User-centered System Design

Agile
Methods - what are they? Why do we have them?
Failed IT projects

1994
- Success: 16
- Cancelled: 31
- Partly failed: 53

1996
- Success: 27
- Cancelled: 40
- Partly failed: 33

1998
- Success: 26
- Cancelled: 28
- Partly failed: 46

2000
- Success: 28
- Cancelled: 23
- Partly failed: 49
Why failed projects?

- Reasons?
  - Low usability
    - Not able to produce a good enough system?
  - Never delivered, or late
    - Not enough pressure/skill?
  - Missing functionality, or wrong functions
    - Not knowing what to do?
The selling game

Analysis:
needs

Req.

Contract

Analysis:
time & money

Customer  Developer
Agile: everything on this page is wrong

“Requirements engineering (RE) is receiving increasing attention as it is generally acknowledged that the early stages of the system development life cycle are crucial to the successful development and subsequent deployment and ongoing evolution of the system.

- As computer systems play increasingly important roles in organizations, it seems there is a need to pay more attention to the early stages of requirements engineering itself (e.g., [6]).”

Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering, Eric S. K. Yu, paper 1 for Neil
Learing from the best

- What is a successful system?
- Why is the reason it is good do you think?
  - Was this planned or by chance?
Are we A or B?
Agile

- A reaction against the "heavy" methods and failed projects
  - Too little feedback
  - Too much method
  - Too large systems
  - Too many functions
  - No one likes these methods
  - The methods don’t deliver, they fail

- Overall idea:
  - Minimize risk by developing software in short amounts of time.
Agile

- Main Entry: agile
  Pronunciation: 'a-j&l, -"jl(-&)l
  Function: adjective
  Etymology: Middle French, from Latin agilis, from agere to drive, act
  1: marked by ready ability to move with quick easy grace <an agile dancer>
  2: having a quick resourceful and adaptable character <an agile mind>
  - agilely /-j&(l)-IE, -"jl(l)-IE/ adverb
Key principles

- We cannot do as before. A change is needed.
  - **Individuals and interaction** over processes and tools
  - **Working software** over comprehensive documentation
  - **Customer collaboration** over contract negotiation
  - **Responding to change** over following a plan

- [http://agilemanifesto.org/](http://agilemanifesto.org/)
Agile Manifesto Principles, 1

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
Agile Manifesto Principles, 2

- Working software is the primary measure of progress.
- Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from **self-organizing teams**.
- At regular intervals, the team reflects on how to become more effective, then **tunes and adjusts** its behavior accordingly.
Is this a development process?
Also a tool

- "Make sure there are whiteboards and coffee corners all over the building." (IBM)
What is Agile development?

- Framework, of values and principles

- FDD
- Agile Modeling (Ambler)
- eXtreme Programming (Beck)
- Crystal (Cockburn)
- Adaptive Software Development (Highsmith)
- Scrum
- DSDM (Stapledon)
Scrum

- Ken Schwaber in Financial Times: Scrum is different, he says, as it "takes people out of cubicles and puts them in a room with whiteboards."

http://www.controlchaos.com/module/FT.php

- Observation
  - SW development can be very different, in different places
Scrum

- Takeuchi and Nonaka, 1986
- The paper that started it all:
  - The new new product development game

"The rules of the game in new product development are changing. Many companies have discovered that it takes more than the accepted basics of high quality, low cost, and differentiation to excel in today's competitive market. It also takes speed and flexibility."
Characteristics of Scrum

- Observation:
  - *Using small, cross-functional teams historically produces the best results*

- Based on a rugby metaphor…
  - A **Scrum Master** acts as project manager/buffer to the outside world
  - **Product Owner**, represents stakeholders
  - **Team**, developers (less than 10)
Scrum actions

- **Sprint**, 30-day iteration
  - "backlog" of requirements, decided at the scrum
    - Managed by Product Owner
- **Daily meeting, called a "scrum"**
  - Starts precisely on time with team-decided punishments if late
  - Only "pigs" may speak (pigs and chickens)
  - The meeting is time boxed at 15 minutes; all attendees stand up
  - The meeting should happen at the same location and same time every day
- **All pigs say:**
  1. What have you done since yesterday?
  2. What are you planning to do by tomorrow?
  3. Do you have any problems preventing you from accomplishing your goal?
Sprint

