

---

# Agile Principles / Miniature (DIT191 / EDA397)

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

# Organizational

---

- Thursday
  - Customer meeting
- Exam date
  - Jun-2<sup>nd</sup>, 2pm

# Agenda today

- Course overview
- Miniature
- Agile Principles revisited



# Sprint 1: Getting started



[http://commons.wikimedia.org/wiki/File:Sprint\\_01.jpg](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	Sprint 3
Sprint 2	Refactor a program/design	
	Be member of agile team	
	Incremental planning using user stories	

# 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: 5 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: 5 minutes

# Retrospective of Applied Strategy

---

## What did work well?

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

## What should we change?

- Introduce Integration Management
- More and faster feedback

## What did not work well?

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

# 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 should we change?

## What did not work well?

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

# 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 Manifesto

## Manifesto for Agile Software Development



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.

<http://agilemanifesto.org>

- Began as a provocation: Plan-driven development did not save the Software world...
- Now a very serious movement, well adapted in industry.
- There are a couple of established agile methods: How to integrate these values in everyday software development

# Agile Principles

---

1. Early and continuous delivery of valuable software
2. Welcome changing requirements, even late
3. Deliver working software frequently
4. Business people and developers must work together
5. Build projects around motivated individuals
6. Face-to-face communication is most effective and efficient
7. Working software is the primary measure of progress
8. Sustainable development
9. Continuous attention to technical excellence and good design
10. Simplicity is essential
11. Self-organizing teams
12. Regular reflection