Agile overview

Extreme Programming (XP)

- Quick feedback
  - Test Driven Development (TDD)
  - Customer participation
  - Paired programming
- Continuous Integration (CI)
  - Refactoring
  - Frequent small releases
- Understanding and Simplicity
  - Simplicity of design
  - Standards for code writing
  - Collective responsibility for code
- Social security
  - 40-hour work week

Scrum

- Small multifunctional teams
- A well-decomposed list of tasks with priorities (backlog of the product and sprint backlog)
- Sprint execution control
- Iterations with the final demonstration to the client
- Process Retrospective

Manifesto for Agile Software Development

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
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

http://agilemanifesto.org/

12 Agile Manifesto Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity--the art of maximizing the amount of work not done--is essential.

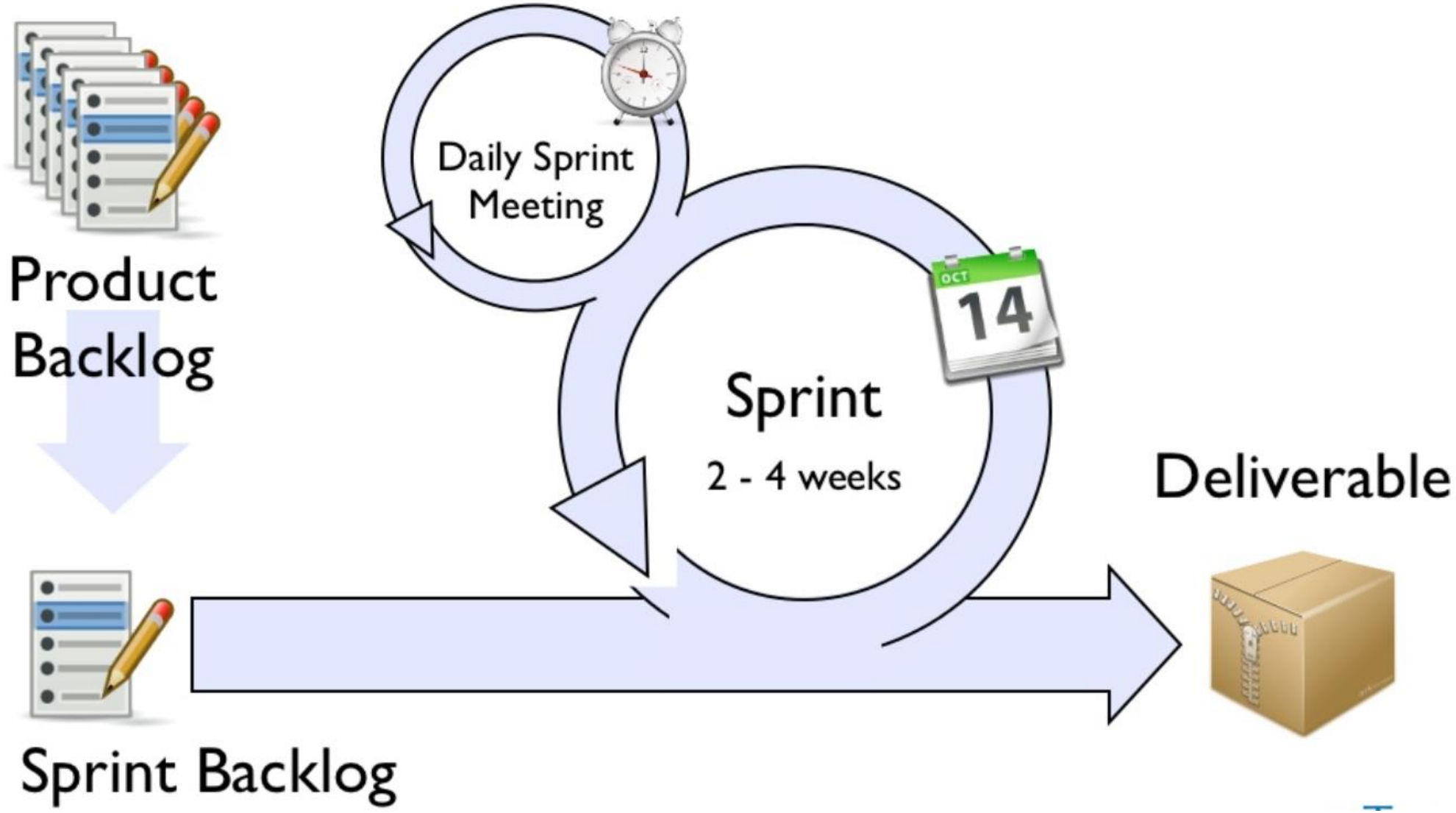
11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile principles

