Scrum, Creating Great Products & Critical Systems

What to Worry About, What’s Missing, How to Fix it

Neil Potter
The Process Group
neil@processgroup.com
processgroup.com
Agenda

• Scrum / Agile Overview
• What to Use From The Agile Manifesto
  – All of it / some of it?
• Definition of Scrum
• Scrum Risks to Look Out For and What to do About Them
  – Mistaking speed for progress
  – 1-liner requirements (the devil is in the details)
  – Missing architecture / design
  – Missing final system test / validation
  – Missing configuration management
  – Missing risk management
  – Missing process assurance (known as ScrumBut)
Definition of Agile and Scrum

• Agile
  – “Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.”

  Wikipedia.

• Scrum
  – Scrum is an agile process for software development. Scrum consists of predefined milestones and events that scope, estimate, plan and status the project.
Agile Manifesto

“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.”

What to Use?

“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

Both, based on goals and challenges

That is, while there is value in the items on the right, we value the items on the left more.”

Definition of Scrum

1 day

2-4 week Sprint

Potentially Deliver

Review Backlog
Sprint Planning
Analysis
Design
Code
Test
Sprint Review
Sprint Retrospective

Team defined tasks

Product Backlog
Release Planning
Sprint Planning
Analysis
Design
Code
Test
Sprint Review
Sprint Retrospective

Potentially Deliver

Review Backlog
Sprint Planning
Analysis
Design
Code
Test
Sprint Review
Sprint Retrospective

2-4 week Sprint

© Copyright 2009-2014 The Process Group. All rights reserved. www.processgroup.com
### Scrum Positives

<table>
<thead>
<tr>
<th>Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work is chunked in 2-4 week increments (Sprints).</td>
</tr>
<tr>
<td>Scope changes are managed using the Backlog, iteration planning and the rule “No changes during a sprint.”</td>
</tr>
<tr>
<td>Small (1-4 week) iterations create team momentum and early feedback on progress and technical solutions.</td>
</tr>
<tr>
<td>Scrum process can be learned and used in less than 2 days.</td>
</tr>
<tr>
<td>Daily Standup meetings and Burndown charts provide quick and easy project status.</td>
</tr>
<tr>
<td>Engineering practices (often used in Agile) can reduce risk:</td>
</tr>
<tr>
<td>• Automated unit tests.</td>
</tr>
<tr>
<td>• Pair programming (2 developers working together).</td>
</tr>
<tr>
<td>• Test driven development (write a test first).</td>
</tr>
<tr>
<td>• Continuous integration (e.g., daily builds to find problems).</td>
</tr>
</tbody>
</table>
Scrum Risks to Look Out For and What to do About Them
Does 100% Scrum do What You Need?

- Plan for software aspects of certification
- Software development plan
- Software verification plan
- Software configuration management plan
- Software quality assurance plan
- System requirements
- Software requirements standards
- Software design standards
- Software code standards
- Software design description
- Source code / Executable object code

- Review of all requirements, design and code
- Testing of executable object code
- Code coverage analysis
- Unit testing
- Integration testing
- Black-box and acceptance testing
- Software quality assurance records
- Software conformity review
- Traceability
- Independent testing

If your team is full-Scrum – know what is missing
Mistaking Speed For Progress
Speed vs. Progress

With no:

- **Design:**
  - Architecture + design notes
- **Peer-reviews to find defects**
- **Checks for interface for errors**
- **System test**
- **Analysis of verification results:**
  - e.g., defect density, pass/escape rate, cause

Great product?

Implement yourself into a corner?
1-Liner Requirements
(The Devil is in The Details)
## Requirements / Backlog

<table>
<thead>
<tr>
<th>Priority</th>
<th>User Story</th>
<th>Estimate (Points)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>As an account owner, I want to withdraw cash</td>
<td></td>
</tr>
<tr>
<td></td>
<td>As an account owner, I want to deposit cash</td>
<td></td>
</tr>
<tr>
<td></td>
<td>As an account owner, I want to deposit check(s)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>As an account owner, I want to deposit foreign check(s)</td>
<td></td>
</tr>
</tbody>
</table>

Typical format: As a <user>, desired action

1-liners can be ambiguous and lead the developer to guess
Sample Use Case for an ATM
Automated Teller Machine

Elicitation question

Name: Withdraw Cash
What does the user want to do?

Actor: Account Owner
Who?

Description:
The user withdraws a specific amount of cash from a specified account.

