Agile & Scrum Starter

Software Engineering II
WS 2020/21

Enterprise Platform and Integration Concepts

Image by Brick 101 on flickr: https://www.flickr.com/photos/fallentomato/38066831102/ (CC BY-NC 2.0)
1. The Case for Agile
2. The Scrum Process
3. Scaling Scrum
How Projects Fail

- Delivering late
- Delivering over budget
- Delivering the wrong thing
- Unstable in production
- Costly to maintain
Why Projects Fail

- Smart people trying to do good work
- Stakeholders are well intended

Process in traditional projects

- Planning
- Analysis
- Design
- Code
- Test
- Deploy

- Much effort for
  - Documents for formalized hand-offs
  - Templates
  - Review committees
Why Projects Fail

“The later we find a defect, the more expensive it is to fix it!”

Does front-loading a software development process make sense?

Reality shows:
- Project plans are wonderful
- Adjustments & assumptions are made during analysis, design, code
- Re-planning takes place
- Example: Testing phase at the end
  - Tester raises a defect
  - Programmer claims he followed the specification
  - Architect blames business analyst etc.
  - Exponential cost
Why Projects Fail

- People are afraid of making changes to project plan
- Unofficial changes are carried out
- Documents get out of sync
- ...

Again, why do we do that!?  
To minimize the risk of finding a defect too late...
A Self-Fulfilling Prophecy

- Traditional front-loaded process to minimize costs of change
  - Project plan
  - Requirements specification
  - High-level design documents
  - Low-level design documents
  - Idea: Specify everything, then execute
- This process can cause exponential costs of change
  - A self-fulfilling prophecy

*This makes sense for a bridge, ship, or a building but* software (and Lego) are easy to change!
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/
How Agile Methods Address Project Risks

No longer late or over budget
- Tiny iterations
- Easy to calculate budget
- High-priority requirements first

No longer delivering the wrong thing
- Strong stakeholder communication
- Short feedback cycles
How Agile Methods Address Project Risks

No longer unstable in production
- Delivering each iteration
- High degree of automation

No longer costly to maintain
- Maintenance mode starting with Sprint 2
- Maintenance of multiple versions during development
The Cost of Going Agile

**Outcome-based planning**
- No complete detailed project plan

**Streaming requirements**
- A new requirements process

**Evolving design**
- No complete upfront design: flexibility required
- Emergent Design

**Changing existing code**
- Need for refactoring
The Cost of Going Agile

Frequent code integration
- Continuous integration

Continual regression testing
- Add nth feature; test n-1 features

Frequent production releases
- Organizational challenges

Co-located team
- Easy communication, keep momentum
Pros and Cons

- Short planning horizon
- No up-front design
- Stories instead of requirement documents
- Extreme ideology
1. The Case for Agile
2. The Scrum Process
3. Scaling Scrum
Scrum

- **Product Owner**
- **Team**
- **Scrum Master**

**Product Backlog** → **Sprint Backlog** → **Sprint** → **Working increment of the software**

- **Planning**
- **Daily Scrum**
- **24 h**
- **2 weeks – 1 month**
- **Review/Retrospective**

"Scrum process" by Lakeworks. (CC BY SA 3.0) via Wikimedia Commons
The Team

- Customer
- Product Owner
- Scrum Master
- Management
- Developers
Product Owner

Responsibilities

- **Customer** communication
  - Contact person for team
- Product Backlog
  - **User Stories**
  - Priorities
- Acceptance Criteria & Tests

http://agilemodeling.com/essays/productOwner.htm
Scrum Master

Responsibilities

- **Process** manager
  - Moderator in meetings
- Management communication
  - Remove *impediments*
- Enabler, not boss
Developers

Responsibilities

■ Communication
  □ Critically discuss all inputs
  □ Honestly share important information
  □ Represent team as expert

■ Sprint Backlog
■ Developing ;-)
List of work items

- Requirements (modification requests)
  - Features
  - Bug fixes
- Ordered/prioritized
In Scrum, requirements are often defined as user stories:

“As <role>, I want <feature> to <reason>”

Requirements need to fulfill INVEST properties:

- **I**
- **N**
- **V**
- **E**
- **S**
- **T**

In Scrum, requirements are often defined as **user stories**: “As <role>, I want <feature> to <reason>”

Requirements need to fulfill **INVEST** properties:

- I – Independent
- N – Negotiable
- V – Valuable
- E – Estimable
- S – Small
- T – Testable

Planning Meeting

Filling the sprint
- Estimate Backlog items
- Move items from Product to **Sprint Backlog**

Defining the work
- **Break down** Backlog items into tasks
- PO not required

Total time: 2 hours per week of sprint
For better planning, stories are broken down into tasks

Tasks should be **SMART**:

- **S** – Specific
- **M** – Measurable
- **A** – Achievable
- **R** – Relevant
- **T** – Time-boxed

List of tasks for a sprint

- Tasks are **signed-up** for, not assigned
- During the sprint
  - **No new features**
  - Team may change/add tasks
Daily Scrum Meeting

Status update
- Last achievements
- Next steps
- Problems

Max. 2 min per person

Discussions?
- Schedule subsequent expert’s meeting
Acceptance of Features

- Demo to PO
  - PO should be prepared
  - Optional: invite other stakeholders
- Comments by developers
Internal team evaluation

- PO not required
- Discuss process and problems
- Measure improvements
Potentially shippable increment

- Complete according to **Definition of Done**
  - Even if not actually released
- **No regrets** if project ended now
Scrum

Team
- Product Owner
- Scrum Master
- Developers

Meetings
- Planning
- Daily Scrum
- Review
- Retrospective

Artifacts
- Product Backlog
- Sprint Backlog
- User Stories
- Software Increment

November 6, 2020
Break