- A **Product Owner** compiles all the changes planned for the product and prioritizes the possible functionalities.
- The result of the Product Owner’s work is a **Product Backlog** – a to-do list that is constantly reprioritized. Before each Sprint, the highest prioritized goals are transferred to a **Sprint Backlog**.
- Together with a user, the project members form a **Scrum Team** consisting of 5–9 people. During discussions with the Product Owner, the goal of the Sprint is determined and the prioritized functionality is broken down into detailed tasks. The team is self-organized and the members have a joint responsibility for the results.
- The **Scrum Master** coaches the development team, removes any possible impediments and constantly works to ensure that the team has the best possible circumstances for realizing the goals fixed for the Sprint.
Reading

- Scrum in 5 minutes

- Ken Schwaber, co-developed the Agile process, Scrum. Video from Google Tech Talks:
At the same time...

- OOP is different from traditional procedural programming
- The business requirements were changing
  - Time-to-market important
  - Design-driven market
  - Need to show progress
  - Short product life expected
- OOP-people see the successful patterns in problems like this
XP

- A *heavyweight methodology* has many rules, practices, and documents. It requires discipline and time to follow correctly.
- A *lightweight methodology* has only a few rules and practices or ones which are easy to follow.
Extreme Programming

- A deliberate and disciplined approach to software development
- To be more agile than traditional methods
- Create better software
  - Stresses customer satisfaction
- Emphasizes team work
Main aim of XP is to reduce the cost of change

![Graph showing cost over time, comparing 'Then' and 'Now' scenarios.](chart.png)
5 values of XP

- Communication
- Simplicity
- Feedback
- Courage
- Respect
Communication tools?

Communication Effectiveness

Richness of communication channel

- Paper
- 2 people on email
- Videotape
- 2 people at whiteboard
- 2 people on phone
Communication

- All communication must be rapid
  - No formal documentation
- Frequent communication
  - One single room for all
- Small note paper
  - Small design
Simplicity

- Design for what you know
- Do not plan ahead!
  - No “what if…”
  - No big design up front
  - *What most people find difficult about XP*

- A simple design…
  - Can easily be understood
  - Has higher quality, because it is stable
  - Is easier to change
  - Is faster to implement, has less errors
What exit is the right one?
Feedback

- From system
  - Unit testing: test first, then code

- From customer
  - Always present in room, takes part in acceptance tests

- From team
  - Direct time estimates on new functionality
User stories

- Something a user wants the system to do. Testable.
User stories
Courage

- Not to do more than required
- To be able to **re-factor** code fast enough
- To **review** other’s code (and your own)
- Knowing when to **throw away a bad solution**

**Issues**
- No place to hide!
- This is quite a big problem
Respect

- Not to break other’s code
- Aim for high quality
- Be motivated and loyal
What’s extreme?

- If tests are good, test all the time (start with Unit tests)
- If code review is good, always do reviews (pair programming)
- If iteration is good, deliver often
- Simplicity, do the simplest possible solution
- Users, have them in the same room
XP’s 12 activities

- Fine scale feedback
  - Pair programming
  - Planning Game (once per iteration, typically once a week)
  - Test driven development
  - Whole team (everyone needed is there)

- Continuous process
  - Continuous Integration
  - Refactoring or Design Improvement
  - Small Releases (called “builds”, alpha)

- Shared understanding
  - Coding Standards
  - Collective Code Ownership
  - Simple Design
  - System Metaphor

- Programmer welfare
  - Sustainable pace
User stories
Is Agile user-centered?

- No:
  - Focus on code
    - No explicit way to handle usability
  - Involves users, maybe not the right ones, not in the right environment
  - Hard for users to speak about their needs
  - Small increments – no overall view of the situation and the interaction
Is Agile user-centered?

- Yes:
  - User stories, by users
  - Direct communication
  - Simple representations
  - Iterations
  - Continuous testing and delivery
  - Change is welcome
  - About people, not processes
Reading

- Agile Modeling and eXtreme Programming
  - [http://www.agilemodeling.com/essays/agileModelingXP.htm](http://www.agilemodeling.com/essays/agileModelingXP.htm)

- Is Design Dead? Martin Fowler
  - [http://martinfowler.com/articles/designDead.html](http://martinfowler.com/articles/designDead.html)

- William Hudson: Adopting UCD within an agile process
  - [http://www.syntagm.co.uk/design/articles/ucd-xp03.pdf](http://www.syntagm.co.uk/design/articles/ucd-xp03.pdf)

- Larry Constantine: Process Agility and Software Usability