Trigger: Account Owner selects Withdrawal action
What initiates it?

Preconditions: [What state is the user / system before the event?]
1. The Account Owner is logged in to the ATM.
2. The Account Owner has at least 1 account with a positive balance.
3. The ATM contains cash.

Postconditions: [What state is the user / system after the event?]
1. The requested amount of cash has been dispensed.
2. The account balance is reduced by the amount withdrawn plus any fees.
3. The ATM cash balance is reduced by the amount withdrawn.

Priority: High
Sample Use Case for an ATM

Normal flow: [What is the ideal flow?]

<table>
<thead>
<tr>
<th>Actor Actions</th>
<th>System Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Select Withdrawal action.</td>
<td>2. Display user’s accounts.</td>
</tr>
<tr>
<td>3. Select desired account.</td>
<td>4. Ask user to enter amount.</td>
</tr>
<tr>
<td>5. Enter desired amount.</td>
<td>6. If amount is okay, dispense cash and debit account.</td>
</tr>
<tr>
<td>7. Remove cash from dispenser.</td>
<td></td>
</tr>
</tbody>
</table>

• Alternative Flow: [Any alternatives?]
  – Step 4: display list of common amounts, let user select one

• Exceptions: [Conditions the system needs to deal with?]
  – Step 6: amount is not a multiple of $20.00
  – Step 6: amount exceeds $400
  – Step 6: amount exceeds account balance
  – Step 6: amount exceeds cash available in ATM
Missing Architecture / Design
Missing Final System Test / Validation
Design

• Why design?
  – Identify potential **sticky areas** that need to be investigated.
  – Find **errors** earlier.
  – Clarify and **communicate** concepts and definitions to others.

• Design example:
  – **Architecture, interfaces, data definitions, constraints**
    » e.g., design for testing, expansion, security, portability and technology.
    » Textual design notes (in backlog or code header), modeling languages, event tables, flow/timing diagrams, pseudo code.

• You might consider:
  - Sprint N: 80% design, 20% coding/test.
  - Sprint N+1: 60% design, 40% coding/test.
  - Sprint N+2: 40% design, 60% coding/test.
Final System Test / Validation

Sprint

| 70% R 30% | 70% R 30% | 70% D 30% | 70% D 30% | 70% C 30% | 70% C 30% | 70% C 30% | 40% T 60% | 50% T 50% | 60% T 40% | 70% T 30% | 80% T 20% | 90% T 10% | 100% T 0% |

- 70% Requirements
- 30% Other (Requirements, Design, Code, unit Test, system Test, acceptance Test?)

processgroup.com/monthlytidbits.html#tidbit12
Missing Configuration Management
What Version of _ Are You Working On?

What outputs do you want to:
- Not lose
- Share (the same version)
- Know the history of
- Find easily with confidence

Consider:
- Versioning / labeling (baseline)
- Tracking changes after a baseline
- Making sure everyone has version X
- Tools: SVN, GIT, Windchill, SharePoint..
Missing Risk Management
Where to Add Risk Management

**Agile perspective**
High-risk features are worked on in earlier sprints.

**Revision:** Add risk management to:
- Release Planning
- Sprint Planning
- Daily Standups
## Risk Management

Action items will likely be more numerous and detailed

Status of action items

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>New operating system might not be stable</td>
<td>Unable to ship</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>Test OS more</td>
<td>Identify 2nd OS</td>
<td>Joe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System communication problems with XYZ</td>
<td>Feature x unavailable</td>
<td>8</td>
<td>9</td>
<td>72</td>
<td>Develop sys interface doc</td>
<td>Add replan milestone</td>
<td>Kim</td>
<td></td>
<td></td>
</tr>
<tr>
<td>We may not have the requirements right</td>
<td>Delay next project + no usage of demo</td>
<td>9</td>
<td>6</td>
<td>54</td>
<td>Prototype of UI Use Case style requirements</td>
<td>Incremental release Limit distribution</td>
<td>Lois</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements may change late in the cycle</td>
<td>New defects</td>
<td>7</td>
<td>7</td>
<td>49</td>
<td>Prototype top 10 requirements</td>
<td>Limit distribution</td>
<td>Joe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database S/W might be late</td>
<td>Revenue delay</td>
<td>4</td>
<td>8</td>
<td>32</td>
<td>Check with supplier</td>
<td>Help supplier</td>
<td>Pete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database expert might leave</td>
<td>Schedule delay</td>
<td>2</td>
<td>10</td>
<td>20</td>
<td>Make sure Jim is happy</td>
<td>Earmark Fred</td>
<td>Jane</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>327</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Missing Process Assurance (Known as ScrumBut)
ScrumBut

