Scrum Review

Get to know

- Name
- Experience
- CS 191 take aways
- CS 192 expectations
- Plans after college

Overview

- **Part One**
  - **Traditional Software Process**
  - **Introduction to Agility**
    - Agile Manifesto & Principles
    - Agile Project Management
    - Agile Engineering and Testing
    - The Agile Customer/Product Owner
- **Part Two**
  - **A Manager’s Guide to Scrum**
    - Description
    - Framework
    - Benefits & Organizational impact
  - **Moving Forward…**

Part One

- **Traditional Software Process**
  - Process Overview
  - The Waterfall Process
- **Introduction to Agility**
  - Agile Manifesto
  - Characteristics of an Agile Process
  - Agile Project Management
  - Agile Engineering
  - Agile Testing
  - The Agile Customer/Product Owner

Software Development Process

- A Software Development Process is there to help us build and deliver high quality software to satisfy customer/market demands.
- Questions to ask about your current process:
  - Do we really follow it?
  - Does each step provide added value?
  - Does everyone see the benefit of following the process?
  - Does it work?
    - Does it help us deliver high customer value?
    - Does it help us deliver on time?
    - Does it help us stay within budget?
  - If YES to these questions. Why change it?
  - If NO to any one, then read on…

The Waterfall Process 1

- The “Waterfall” is the product development process currently used by the majority of Software Development organizations since the early 1970’s
- Based on a Tayloresque[1] factory model: rule by compliance; strictly authoritarian and hierarchical.
- A phased, defined, approach
  - Relies on documentation to convey essential information
  - Hand-offs between different functional groups are the project “milestones”
  - Requires clear “command & control” principles to work
  - Resists change

1. “The Principles of Scientific Management” by Frederick W. Taylor (1911)

The Waterfall Process 2

- The original Waterfall paper was written by Winston Royce in 1970
- Royce recommended the use of a “waterfall” model as an *ideal* way of developing software
- But Royce actually discredited the concept himself … *in the same paper*
    - *“The problem was that he only discredited it in words. He included a diagram of the waterfall process in the paper and many people only looked at the pictures, liked the way it ‘worked’ and adopted it. He didn't realize that he needed to put a big red X over the picture to visually communicate that it was a bad idea.”*

      -- Bret Pettichord (post to agile-testing group, 2005
- FORD, Lean manufacturing – Toyota and Honda

The Agile Manifesto (2001)

- Not an “ideal” way - this comes from real experience

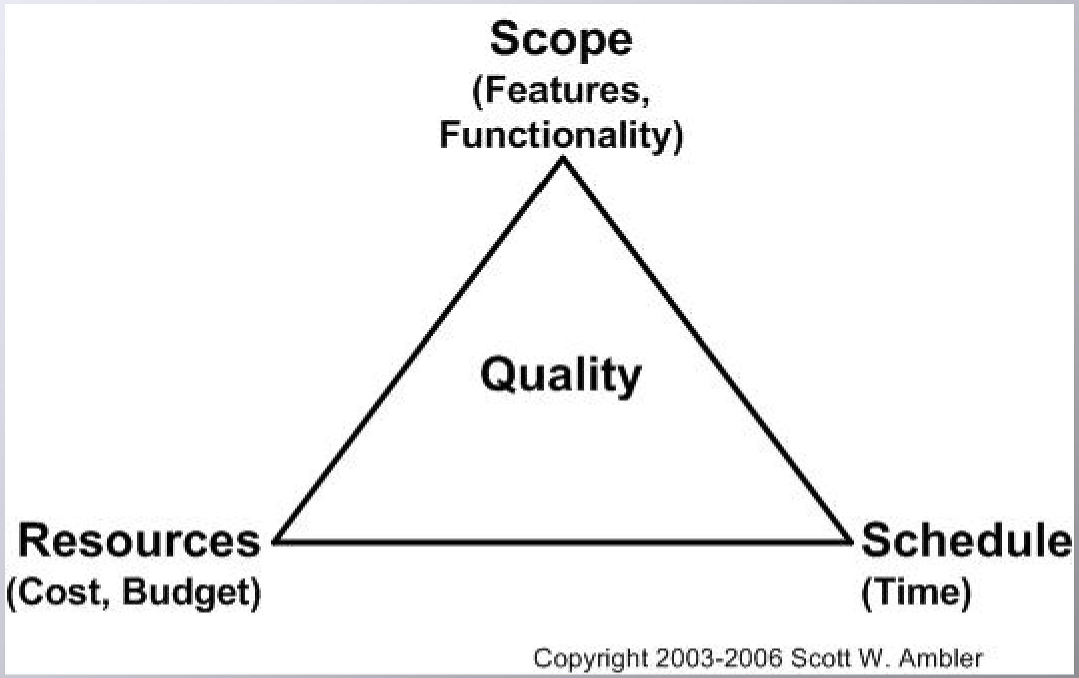
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- ***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.