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

Scrum Sprint Cycle



1 Put the customer at the center

Beck: *You will get better results with real cutomers. They are who you are trying to please*

2 Accept change

Poppendieck: *While in theory Object Oriented development produces code that easy to change, in practice OO systems can be as difficult to change as any other.*

Extendibility!

3 Let the team self-organize

Traditional view: managers tell workers to do their job

Agile view: managers listen to developers, explain possible actions, provide suggestions for improvements

Leader is there to:

- Encourage progress
- Help catch errors
- Remove impediments
- Provide support and help in diffucult situations
- Make team spirit

4 Maintain sustainable pace

People perform best if they are not overstressed

Developoers should not work more than 40 hour weeks

To achieve:

- Frequent code-merge
- Always maintain executable, test-covered, high-quality code
- Constant refactoring, helping keep fresh and alert minds
- Collaborative style
- Constant testing

5 Develop minimal software

Minimalism:

- Minimal functionality
- Product only
- Only code and tests

- Jeffries: *this principle reminds us always to work on the story we have, not womething we think we're going to need. Even if we know we're going to need it.*

5 Develop minimal software

Poppendieck: *Our software systems contain far more features than are ever going to be used. Extra features increase the complexity of the code, driving up costs nonlinearly. If even half of our code is unnecessary — a conservative estimate — the cost is not just double; it's perhaps ten times more expensive than it needs to be.*

Cockburn: *You get no credit for any item that does not result in running, tested code. Okay, you also get credit for final deliverables such as training materials and delivery documentation.*

6 Develop iteratively

- *User interface*
- *Business logic*
- *Networking*
- *Database*

The Closed-Window Rule: During an iteration, no one may add functionality

(or: the sprint is cancelled)

7 Tests

Do not move on until all tests pass

Excellent principle, to be reconciled with practical constraints

Need to classify complexity

8 Express requirements through scenarios

“Stories are more than just text written on an index card but for our purposes here, just think of user story as a bit of text saying something like

Paginate the monthly sales report

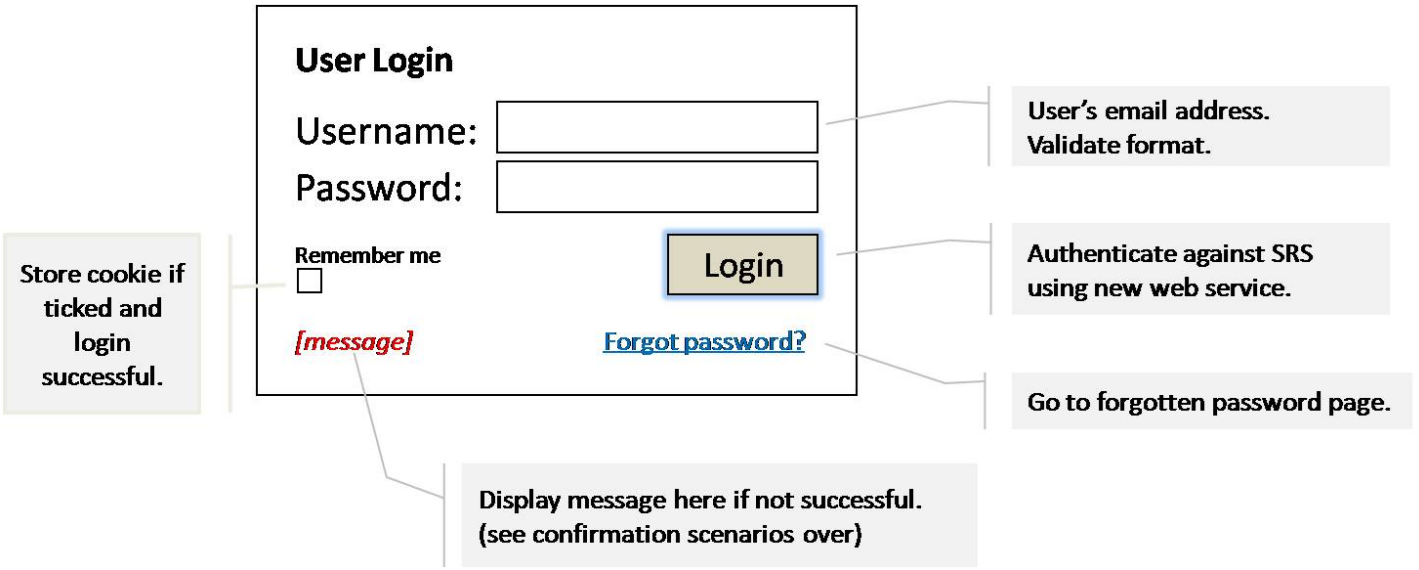
Change tax calculations on invoices.

