Mature agile development using HP Quality Center

Gerald Heller
software process optimization

Vivit TQA webinar
September 22, 2009

Using QC with agile practices

Agile fundamentals
Expectations & challenges
Quality Center for agile product development
Experiences & conclusions
The 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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Craig Larman
Jim Highsmith
Tom Cagley
Ron Jeffries
Kent Beck
Robert C. Martin
Steve McConnell
Ken Schwaber
Jeff Sutherland
Dave Thomas

Agile methods

xP
Lean
Scrum
DSDM
ASD
Crystal
Agile terminology

- Sprint
- Test driven development
- Velocity
- Pig
- Lean-agile
- Chicken
- Backlog
- Burn down
- Constant refactoring
- Daily Scrum
- Product owner
- Retrospective
- Pair programming
- Scrum Master

http://www.netobjectives.com/glossary

Agile terminology – roles

- Sprint
- Test driven development
- Velocity
- Pig
- Lean-agile
- Chicken
- Backlog
- Burn down
- Daily Scrum
- Constant refactoring
- Retrospective
- Product owner
- Scrum Master

http://www.netobjectives.com/glossary
Agile terminology – work to be done

- Sprint
- Test driven development
- Velocity
- Pig
- Lean-agile
- Chicken
- Burn down
- Constant refactoring
- Daily Scrum
- Product owner
- Retrospective
- Pair programming
- Scrum Master

Backlog unit

http://www.netobjectives.com/glossary
Backlog management

Top Priority

Product Backlog

Release Backlog

In-Out Borderline

Low Priority

Backlog management

Top Priority

Product Backlog

Release Backlog

In-Out Borderline

Low Priority

Sprint Backlog
Backlog management

Scrum overview

Scrum is the most dominant agile method
Expectations for agile

• Focus on highest customer value
• Be able to work on incomplete information
• Travel light
• Incremental delivery
• Visibility into progress for all parties
• Quality results as early as possible

Agile development – state of practice

• Experimentation phase comes to an end
• Signals of maturity
  – Siemens Nokia Networks, SAP, HP, AOL, …
• Standardization initiative: IEEE 1648
  – recommended practice for agile development
• More and more tools support agile
• Agile merged with existing practices
Tool expectations for agile

- Support key agile practices
- Can be tailored to needs of all stakeholders
- Provide progress and value reporting
  - Based on iterations
  - Based on release
- Span entire development lifecycle

QC releases and cycles
QC releases and cycles – details

Structure of an agile project

Multi-backlog management to enable Scrum-of-Scrum

First level shows backlog and cycle folders
Product backlog

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Edit</th>
<th>View</th>
<th>Features</th>
<th>Analysis</th>
</tr>
</thead>
</table>

Filter: [Release] [Requirements] [Advanced order system] [User stories] [S].

<table>
<thead>
<tr>
<th>Name</th>
<th>Backlog Position</th>
<th>Target Cycle</th>
<th>Target Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced order system</td>
<td>———</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>Story Kernel</td>
<td>———</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>User stories</td>
<td>———</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>S. Backend</td>
<td>———</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>S. Desktop UI</td>
<td>———</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>Sprint 1</td>
<td>———</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US3</td>
<td>1</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US2</td>
<td>2</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US1</td>
<td>3</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US11</td>
<td>4</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US12</td>
<td>5</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US13</td>
<td>6</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US14</td>
<td>7</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US22</td>
<td>8</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>US16</td>
<td>9</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>Sprint 2</td>
<td>———</td>
<td>———</td>
<td>———</td>
</tr>
<tr>
<td>Sprint 3</td>
<td>———</td>
<td>———</td>
<td>———</td>
</tr>
</tbody>
</table>

User Story – definition

- A **User Story** is a software system requirement formulated as one or two sentences in the everyday or business language of the user
  - User stories are used with Agile software development methodologies for the specification of requirements
  - A User Story must be small enough to be completed in one cycle
  - A User Story is an informal statement of the requirement as long as the correspondence of acceptance testing procedures is lacking

