



Introduction to Software Engineering

Scrum Best Practices

CMPS115 – Summer 2017

Richard Jullig





Acknowledgements

- Material based largely on Construx sources
- Construx: www.construx.com
- Steve McConnell
 - Author of *Code Complete* (2nd ed.), *Rapid Development*, ...



Scrum Best Practices (8 Story Points)

- Learn about Scrum Best Practices
 - As a student of Scrum
 - I need to learn more about Scrum Best Practices
 - So I can more effectively participate in and/or manage software projects



The Definition of Done

- Agile emphasis
 - deliver actual functionality
 - Avoid waste
 - Or: maximize work not done
- Strict definition of progress
 - User stories: Delivered story points
 - Tasks: work remaining in (ideal) working time
- Needed:
Strict definition of Completion: **Definition of Done**
 - For user stories
 - For sprint tasks
- Each team creates its own definitions
- Important: **Apply** Definition of Done **consistently**



Definition of Done (Tasks) – Sample Check List

- *Did you build the thing right?*
(from *engineering perspective*)
- Code checked into repository (Github)
- Code reviewed for standards compliance
- Code reviewed by team member or Walk-through performed
- External/Public API documented
- Unit test definition complete
- Unit tests run without error
- Non-functional tests (e.g. usability, performance) passed
- Regression tests run without error
- Static code analysis performed and passed
- Test coverage measured and achieved
- ...



Acceptance Criteria (1)

- *Did you build the right thing?*
(from *user/customer perspective*)
- Acceptance Criteria
 - Objective criteria to determine whether a user story has been fully and correctly implemented
 - Basis for functional testing of completed user stories
 - Basis for acceptance test before product release
 - Detailed requirements
 - Basis for **TDD** or **BDD**
 - TDD: **T**est **D**riven **D**evelopment
 - BDD: **B**ehavior **D**riven **D**evelopment



Acceptance Criteria (2)

- User stories in Product Backlog
 - User story: an invitation for a conversation
 - **Product backlog**: a **wish list**, shopping list
 - Anyone can add to it
- User stories in Sprint Backlog
 - User story: high-level description of **required functionality**
 - **Sprint backlog**: **team commitment** to deliver software that satisfies the requirements
- **Acceptance criteria**
 - How to test that the delivered software satisfies a requirement



Acceptance Criteria (3)

- Sprint Planning
- For each user story in Sprint backlog
 - Conversation of Product Owner with team
 - Work out details of user story
 - Agree on acceptance criteria
- Record acceptance criteria for each user story
 - E.g. on back of card/post-it for user story

User story: View “miles”

As a frequent flier

I want to see my current “miles”
so that I can plan for free tickets.

Acceptance Criteria: View “miles”

- ☐ View earned miles as of last statement
- ☐ View pending miles (since last statement)
- ☐ View total miles (earned + pending)
- ☐ View miles within 30 days of expiration
- ☐ ...



Definition of Done (User Story) – Sample Check List

- *Did you build the right thing?*
(from *user perspective*)
- All tasks for user story are done
- Tests performed and passed for all acceptance criteria
- User documentation/Help functions complete
- Inspected and accepted by Product Owner
- ...



Team Working Agreements

■ Logistics

- Work room
- Meeting times
- Project repository
 - Location
 - Organization
- Communication channels

■ Development environment

- Platform
 - Virtual machines
- IDE
- Other tools

■ Coding style/standards

■ Work (Process) Patterns

- Definition(s) of Done
- Team collaboration
- Collaboration with experts (SME: subject matter experts)
- Areas of responsibility
- Work hand-off/ integration

■ Product Design Patterns

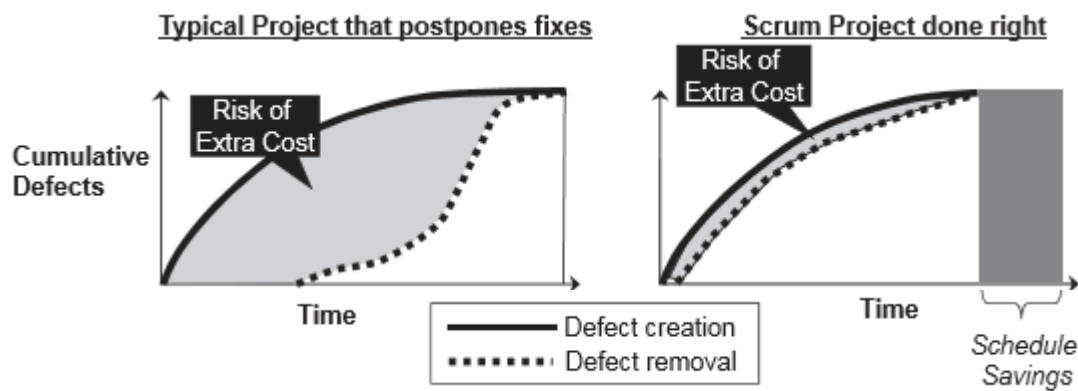
- UI look and feel
- Product architecture
- Common approach to common problems
- Error handling
- ...

