Other Agile Methods

Beatrice Åkerblom
beatrice@dsv.su.se

Scrum -- Introduction

- SCRUM is a loose set of guidelines that govern the development process of a product, from its design stages to its completion.
- SCRUM has been successfully employed by hundreds of different companies in many different fields, with outstanding results.
- There are many similarities between SCRUM and XP, but one of the major differences is that SCRUM is a fairly general set of guidelines that govern the development process of a product.
- SCRUM is often used as a "wrapper" for other methodologies, such as XP or CMM, to guide the overall process of development when using these other methodologies.

SCRUM Values

- The SCRUM values are derived from the Agile values of software development:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
The SCRUM Process

- The scrum process has 3 main phases:

<table>
<thead>
<tr>
<th>Pre-game</th>
<th>Mid-game</th>
<th>Post-game</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning &amp; high-level design</td>
<td>Develop &amp; Wrap</td>
<td>Closure</td>
</tr>
</tbody>
</table>

Daily SCRUM Meeting

- A 15-minute SCRUM meeting is held every day.
- The SCRUM Master asks the three questions, and all members of the team and interested parties take part and give feedback.
- The meeting should be held at the same place every time, so that people know where to go.

1. What have you accomplished since the last meeting?
2. Are there any obstacles in the way of meeting your goal?
3. What will you accomplish before the next meeting?

Sprint

- Before a sprint is begun, a Sprint Planning Meeting is held to determine what features are to be implemented in that sprint.
  - Develop the product further - implement, test, and document.
  - Wrap up the work - get it ready to be evaluated and integrated.
  - Review the work done in this sprint.
  - Adjust for any changes in requirements or plans.
Creating A Backlog

- The product owner compiles all the requests and specifications that are the basis of the changes of the product, e.g. new functions and bug fixes. After the goals have been defined, the entirety is broken down into segments.
- Each segment should in part create business value and in part be sub-deliverable.
- A prioritised list is made at the same time – the product owner personally makes the decisions at this point.
- When it is time to start a new Sprint, the product owner “freezes” the foremost items on the to-do list and summons the SCRUM team to a meeting.

Backlog

1. Product Backlog -- a repository for requirements targeted for release at some point. High level requirements with high level estimates provided by the product stake-holders.
2. Release Backlog - Requirements pulled from the product backlog and identified and prioritised for an upcoming release. Contains more details about the requirement and low level estimate which are usually estimated by the team.
3. Sprint Backlog - A result from each sprint planning is a backlog of requirements/sub-requirements estimated to be completed at the end of the sprint where the requirements from the release backlog are broken down into manageable chunks that can be accomplished typically in 8 - 16 hours.

The Scrum Team

- The SCRUM team consists of 2 groups:
  - the interested team, which consists of people who are interested, but who will not be doing the work
  - the working team -- people who are interested, and will be doing the work on the project

The Scrum Team, cont’d

- A SCRUM is a self-empowered team where everyone had the global view of the product on a daily basis
- The development team should perform as a sport team, every team member working independently but towards the same goal
- A team typically 6-9 working members, although SCRUM has been successfully used with more members
- The team members decide how the work is arranged and how assignments are distributed
- There are no set project roles – everyone should be able to swap tasks with another member
The Scrum Team, cont’d

- If a project is run in more than one SCRUM team, there should be people to act as bridges between the teams
  - attending meetings of more than one SCRUM team
  - act as a communication bridge between the teams

The team leader (SCRUM Master)

- The team's leader is called the SCRUM Master
- The SCRUM Master should be one of the members of the working team -- that is, he should be one of the people who is actually doing the work on the project
- The SCRUM Master measures progress, removes impediments, and leads the team meetings

The Product Owner

- The product owner represents the voice of the customer and ensures that the Scrum team works with the right things from a business perspective
- The product owner administers a product Back-log
- The product owner is often a customer, but can also be part of the internal organisation

SCRUM divided into 3 types by Jeff Sutherland

- Type A – isolated cycles of work
- Type B – Overlapping iterations
- Type C – All at once
Agile Model Driven Development (AMDD)

- **Agile Model Driven Development (AMDD)** is a collection of values, principles, and practices for modelling and documenting software that can be applied on a software development project in an effective and light-weight manner
- AMDD is based on proven software engineering principles
- AM is meant to be tailored into other, full-fledged methodologies such as XP or RUP, enabling you to develop a software process which truly meets your needs.

Values of AMDD

- **Communication** - Models promote communication between team members, as well as team members and project stake-holders
- **Simplicity** - Models are critical for simplifying both software and software process
- **Feedback** - Communicating your ideas through diagrams, gives quick feedback, enabling you respond
- **Courage** - You may need to change direction by either discarding or refactoring if some decisions prove inadequate
- **Humility** - The best developers have the humility to recognise that they don’t know everything, all project stake-holders have their own areas of expertise and have value to add to a project

Principles of AMDD

- **Assume Simplicity** - As you develop you should assume that the simplest solution is the best solution
- **Embrace Change** - Your project’s environment changes as your efforts progress, and as a result your approach to development must reflect this reality
- **Enabling the Next Effort is Your Secondary Goal** - Part of fulfilling the needs of your project stake-holders is to ensure that your system is robust enough so that it can be extended over time
- **Incremental Change** - You do not need to capture every single detail in your models, you just need to get it good enough at the time
Principles of AMDD, cont’d

