Definition of Done
An Organizational Perspective

August 9, 2010

Topics to Cover

- Agreement on “Done”
  - Examples of Agreements
  - Organizational Need to “Standardize”
  - StoryoTypes
  - Summary
In the Front Burner (Sprint Backlog)

- The Team has an Agreement on “done” for each story
- This is an Agreement amongst the team (includes the PO, remember) that answers the question:
  - How will we (the Team) know that we are done with this story?
  - Must be verifiable by the Team
  - Can’t be dependent on the “kindness of strangers”
- The Team “owns” this Agreement – it can’t be agreed to for them
  - The Agreement is negotiated in “good faith” by the PO and the rest of the Team
  - The PO represents external stakeholders
- If the Team can’t accept the agreement
  - We must change the agreement so they can, or
  - Treat the story as an Epic (discussed later)
Externally Visible Scope vs Technical Debt

Acceptance Criteria
What we “Deliver”
Externally Visible

Definition of Done
Technical Debt
Invisible from Outside

Form of the Agreement

- These agreements have different forms for different kinds of stories (storytypes), but for a functional story I think there are three parts to an Agreement:
  - What will be verified to prove that the story provides the stakeholder-requested value?
  - What will be verified to prove that we have mitigated the risk of producing technical debt?
  - Other Agreements, like which SMEs we’ll talk to, the simplifying assumptions, what’s out of scope, etc

- We will discuss agreement for various StoryoTypes
  - Production (coding) stories
  - Analysis stories – produce Development stories
  - Business Support stories – non-development support of the Business
  - Chores – stories that have no explicit business value
Iron Triangle and “Doneness”

- Agile Planning is actually about balancing effort, scope, and technical debt
- The expected scope and debt is seldom documented in detail, but maybe if should be
  - I like to see an explicit DoneList so that we can “check it off”
- DoneList Agreement has two parts
  - Scope Side (acceptance), usually defined by tests, time boxes, etc
  - Debt Side (doneness), usually defined by inspections, process steps, etc
- The DoneLists are different for different storyotypes, and the tasks exist in order to get the DoneList completed

Topics to Cover

- Agreement on “Done”
  - Examples of Agreements
  - Organizational Need to “Standardize”
  - StoryoTypes
  - Summary
Seeing is Believing...

- Probably the best way to understand this concept is to see some examples
- In the following examples, we some some “complete” stories – stories that have been agreed to and accepted into the Front Burner (Sprint Backlog)
  - What we see here defines the “what” for the story
  - The “how” will be explored, defined, and implemented as the story is worked on
  - There could be more stuff than this, but we usually defer it until we have committed to actually doing the story
- First we’ll look at some stories of different types, and then we’ll discuss Epics

Sample Development Story

<table>
<thead>
<tr>
<th>Get List of Flights from CUTLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size:</strong> 8 SPs <strong>Type:</strong> [backbone]</td>
</tr>
<tr>
<td>As a &lt;flyer&gt; I want &lt;to have a list of flights that matches my itinerary&gt; so that &lt;I can choose one that works for me&gt;</td>
</tr>
</tbody>
</table>

**General:**
- Joe (SME) is the expert on CUTLASS
- Simplifying Assumptions: One Way, Single Leg, No Seat Selection, Single Passenger, Full Fare, No Luggage

**Acceptance:**
- Pass in an itinerary and get a list of Flights back

**Doneness:**
- Review Architectural Decisions with Team
- Design Review
- Review Functional Test Strategy
- Review Unit Tests
- Verify Tests passing on Development Machine
- Code Review
- Functional Tests Written
- Verify Tests passing on Integration Box
- Add Tests to Regression Test Suite

**Tasks**

- **Architecture and Design**
  - 32 hrs

- **Write Functional Tests**
  - 12 hrs

- **Code and Unit Test**
  - 80 hrs
Sample Analysis Story

**Analyze Shopping for Flights**

*Size: 2-day Timebox  Type: [analysis]*

As a `<developer>` I want `<some stories for "shopping for flights">` so that `<I'll have some work to do>`

**General:**
- Amir (Team Member) is the Coordinator

**Acceptance:**
- The Backbone Story is in the Backlog
- There is at least one validated (with the SMEs) story, based on this backbone story, ready to be "worked on"

**Doneness:**
- Identify SMEs (Subject Matter Experts) for "Shopping for Flights" and document in epic
- Meet with the SMEs and discuss the issues, document what you get in the Wiki
- Generate, and validate, the "backbone" version of this epic
- Have a meeting with the Team to discuss it...

