate "big picture"
- Use metaphors

Round 2

- Same rules as in Round 1, except …
- Shorter Iterations:
  - Developers can send Shape/Picture back
  - Customers can write change request for a Shape or continue with next Shape
- Time for this round: 10 minutes

Retrospective of Applied Strategy

What did work well?

- Task management (increments)
- Iterations
- Metaphors
- Common language
- Time management

What did not work well?

- Integration of Increments and Iterations to whole picture
- Ambiguity of metaphors

What should we change?

- Introduce Integration Management
- More and faster feedback

Round 3

- Only one customer per team! All others are developers
- Customer is allowed to see drawing and memorize it
- Customers explains the drawing using words only
  - No hands!

- Time for this round: 5 minutes

Retrospective of Applied Strategy

What did work well?

- Task management
- Direct feedback of customer
- Verbal communication

What did not work well?

- Integration is challenging
- Customer cannot keep all developers busy
- Strategy not applicable
- Common language not applicable

What should we change?

Conclusion

- What did we learn?
  - Spatial distance hinders communication
  - Multimodal communication helps
  - Communication has limitations
  - Feedback is important: On Product and on Process level
  - Process Improvement is crucial
  - Feedback minimizes Ambiguities

- Interrupting and Reflecting on the process helps to improve it!

Agile Values

1. Redefined roles for developers, managers, and customers
2. No ”Big Upfront” steps
3. Iterative development
4. Limited, negotiated functionality
5. Focus on quality, understood as achieved through testing

[Mey2014]

Agile Principles – Revised list

(according to [Mey2014])

Organizational

1. Put the customer at the center.
2. Let the team self-organize.
3. Work at a sustainable pace.
4. Develop minimal software:
    1. Produce minimal functionality.
    2. Produce only the product requested.
    3. Develop only code and tests.
5. Accept Change

6. Reflect regularly and improve continuously!

Technical

1. Develop iteratively:
    1. Produce frequent working iterations.
    2. Freeze requirements during iterations.
2. Treat tests as a key resource:
    1. Do not start any new development until all tests pass.
    2. Test first.
3. Express requirements through scenarios.