Many teams have learned the benefits of writing user stories in the form of “As a … I … so that …”

8 User story example

#0001	**USER LOGIN**	Fibonacci Size # **3**
As a **[registered user]**, I want to **[log in]**, so I can **[access subscriber content]**.		

For new features, annotated wireframe. For bugs, steps to reproduce with screenshot. For non-functional stories, explain scope/standards.



Further information is attached to this story on VSTS Product Backlog.

Scrum roles

- (Self-organizing) team
- Product owner
- Scrum Master

Team

The team:

- Is cross-functional
- Has 7+/- 2 members
- Selects iteration goal and work results
- Organizes itself and its work
- Can do everything within guidelines to reach goal
- Demos work results to product owner

Product owner

The product owner:

- Defines product features
- Decides on release date
- Decides on release content
- Responsible for product profitability (ROI)
- Prioritizes features according to market value
- Can change features and priority over 30 days
- Accepts or rejects work results

Scrum master

The Scrum Master:

- Ensures that the team is functional and productive
- Enables cooperation across all roles & functions
- Shields team from external interferences
- Enforces process: daily meeting, planning & review meetings
- Removes impediments
- Normally, does not develop

«Other» people

- Product Marketing
- Marketing Communication
- Directors
- Executives

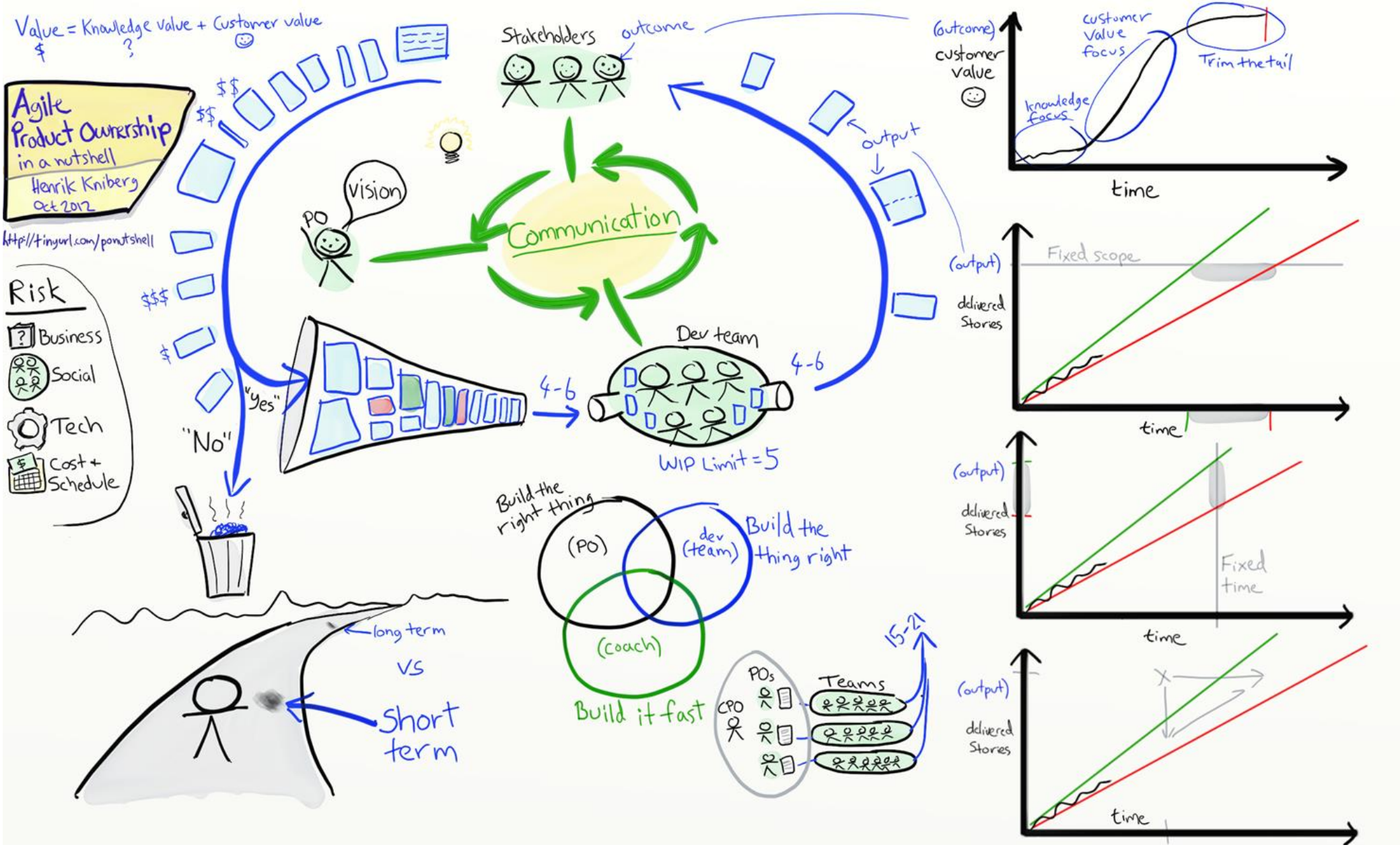
- Can attend dailies but cannot speak. Can participate in planning. Interact with team through Project Manager/ScrumMaster

Product Manager vs Product Owner

- Product Owner owns the message to the Agile team and the Sprint/Iteration Backlog
- Product Manager owns the Roadmap / Strategy / Vision / Product Backlog
- Often they are the same person

Product Manager vs. Product Owner (cont'd)

Product Manager	Product Owner
Market Sensing / Problem Statements	Tracks internal deliveries
Release Objectives	Iteration Objectives
Strategic Direction	Day-to-day tactical direction
Market Use Cases/Scenarios	System Use Cases/Scenarios
Understands overall solution	Understands architecture and design
Roadmaps	User acceptance tests
Manage Release Portfolios and Backlogs	Manage Iteration and Cross Project priorities
Provides Vision	Provides Implementation
Messaging & Positioning	Unblocks teams throughout iteration
Directs Product Owner	Takes direction from Product Manager
Delivers the Release	Delivers the Iteration



http://blog.crisp.se/2012/10/25/henrikkniberg/agile-product-ownership-in-a-nutshell

Daily meetings

Held every morning

Goal: set the day's work, in the broader context of the project

Time-limited, usually 15 minutes ("stand-up meeting")

Involves all team members, with special role for "committed" (over just "involved")

Focus:

- Defining commitments
- Uncovering impediments
- Resolution will take place outside of meeting