Wikipedia, July 2009
User Story – sample properties

• **Name**
  - Meaningful to give an idea what the User Story is about

• **Description**
  - Short, understandable by all team members
  - Supplied with acceptance criteria

• **Priority**
  - Supplied by backlog position

• **Owner**
  - Responsible to manage the User Story, ensure it’s properties are set correctly, it is broken down into tasks and tasks are assigned to owners

• **Estimated effort**
  - Macro level estimate that is done before it is broken down into tasks

• **Status**
  - Describes the current situation of the User Story

---

User Story - details

![Diagram of User Story details](image)

- Modeled as a requirement type
- High-level user story estimate
Two level planning (at least)

Before Sprint

User story size

Within Sprint

Task size

Cycle planning – top down

Initial high-level estimates based on User Stories prior to Sprint start

User story estimates
Task – sample properties

- **Name**
  - Meaningful to describe the task

- **Description**
  - If the name is not self-descriptive, explain here in more detail what the task is about

- **Owner**
  - Responsible to implement the task

- **Effort Estimates**
  - Original Estimated Effort: detailed original estimate for the task, will never be updated once set
  - Current Estimated Effort: overall effort estimate for the task; initially set to the original estimated, but is updated as the owner learns more about the task
  - Actual Spent Effort: how much is already spent on the task, should be updated daily
  - Remaining Effort: is calculated

- **Origin**:
  - How was the task added? As part of the task breakdown of the User Story (Planned) or discovered during the work on the User Story (Discovered).

---

**Task – in QC**

Task modeled as requirement type
Cycle planning – bottom up

Detailed estimates based on tasks during Sprint

Effort estimation and calculation

- **Release Level**
  - Original Estimated Effort: Manual Entry
  - Current Estimated Effort: Calculated (sum of Cycles)
  - Actual Spent Effort: Calculated (sum of Cycles)
  - Remaining Effort: Calculated (sum of Cycles)

- **Cycle Level**
  - Original Estimated Effort: Manual Entry
  - Current Estimated Effort: Calculated (sum of User Stories)
  - Actual Spent Effort: Calculated (sum of User Stories)
  - Remaining Effort: Calculated (sum of User Stories)

- **Story Level**
  - Original Estimated Effort: Manual Entry
  - Current Estimated Effort: Calculated (sum of Tasks)
  - Actual Spent Effort: Calculated (sum of Tasks)
  - Remaining Effort: Calculated (sum of Tasks)

- **Task Level**
  - Original Estimated Effort: Manual Entry
  - Current Estimated Effort: Manual Entry
  - Actual Spent Effort: Manual Entry
  - Remaining Effort: Manual Entry

High-level User Story estimate at 15 hours
Detailed estimates based on tasks sum up to 17
Effort aggregation

Aggregated effort for estimates and remaining work

All items are assigned to a specific release and cycle

Sprint burn-up

Effort to burn down

Current Estimated Effort  Actual Sprint Effort
User stories to test

A) Create test from User Story

B) Link test to User Story

Check filter capabilities for best results

User stories
Built-in coverage by cycle

2 Filters need to be aligned! Coverage and View Filter

Quality User Story – fixing defects
Experiences

• Tool customization must match team’s experience with agile
• Team experience with agile has a strong impact on required tool customization
  – Start simple
  – Grow structure and process
• Harmonize across teams
• QC provides a solid base for agile development
• QC customization is prerequisite for agile success

QC customization for agile

Easy
– Requirement types: User Story and Task
– Custom attributes
– Role support
– Sprint support

Medium
– Reporting
  • Sprint Burn-down, burn-up
  • Release Burn-down, burn-up
  • Effort per engineer
  • Effort calculation

Challenging
– Backlog organization
  • Ordering based on unique priority
  • Automatic inheritance of values from container object
  • Aggregation of data
  • Scrum of scrum

need for thorough design of customized solution increases
Thank you!

Questions, feedback: contact me
Further information & news: agileQC.net

Gerald Heller
software process optimization
gerald.heller@swq4all.de