What working agreements does your team find useful?



Avoid Waste: Hidden Cost of Defects

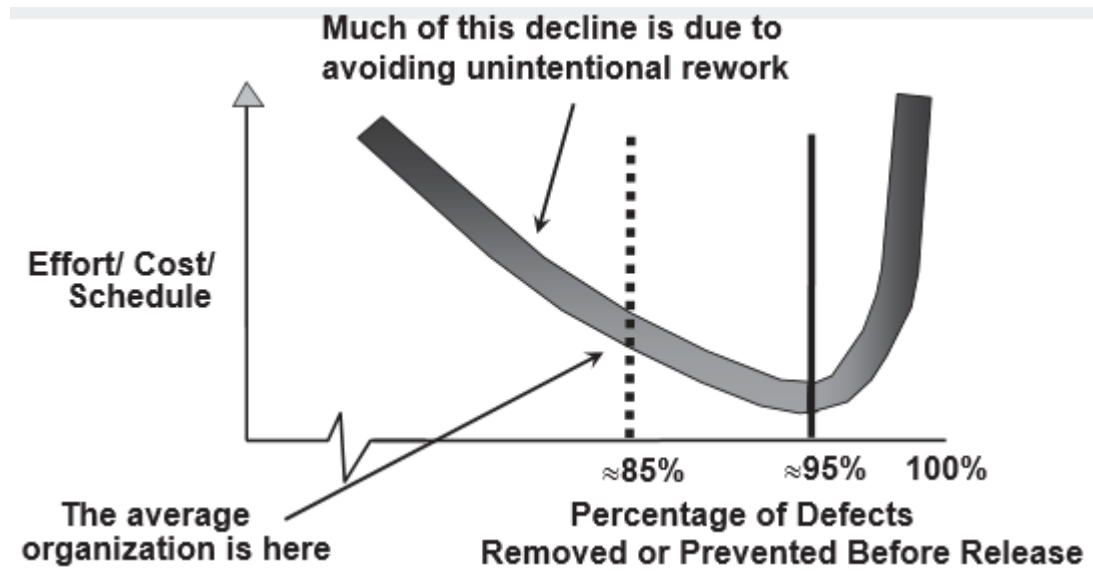
- Leaving problems unresolved increases risk of failure
 - Defect build up: risk of extra cost (effort, time, money)
 - **Technical debt**: steep interest rate; may be hard to pay off
- Minimize time between defect insertion and defect detection and correction
 - Cost increases exponentially with time elapsed





Higher Quality → Faster Development

- Eliminating more defects earlier makes progress faster
 - Reduces effort, schedule, cost
 - Up to a point (about 95%)





Product Increment Strategies

- Layer by layer
or
- Slice by slice ???
- Conventional approach (not necessarily wrong)
 - Layer by layer
 - Example: Making a pizza
 - Make the dough
 - Spread on tomato sauce
 - Add vegetables
 - Add meats
 - Bake
 - Sprinkle on extra cheese and/or hot pepper flakes
- When will customer know whether she likes it?



Product Increments: Vertical Slices

- Start with
 - Releasable system
 - Minimal functionality
 - Establishes feasibility
 - Gives a first taste
- Add increments
 - Releasable system
 - Increased functionality
 - All layers expand as needed
- Vertical slice (“Sashimi”) increments
 - practice essential to successful Scrum projects



More Tips for Effective Scrum

- ✓ Have a defined team room
- ✓ Have Stand-up meeting at the same time and place every time, preferably in the team room
 - ❑ Makes regular updating of Scrum Board and Burn chart more likely
- ✓ Enforce working agreements
 - ❑ Meeting attendance, task status updating, ...
- ✓ Post the Three Questions in **BIG LETTERS!**
 - ❑ Scrum Master should listen for impediments and issues needing follow-up (not ask the three questions)
- ✓ If you use Scrum project management software, use a Scrum Board to track progress anyway.



