Agile

- Agile Principles, Methods, Practices
- Agile dev in SYSC4806

[Credit: Many slides taken from University of Washington course CS403, by Marty Stepp]

“Agile”

Principles

Methods

Practices

Tools

Agile principles

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

(agile manifesto, 2001)

Agile Methods

- Scrum
- Extreme Programming
- Kanban
- ...

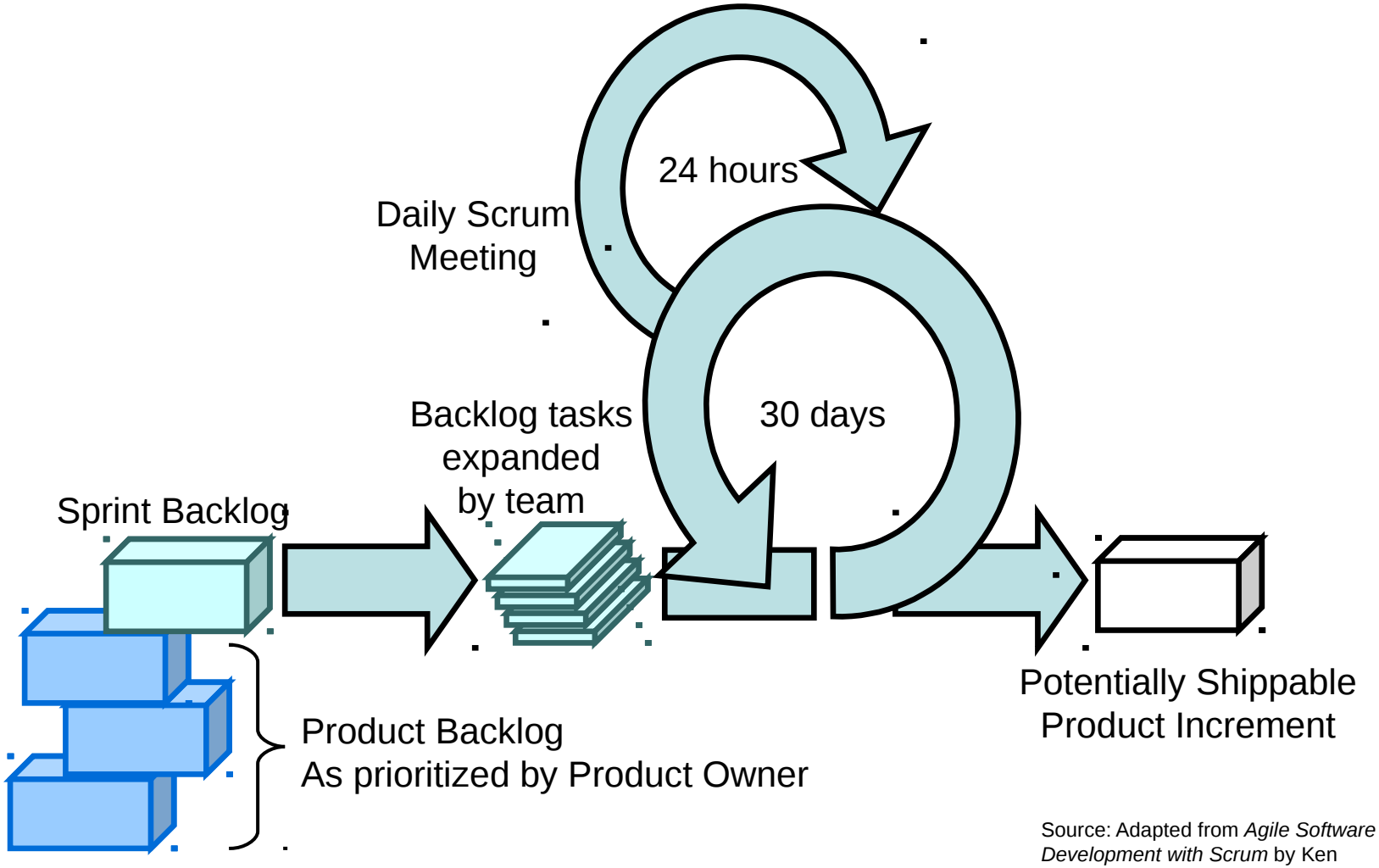
Agile Practices

- Scrums (short stand-up meetings)
- Test-driven development (TDD)
- Code reviews
- Pair programming
- Self-organizing teams

Scrum

- Methodology developed in the 1990s
  (Jeff Sutherland, Ken Schwaber, Mike Beedle)
- Centered around short “sprints”, with daily scrums
- Artifacts:
  - Product backlog
  - Sprint backlog
  - Burndown charts
- 3 roles:
  - Product Owner
  - Scrum Master
  - Team

Scrum



Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Scrum Roles

- Product Owner
  - Possibly a Product Manager or Project Sponsor
  - Decides features, release date, prioritization, $$$
- Scrum Master
  - Typically a Project Manager or Team Leader
  - Responsible for enacting Scrum values and practices
  - Remove impediments / politics, keeps everyone productive
- Project Team
  - 5-10 members; Teams are self-organizing
  - Cross-functional: QA, Programmers, UI Designers, etc.
  - Membership should change only between sprints