**Tasks**

- Meeting with SMEs
  - 4 hrs
- Document in Wiki
  - 6 hrs
- Generate Production Story
  - 4 hrs

Sample Business Support

**Support Sales with Company ABC**

*Size: 2-day Timebox  Type: [bus spt]*

As a `<sales guy>` I want `<to sell the new capabilities to Company ABC>` so that `<we'll make some good money>`

**General:**
- Sandy (from Sales) is the Coordinator

**Acceptance:**
- Our Team gives Sandy 2 days of its time to support her sales efforts with Company ABC

**Doneness:**
- Bring Sandy and other sales guys up to speed on new capabilities
- Go with Sandy to see Company ABC to provide technical support
- Work with Sandy to make sure bid correctly describes the new capabilities

**Tasks**

- Brief Sales On New Capabilities
  - 2 hrs
- Meeting with Client
  - 8 hrs
- Work on Bid With Sales
  - 4 hrs
Sample Infrastructure Story

Install Copy of CUTLASS in Lab
Size: 8 SPs  Type: [enviro]

As a developer I want to have my own copy of CUTLASS to play with so that I can figure out how it works

General:
- Joe (SME) is expert on CUTLASS
- Sam (Team Member) will be Coordinator

Acceptance:
- CUTLASS is "up and running" in the lab

Doneness:
- Get CUTLASS Install from SirJeff
- Sam work with Joe to:
  - Set up clean machine
  - Install CUTLASS
  - Do Smoke Test to see if it works

Tasks

- Set up clean machine in lab
  8 hrs

- Install CUTLASS on new machine
  8 hrs

Discussion of Epics

- Epics are Items that the Team can’t commit to, for any reason (Complex, Unknown, Risky, or Big)
  - Has other Stories inside it
  - There is usually no definition of “done” for an epic, but
  - These inside stories have Agreements that can be committed to

- Example:
  - Epic: “I want page XYZ to render in < 1/10 sec because it’s too slow right now”
    - Epic because team can’t commit to 1/10 second (too risky)
  - Stories inside could be:
    - Do 4 hours worth of improvements to the rendering and measure to see how fast it is
    - Implement algorithm ABC to speed up the rendering
    - Etc

- Note that what the stakeholders get is NOT exactly what they want – it’s what the team can commit to…
Topics to Cover

- Agreement on “Done”
- Examples of Agreements
  - Organizational Need to “Standardize”
- StoryoTypes
- Summary

Scrum Teams Live Within an Organization

- That can provide constraints upon the Scrum Team
- The Team’s process is part of an overall process
  - The team owns what is not constrained
- What should an organizations constrain?
- What process issues are cross-cutting ones?
Organizational Issues

- Organizational Issues:
  - Common Codebase across Teams
  - Want to move people from Team to Team
  - Want commonality of “process”

- Scrum wants:
  - Self-organization and freedom for teams

- A good candidate for standardization across teams is “definition of done”
  - Cross-team integration issues
  - Makes it easier for people to move from team to team
  - But doesn’t micromanage the people themselves…

Cross-Team Integration

- Integration is always hard – probably biggest technical issue in most organizations
- Common Def’n of done helps integration issues
  - Know “what to expect” from the code you are integrating with
  - Know “what to expect” if you need to work with somebody else’s code
  - Know “what to expect” if you need to work with people from other teams
- Allows for cross-team Retrospections about what “done” should mean
  - Good for the codebase
  - Good for the organization
- Part of scrum already that Teams working on same Product should have same “definition of done”
Makes it Easier for People to Move

- A Common Definition of “Done” Leads to common development practices
  - So, at the “working together” level it makes it easier for people to move from team to team
  - Makes it easier to do technical training for an organization

- But, it gives the team flexibility
  - To adapt their own team dynamics to their team members
  - To add more restrictions to the “def’n of done” for their team

- And it’s not micro-managing
  - It’s more of a “what” thing – it’s telling people what they need to do…
  - It’s putting constraints on the hows, but not defining the hows

Topics to Cover

- Agreement on “Done”
- Examples of Agreements
- Organizational Need to “Standardize”
  - StoryoTypes
- Summary
**Categories of StoryoTypes**

- StoryoTypes
  - Stereotype of a story
  - Meszaros, "Using Storyotypes to Split Bloated XP Stories" 2004, XP/Agile Universe
  - Originally used to help in “analysis” or decomposing epics
  - I extend the concept beyond “coding” stories to all types of stories we find in scrum projects
  - There are various Categories of StoryoTypes that I use
    - Production (coding)
    - Analysis (finding stories)
    - Business Support (non-coding)
    - Chores (no immediate Business Value, but not Analysis)