Agile Myths and Scrum Smells and Plagues

- Understanding problems when first using Scrum
 - As a beginning practitioner of Scrum,
 - I need to understand what can go wrong when a team is new to Scrum
 - So I can identify and remove obstacles to successful Scrum adoption



Common Agile Myths (really: Misunderstandings)

- “We’re (super) Agile, so we don’t need (and don’t have) a plan!”
 - Agile planning does not mean no planning
- “We’re Agile, so we can follow any process we want, and nobody can tell us otherwise!”
 - Being Agile does not mean lack of discipline or using an ill-defined process
- “We’re Agile, so we’re just going to get the product out of the door as quickly as possible no matter how!”
 - Being Agile does not mean doing shoddy work or lack of engineering

Scrum is a **simple** process, but doing it right is **not easy**



What are Scrum Smells?

- “Whiffs”, odors that may not seem so bad at first but over time become overwhelming (so you must get rid of them)
- Easy to ignore at first; better get rid of it right away.
- Ignore scrum smells at your own peril!

Later: Code smells. Same story.




Common Scrum Smells

- Zero or > 1 Product Owners
- The Scrum Task Master
 - As opposed to Scrum Master: coach, mentor, facilitator, protector
- “Just !#\$%& do what I say”
- Commitment Phobia
- Self-unmanaged teams
- Burn-ups/downs that don't
- Urgent things crowd out important things



Important/Urgent: Covey's Quadrant

- Steven Covey: author of *The seven habits of highly successful people*

	URGENT	NOT URGENT
IMPORTANT	Quadrant I: Urgent & Important 	Quadrant II: Not Urgent & Important
NOT IMPORTANT	Quadrant III: Urgent & Not Important	Quadrant IV: Not Urgent & Not Important

- Important:
contributes to achievement of
(important) goals

- Urgent:
requires immediate attention

- Q1: Finish homework before deadline
- Q2: Improve programming skills
- Q3: Answering phone
- Q4: Playing video game



The Plague Known as “Scrum-But”

“We use Scrum, but ...

- ... the team doesn’t follow one or more of the required Scrum practices, e.g.
 - No regular stand-up meetings
 - Stand-up meetings don’t stick to three topics or timebox
 - No permanent Scrum Master or Product Owner
 - Sprint length varies
 - “We’ll fix bugs in the stabilization sprint!”
 - No sprint review
 - No sprint retrospective
 - “What’s the use?”
 - ...

What do you do about this?



Engineering Product and Process

- Start with a coherent vision and a realistic plan
 - “Begin with the end in mind”
- Continuous improvement
 - Requires measures for process and product
- Focus on quality to increase productivity
 - “If now is not the time to do it right, when will you find the time to do it again?”

“Engineering is doing for a nickel what any damned fool can do for a dollar.”
- Henry Ford



Avoid Flaccid Scrum

- Flaccid Scrum (Martin Fowler)
 - Scrum project management practices
 - Without technical software engineering practices
- Leads to inferior software quality ...
 - Technical debt build-up ...
- ... and diminishing productivity
 - ... will slow team over time



Technical Practices within Scrum

Many borrowed from **XP** (e**X**treme **P**rogramming)

- Done Criteria
- Peer review / Pair Programming
- Clean code
- TFD (test-first development)/TDD (test-driven development)
- Continuous integration
- Version Control
- Test coverage criteria
- Static Analysis Tools



There Is No “I” in TEAM

- Shared product vision
 - Team members need to share a common product vision
- Consensus
 - Product/Release backlog
 - Sprint goal and backlog
 - Level of effort estimation
 - User stories in story points
 - Sprint tasks in ideal working hours
- Joint responsibility
 - Release plan
 - Sprint plan
 - Sprint report
 - Scrum practices (stand-up, planning meetings, sprint post mortem)



There is nothing but “I”s in TEAM

TEAM

- “It’s not my job”
 - **Only if** that job is already done or being done by somebody else
- Each team member responsible for reaching team goal
 - Proactive in committing to tasks
 - Diligent in completing tasks committed to
 - Willing to ask for and accept help when needed
 - Willing to offer and provide help when needed
- There is no task that needs doing that you are not allowed to do