Planning meeting

At beginning of every sprint

Goal: define work for sprint

Outcome: Sprint Backlog, with time estimate for every task

8-hour time limit

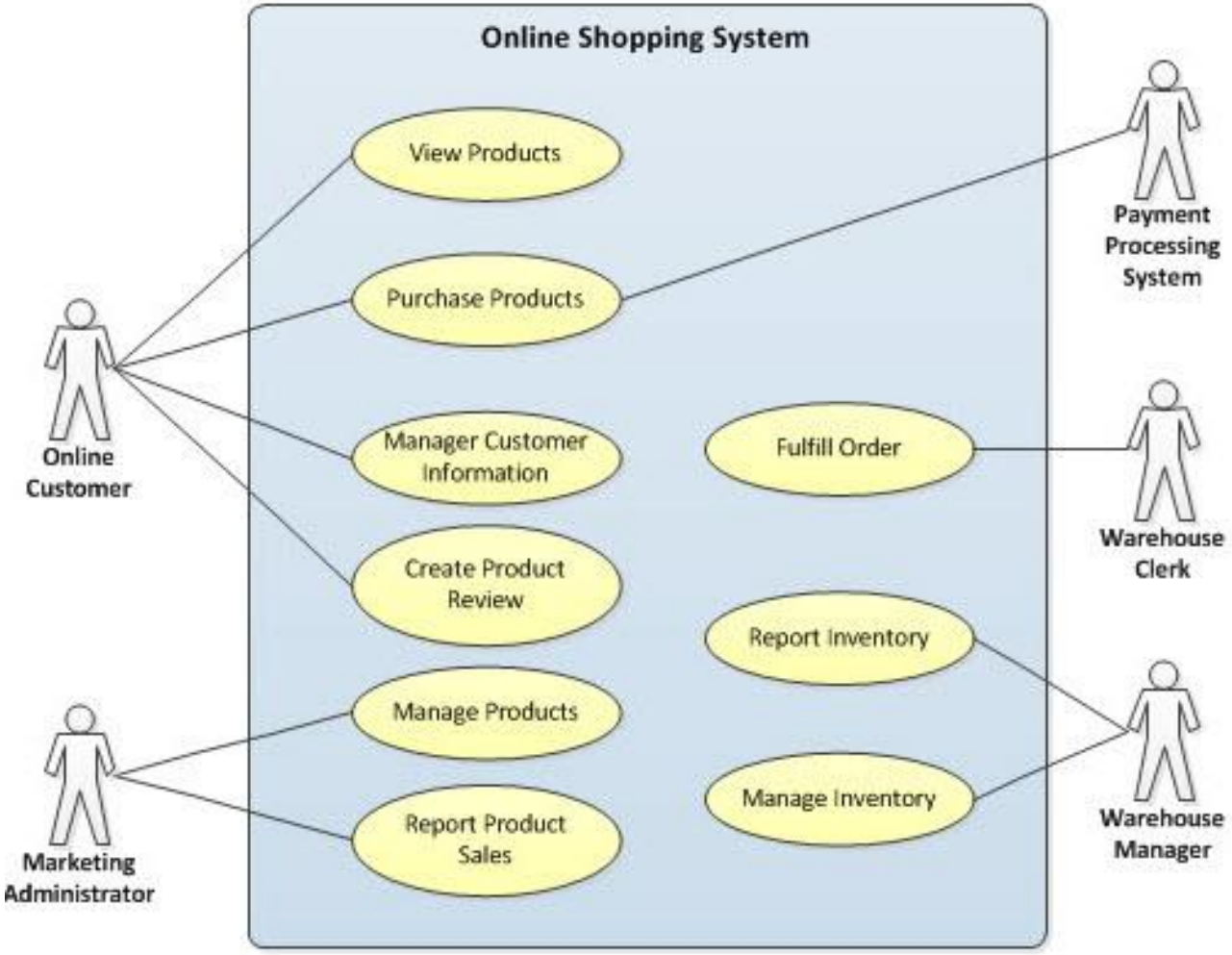
- 1st half, product owner + team: prioritize product backlog
- 2nd half, team only: plan for Sprint, producing sprint backlogHeld every morning

Agile artifacts: use case

Describes how to achieve single business goal or task through the interactions between external actors and system

One of the UML diagram types

Use case example



Use case vs User story

User story:

- Very simple
- Written by customer
- Incomplete, possibly inaccurate
- Does not handle exceptional cases
- Starting point for additional discussions with customer