• “I do Scrum, but I don’t do:
  – User stories
  – Automated testing
  – Sprints
  – Burndown charts
  – Retrospectives

What does your team do?
Are they Agile or Agile-declared?
Adding Gates and Governance

- Establish **gates at the end of each, or several, sprints:**
  - Negotiate **upfront** what will be available at the gates (some design, some code, some tests).
  - Work with **process QA staff** early so that there is agreement as to what will be audited and when.
Summary

• Scrum was initially designed for **small co-located teams**:
  – It does not automatically contain everything you want.

• **Not all Agile/Scrum teams actually do Agile/Scrum**:
  – Ask them what they do!

• **Scrum teams might see little value in “documentation”**:
  – Use documentation to address communication and memory challenges.
  – Capture information in current tools or team workflow tools (e.g., RallyDev, Jira).

• **Don’t be scared to add practices (and not break Scrum)**:
  – **Add practices to sprints** to increase quality & reduce risk:
    » e.g., Requirements, architecture, design – WHILE creating **working code**.
Q&A
Additional Material
### Example Risk Mitigation Actions

<table>
<thead>
<tr>
<th>Potential Problem Using Scrum</th>
<th>Bring Your Own ____</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambiguous 1-line user stories.</td>
<td>Requirements Analysis</td>
<td>Elicit requirements. Use Cases. Peer review requirements.</td>
</tr>
<tr>
<td>Little/no design. Design flaws. Architecture difficult to modify.</td>
<td>Design</td>
<td>Architecture diagram. Larger % of design activity in initial sprints: – Sprint 1: 80% design, 20% code. – Sprint N: 5% design, 95% code.</td>
</tr>
<tr>
<td>No system-level testing in sprints leading to system defects.</td>
<td>System testing</td>
<td>Add system testing as a % of later sprints. Add a test/cleanup sprint.</td>
</tr>
</tbody>
</table>
### Example Risk Mitigation Actions

<table>
<thead>
<tr>
<th>Potential Problem</th>
<th>Bring Your Own ____</th>
<th>Example</th>
</tr>
</thead>
</table>
| No version control of documents or code leading to loss or incorrect versions being used. | Configuration management | Naming conventions.  
CM tool / storage definition.  
Access / edit controls. |
| Scrum But:  
“We do Scrum, but we don’t have a backlog, Product Owner, sprint backlog or do daily standup meetings.” | Process Assurance | Independent member of one Scrum team audits the practices of another Scrum team.  
Monthly management review of Scrum activities and progress. |
| Surprises (technical, personnel, supplier) that could have been foreseen and mitigated. | Risk Management | Add a risk assessment to the Sprint planning meeting.  
Track risks in daily standup meeting. |
| Scaling for large teams, or many small teams that are dependent upon each other.  | Program management | Add system engineering / program management. |
Scrum - Under the Hood

- 70% Requirements
- 30% Other (Requirements, Design, Code, unit Test, system Test, acceptance Test?)
Hybrid or Separate

• **All teams adopt some Agile / Scrum practices:**
  – Increments of 2-4 weeks (for whatever the work is).
  – Potentially deliverable components early on.
  – Burndown charts (based on points and effort).
  – Continuous integration.
  – Pair programming.
  – Daily standup meetings.

• **Alternative - keep traditional & Scrum alive:**
  – Define an architecture that compartmentalizes Scrum and non-Scrum approaches
    » Scrum for high-risk, unknown sections.
    » Existing approach for everything else.
    » The two approaches meet at the system testing phase

[Mike Cohn - Succeeding with Agile]
Incremental Life Cycle

- **Hybrid option:**
  - Simplify existing deliverables from Waterfall.
  - Move to an incremental life cycle (a balance between Scrum and Waterfall).

![Diagram showing incremental life cycle phases: Planning, Prototype, Requirements, Architecture, Sprint 1, Sprint 2, Sprint 3. Phases are connected with arrows indicating flow and defined points to manage change.](image-url)
References

• Implementing Scrum (Agile) and CMMI Together. Process Group Post newsletter, March 2009.

• Adding Practices to Scrum to Achieve Your Goals (and comparison with CMMI Level 3).


• Agile Methods And Safety-critical Software, Peter Gardner, Silver Atena (YouTube).