http://agilemanifesto.org

The Iron Triangle



Characteristics of an Agile Process

- Empirical (relies on observation and experience)
- Lightweight
- Adaptive
- Fast – but never hurried
- Exposes wastefulness
- Customer-centric
  - Example: making a game
- Pushes decision making to lower levels
- Fosters trust, honesty and courage
- Encourages self-organization

Agile Project Management

- Can wrap around most existing practices
  - But most effective when the practices are also Agile
- Supports Iterative and incremental development
- Uses Inspect/Adapt principles
  - For project planning (daily/iteration/release)
  - To ensure highest customer value
- Tracks time remaining only
  - Does not track
    - People. Accuracy of estimates. Task dependencies
- Example methodologies: Scrum, Crystal

Agile Engineering

- Essential Practices
  - Regular refactoring (many times daily)
    - This produces well-componentized designs, clear APIs and clean code without duplications
  - Frequent check ins (many times daily)
  - Unit Testing
    - Leading to Test Driven Development (TDD)
  - Continuous Build and Integration
    - Running automated tests on each build
  - Just-in-time code reviews (e.g. pair programming)
- Example methodologies: XP, Agile Modeling

Agile Testing

- Early involvement
  - An Agile project begins when testers convert high-level requirements into testable specifications.
- Work as part of the development team
  - The testers work with the developers to pick unit test and acceptance test frameworks, and to test the software in parallel with development. This requires a shift in thinking.
- Automate everything
  - (wherever possible)
- Test early, test often
  - Never leave the testing until the end

The Agile Customer

- “Customer’ is a role, not a person
  - Also known as Product Manager, Product Owner
  - Proxy for the entire customer group
- Responsible for the Release Plan
- Responsible for managing the Product Backlog
- Determines business value & priority on a regular basis
- Provides information to development team for estimation purposes
- Works with testers to produce clear, testable user stories for each iteration
- Inspects software regularly (e.g. runs acceptance tests) and provides feedback to the development team

Continuous Improvement Culture

- Your job is
  - Doing your job
  - IMPROVING YOUR JOB

What is Scrum?

- Scrum is an iterative, incremental process for developing any product or managing any work. Scrum accepts that the best designs and the clearest requirements can rarely be fully defined ahead of time and need to be allowed to emerge.
- Based on the "lean manufacturing" processes used by the Japanese auto industry since the early 1980's, Scrum has been shown to drastically cut waste in all areas of production, to increase productivity and to improve the quality of its deliverables.
- Scrum is *“The Art of the Possible”* – Ken Schwaber

Characteristics of Scrum

- Scrum is an Agile Project Management “wrapper”
  - No specific engineering practices are prescribed
- Iterative and incremental
  - The product progresses in a series of short “sprints”
  - Working software is delivered at the end of each sprint
- No PRD – no requirements “sign-off”
  - Requirements are captured as items in an ever-evolving ‘backlog’
- Scrum teams are self-organizing and cross-functional
  - Team improves by holding regular retrospectives
- **<u>Scrum makes you see your problems</u>**
- **<u>TRANSPARENCY</u>**

- **<u>FILIPINO SCRUM?!</u>**

The Game

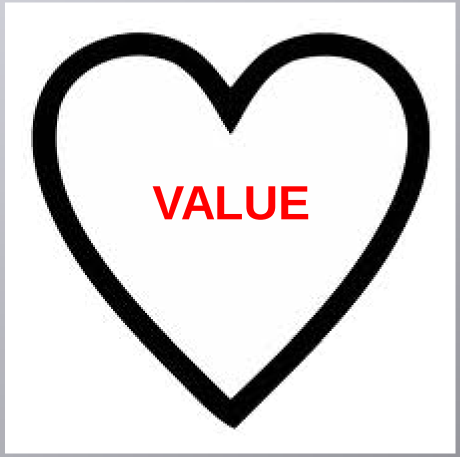
Requirement Side

Development Side

Collaboration

Requirement Side

Development Side



Scrum Roles

- Product Owner
  - Responsible for managing and prioritizing the Product Backlog, and for accepting the software at the end of each iteration
- Scrum Master
  - Responsible for shepherding the team, removing impediments, keeping the process moving, and socializing scrum to the greater organization
- The Team
  - Responsible for estimating size of backlog items, committing to increments of deliverable software – and delivering it. Tracks own progress (with Scrum Master). Is self-organizing – but *accountable* to the product owner for delivering as promised.