- Maximise Stake-holder ROI - Stake-holders deserve to invest their resources the best way possible and not to have resources frittered away by your team
- Model With a Purpose - Ask yourselves why you’re creating the model in the first place and who you are creating it for
- Multiple Models - You potentially need to use multiple models to develop software because each model describes a single aspect of your software
- Quality Work - Nobody likes sloppy work. The people doing the work don’t because they can’t be proud of it, the people coming along later to refactor the work don’t because it’s harder to understand and to update, and the end users won’t because it’s likely fragile and/or doesn’t meet their expectations

- Rapid Feedback - Working closely with your customer, to understand the requirements, to analyse those requirements, or to develop a user interface that meets their needs, provides opportunities for rapid feedback
- Software Is Your Primary Goal - The primary goal is not to produce documentation, management artefacts, or even models; the goal of software development is to produce software
- Travel Light - Every artefact that you create, and then decide to keep, will need to be maintained over time

Practices of AMDD

- Active Stake-holder Participation - AM expands XP’s On-Site Customer practice to have project stake-holders (direct users, their management, senior management, operations staff, and support staff) actively involved in the project
- Apply the Right Artefact(s) - Each artefact has its own specific applications
- Collective Ownership - Everyone can work on any artefact on the project, if they need to
- Create Several Models in Parallel - Because each type of model has its strengths and weaknesses no single model is sufficient for your modelling needs

- Create Simple Content - You should keep your requirements, your analysis, your architecture, your design, as simple as you possibly can while still fulfilling the needs of your project stake-holders
- Depict Models Simply - The majority of the time you only require a subset of the diagramming notation available to you
- Display Models Publicly - This supports open and honest communication on your team since all current models are quickly accessible, as well as with your project stake-holders because you aren’t hiding anything from them
Practices of AMDD, cont’d

- **Iterate to Another Artefact** - When you are working on a development artefact and find that you are stuck, you should consider working on another artefact for the time being.
- **Model in Small Increments** - This will increase your agility by enabling you to deliver software into the hands of your users faster.
- **Model With Others** - Modelling is often done to understand or to communicate your ideas to others, or perhaps seeking to develop a common vision on your project.
- **Prove it With Code** - A model is an abstraction, to prove that it works you have to implement it in code.

Single Source Information - Information should be stored in one place and one place only, you should also model a concept once and once only, storing the information in the best place possible.

Use the Simplest Tools - If you're creating simple models, often models that are throwaways because if you are modelling to understand you likely don’t need to keep the model, then you likely don’t need to apply a complex modelling tool.

Agile Model Driven Development (AMDD)

- **Initial Requirements Modeling (IEM)**
- **Initial Architecture Modeling (IEM)**
- **Model Refining (MRC)**
- **Implementation (IMPL)**
- **Cycle 1: Development**
- **Cycle 2: Development**
- **Cycle 3: Development**

Agile models:
- Fulfil their purpose
- Are understandable
- Are sufficiently accurate, consistent and detailed
- Provide positive value
- Are as simple as possible

Agile models are just barely enough!
Agile Models

AMDD Documentation

- Travel light – You need far less documentation than you think
- Agile documents:
  - Maximise stake-holder investment
  - Are concise
  - Fulfil a purpose
  - Describe information that is less likely to change
  - Describe “good things to know”
  - Have a specific customer and facilitate the work efforts of that customer
  - Are sufficiently accurate, consistent, and detailed
  - Are sufficiently indexed

AMDD Documentation

- Valid reasons to document:
  - Your project stake-holders require it
  - To define a contract model
  - To support communication with an external group
  - To think something through

Communication Modes

- Always Strive to Use the Most Effective Approach
**Active Stake-holder Participation**

- The stake-holders are the experts, shouldn’t they model?
- Project stake-holders should:
  - Provide information in a timely manner
  - Make decisions in a timely manner
  - Actively participate in business-oriented modelling

**Model With Others**

- The modelling equivalent of pair programming
- You are fundamentally at risk whenever someone works on something by themselves
- Several heads are better than one

**AMDD Requirements**

```
Developer
```

- Identify idea or suggestion
- Discuss potential requirement
- Estimate
- Model and document
- Prioritise

```
User
```

**Effectiveness of Requirements Gathering Techniques**

<table>
<thead>
<tr>
<th>Joint Application Design (JAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus Groups</td>
</tr>
<tr>
<td>Face-To-Face Interviews</td>
</tr>
<tr>
<td>Observation</td>
</tr>
<tr>
<td>Electronic Interviews</td>
</tr>
<tr>
<td>Legacy Code Analysis</td>
</tr>
<tr>
<td>Reading</td>
</tr>
<tr>
<td>Collaborative Interaction</td>
</tr>
<tr>
<td>Restricted Interaction</td>
</tr>
</tbody>
</table>

Copyright 2001 Scott M. Palmer
Agile Requirements Gathering

- High level requirements – what should the system as a whole really do?
  - Usage model -- a collection of user stories
  - Initial domain model – class responsibility collaborator (CRC) cards, a slim UML class diagram
  - User interface model – screen sketches, or maybe a user interface prototype
- What level of detail do you actually need?
  - Just “enough” to get the big picture
  - The initial requirements are later analysed, on a just-in-time basis

What level of detail do you actually need?
- Just “enough” to get the big picture
- The initial requirements are later analysed, on a just-in-time basis

Requirements Management

- Changing requirements are a competitive advantage if you can manage them

Relative Effectiveness of User Representatives

<table>
<thead>
<tr>
<th>Effectiveness</th>
<th>Actual Stakeholder</th>
<th>Product Manager</th>
<th>Business Analyst as User</th>
<th>Personas</th>
</tr>
</thead>
</table>

End of Today’s Lecture

Thanks for your attention!