**StoryoTypes and Agreements**

- The most important thing about a StoryoType is that it holds a “common” Agreement that can be reused
  - Common Process Steps
  - Common Constraints (usability, performance, etc)
  - It can also contain common tasks
- Each of these Categories has different “kinds” of Agreements
  - And each StoryoType within the Category has its own variation
- Each organization and Team develops its own catalog of these things over time
  - But I’ll present some guidance and “starter” info in the following slides...
  - And I’ll only do a few of them…
Production StoryoTypes Catalog

- [coding] – generic for writing code (many companies do this)
- Use Case Based StoryoTypes (from Meszaros)
  - [backbone]
  - [alt]
  - [beefup]
  - [interface]
- Other Production StoryoTypes
  - [perf]
  - [bug]
  - [cleanup]
  - [documentation]
- Mixins
  - [hack] – intentional hacking, always comes with a [cleanup] story
  - [arch-sig]

Use Case Based StoryoTypes

- Implemented by Story
- Conform to StoryoType
- Use Case
- Made up of Scenario
- Backbone
  - Part of the backbone
  - An alternative scenario
  - Improving a business rule in an existing scenario
  - Improving the interface for the UC
- Alternate
  - Many of them
  - Single thread
  - Always another one...
- Only one
  - Architecturally significant
  - Simplifying assumptions
  - Single thread
- The [backbone] stories have an architectural element to them
- Our PO's job is to do the stories in the "right" order to keep the value produced on the S-Shaped curve

**S-Shaped Curve for Adding Value for a Use Case**

<table>
<thead>
<tr>
<th>% of Effort Expended</th>
<th>Value Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0.1</td>
</tr>
<tr>
<td>40</td>
<td>0.2</td>
</tr>
<tr>
<td>60</td>
<td>0.3</td>
</tr>
<tr>
<td>80</td>
<td>0.4</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
</tr>
</tbody>
</table>

- Infrastructure
- Must haves
- Buffer, Nice to have, Rework
- Minimally releasable

**Basic [coding] StoryoType “doneness” Criteria**

- Design Review
- Review Functional Test Strategy
- Review Unit Tests
- Verify working on Development Machine
- Code Review
- Functional Tests Written
- Verify working on Integration Box, including Tests
- Add Functional Tests to Regression Test Suite

**or**

- Use XP Practices religiously

**or**

- what works for you…
Other Use Case StoryoTypes

- [coding] story
- [backbone] story
  - Review Architectural Decisions with Team
  - Review Simplifying Assumptions
- [beefup] story
  - Verify Business Rule with SME
- [interface] story
  - Review Interface on White Board
  - Informal Usability Test
  - Make Improvements

- Note that this is not a big deal
- There’s no magic, but it keeps us “straight” and is a good thing to processize

Analysis StoryoTypes Catalog

- Analysis StoryoTypes are used to “find” Production Stories
  - [paper analysis]
  - [process requests]
  - [work with stakeholders]
  - [team walk-thru]
  - [exploratory testing]
  - [usability testing]
Analysis StoryoTypes

- [analysis] story
  - document new stories in backlog
  - work with PO to prioritize stories
  - timebox the work
  - verbal report to Team

- [paper analysis] story
  - Get Documentation and other Sources
  - validate new stories with SMEs

- [exploratory testing] story
  - Verify focus of testing

- [process requests] story
  - validate new stories with requestors

- [work with stakeholders] story
  - schedule a facilitated meeting
  - prepare for innovation game
  - summarize the results

Business Support

- Used to provide support to the Business. They provide Business Value, but aren’t developing product
  - [sales meeting]
  - [trade show]
  - [train users]
  - [support help desk]
  - etc
Chore StoryoTypes

- Used to set up or improve the Team’s environments, infrastructure, etc
  - [new tool]
  - [new hardware]
- These storyotypes are much more “ad hoc” than the previous ones

Topics to Cover

- Agreement on “Done”
- Examples of Agreements
- Organizational Need to “Standardize”
- StoryoTypes
  - Summary
Overview

- Doneness criteria contribute to a Team's success
- Different kinds of stories have different kinds of doneness criteria
- Doneness criteria are good candidates for standardization in an organization
  - Same codebase
  - Same development environment
  - Somewhat the same process
- StoryoTypes are a method for capturing these standardized Doneness criteria
  - For training
  - For retrospections
  - For helping people move from team to team

Any Questions?
Thank You Very Much!