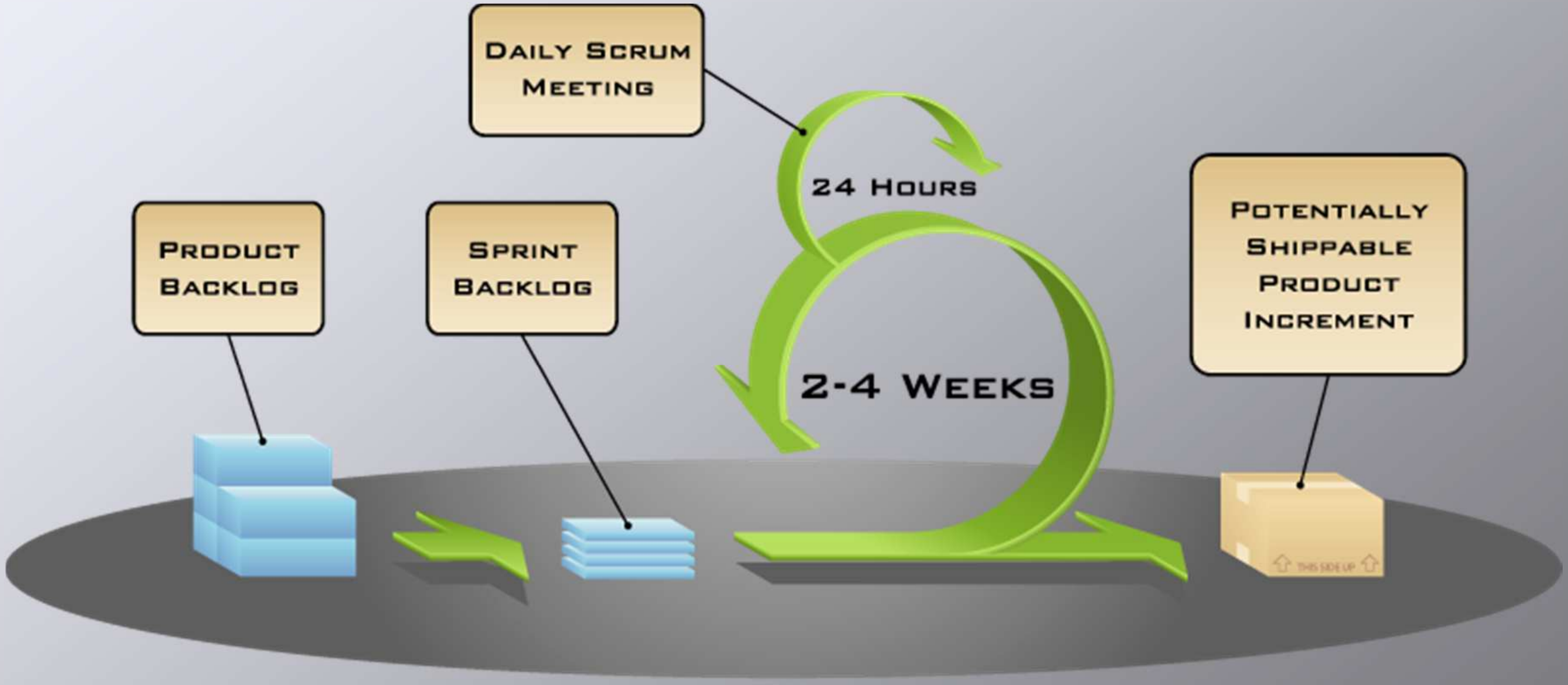
Scrum Meetings

- Estimation Meeting
  - Team meets with product Owner to discuss Backlog Items and assign a relative size value to each.
- Planning Meeting
  - Occurs at the start of each sprint (iteration). Two parts.
    - 1. Product manager and team meet and agree the next product increment.
    - 2. Team then determines the tasks for each backlog item.
- Daily Scrum Meetings
  - Maximum 15 minutes. Team meets to update the task chart and report on progress and impediments.
- Review
  - Team meet with Product Owner at the end of the sprint to demonstrate the working software from the sprint.
- Retrospective
  - Team meets with Scrum Master to inspect and adapt on their process.

The Scrum Framework

How did you practice scrum in CS 191?
No SCRUM-BUT!
Let’s trace scrum…
Artifacts and ceremonies in scrum

The Scrum Framework



Benefits of Scrum

- Frequent delivery of working software
- Released products more accurately meet customer and market needs
- Increased productivity
  - Typically 2-3x, - greater in some cases
  - Development teams produce better quality software, more frequently, working less hours
- Team members report increased sense of empowerment and job satisfaction
- Greater clarity, better visibility, clear accountability

Organizational Impact of Scrum

- Problems become visible – very quickly
  - Ultimately a good thing, but forces immediate resolution to occur
- Organizational hierarchies tend to get leveled
  - Can cause uneasiness among employees
- Management roles change to mentoring/support/ service roles
  - **(Project Management – thing of the past) – relinquish control – servant leadership – POWER IS ADDICTING**
  - Job descriptions may need to be redefined
  - Retraining may need to occur
- All personnel have to rethink the way they work. This is challenging

Moving Forward…

- Scrum is simple – but difficult
  - An understanding and appreciation of Agile values and principles is essential
  - Scrum alone will not create better products.
    - **Engineering, design and testing practices need to become more Agile**
    - Interactions and communication need to become clearer, more personal and more transparent
    - **Honesty, trust and a sense of commitment need to be developed**
    - **When the going gets tough it is easy to slip back into the old way of doing things. Courage is essential.**

Moving Forward

- Good teams have conflicts
- How to continuously improve
- THINK! THINK! THINK!
- Frameworks
- Automation
- Continuous Integration