Sprint Planning

- Prioritize
  - Analyze/evaluate product backlog
  - Select sprint goal
- Plan
  - Decide how to achieve this goal
  - Create sprint backlog (tasks)
  - Estimate cost/time

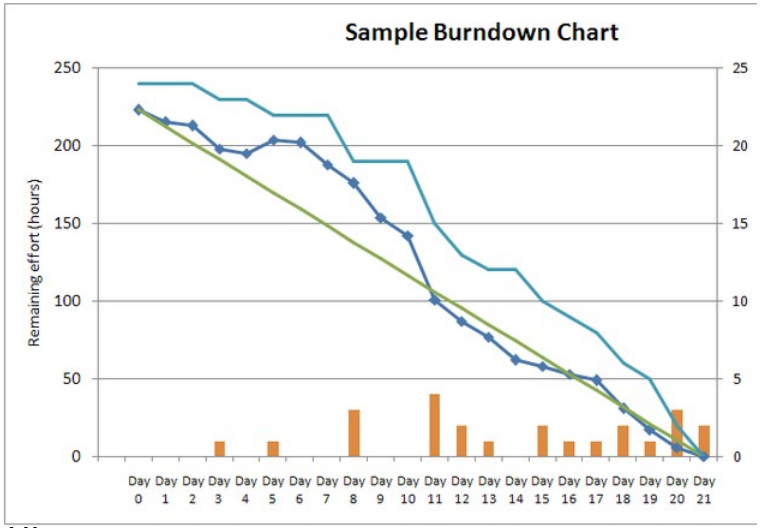
Daily Scrum

- Typical Format:
  - ~15 minutes
  - Standing up
  - Avoid problem solving / decision making

- Each person says in turn:
  - What did I do yesterday?
  - What do I plan on doing today?
  - What's holding me back?

Artifacts

- Product backlog
  - Requirements
  - Typically: user “stories” with points associated (importance/complexity)
  - Prioritized by product owner
- Sprint backlog
  - Tasks to implement story
  - Self-assigned by team members
- Burndown chart
  - Visualization of backlog reducing over time / points gained



Sample Product Backlog

Backlog item	Estimate
Allow a guest to make a reservation	3 (story points)
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As a hotel employee, I can run RevPAR reports (revenue-per-available-room)	8
Improve exception handling	8
...	30
...	50

Sample Sprint Backlog

Sprint 1

01/11/2004					Sprint Day	1 Mo	2 Tu	3 We	4 Th	5 Fr	6 Sa	7 Su
	19 days work in this sprint				Hours remaining	152	152	152	152	152	152	152
	Backlog Item	Backlog Item	Owner	Estimate								
1	Minor	Remove user kludge in .dpr file	BC	8		8	8	8	8	8	8	8
2	Minor	Remove cMap/cMenu/cMenuSize from disciplines.pas	BC	8		8	8	8	8	8	8	8
3	Minor	Create "Legacy" discipline node with old civils and E&I content	BC	8		8	8	8	8	8	8	8
4	Major	Augment each tbl operation to support network operation	BC	80		80	80	80	80	80	80	80
5	Major	Extend Engineering Design estimate items to include summaries	BC	16		16	16	16	16	16	16	16
6	Super	Supervision/Guidance	CAM	32		32	32	32	32	32	32	32

Agile practices for SYSC 4806

- 2-week mini-sprints
  - Planned during labs 1, 3, and 5
- Weekly “scrum” messages
  - Each week, every team member must submit a “scrum” report
  - A few lines with the 3 questions:
    - What did I do this week?
    - What am I doing next week?
    - What's holding me back?
  - Don't miss these!

- Code reviews, Continuous integration and delivery

Code reviews

Why code reviews?

- > 1 person has seen every piece of code
- Prospect of someone reviewing your code raises quality threshold.
- Forces code authors to articulate their decisions
- Even novice programmers can commit to master
- Reduces redundancy, enhances overall understanding

When, how?

- Before every merge to master
- Open pull request on github, @mention other developers, TA
- comment code lines directly in github

Code reviews: actual studies

Average defect detection rates

- Unit testing: 25%
- Function testing: 35%
- Integration testing:45%
- **Design and code inspections: 55% and 60%.**

11 programs developed by the same group of people

- First 5 without reviews: average 4.5 errors per 100 lines of code
- Next 6 with reviews: average 0.82 errors per 100 lines of code
- Errors reduced by **> 80 percent.**
- IBM's Orbit project: 500,000 lines, 11 levels of inspections. Delivered early and 1 % of the errors that would normally be expected.
- After AT&T introduced reviews, study with > 200 people reported +14% productivity, -90% defects.

(From Steve McConnell's *Code Complete*)

Code reviews: What we're looking for

- Verification, not validation
- Coding Style Standards
  - Indentation, Case conventions...
- Comments
  - Enough to make this understandable
- Logic
  - Understand what the code does
  - If you can't understand it, break down PR into smaller Prs
- Coding decisions
  - Nice code, elegant solutions, ...