- Use case:
- More complex
- Written by developer in cooperation with customer
- Attempts to be complete, accurate
- Should handle all possible cases
- Intended to answer any developer questions about customer requirements without further interaction with customer

Product backlog

- Maintained throughout project
- Property of product owner
- Open and editable by anyone
- Contains backlog items: broad descriptions of all potential features, prioritized by business value
- Includes estimates of business value
- Includes estimates of development effort, set by team

- Visualized in “task board” (see next)

Backlog

Backlog is the master list of all functionality

- Features
- Epics
- Stories
- Requirements
- Bugs

- Item Attributes:
- Desctription
- Cost estimate (points or size)
- Priority

Product Backlogs vs Sprint Backlogs

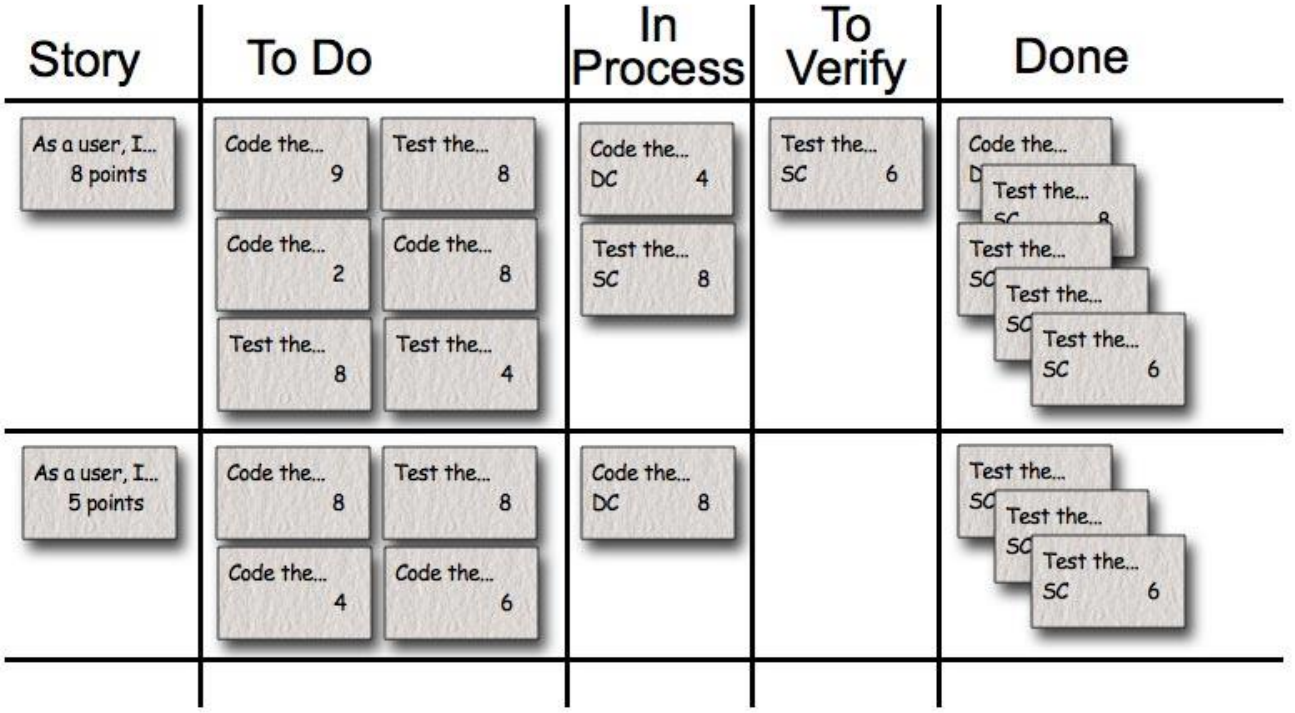
Product Backlog is the master of all functionality for the product

- Features
- Epic
- Bugs

- Sprint Backlog is the list of functionality that the team is commiting that they will complete in the current iteration
- Stories
- Requirements
- Bugs

Task board, story board

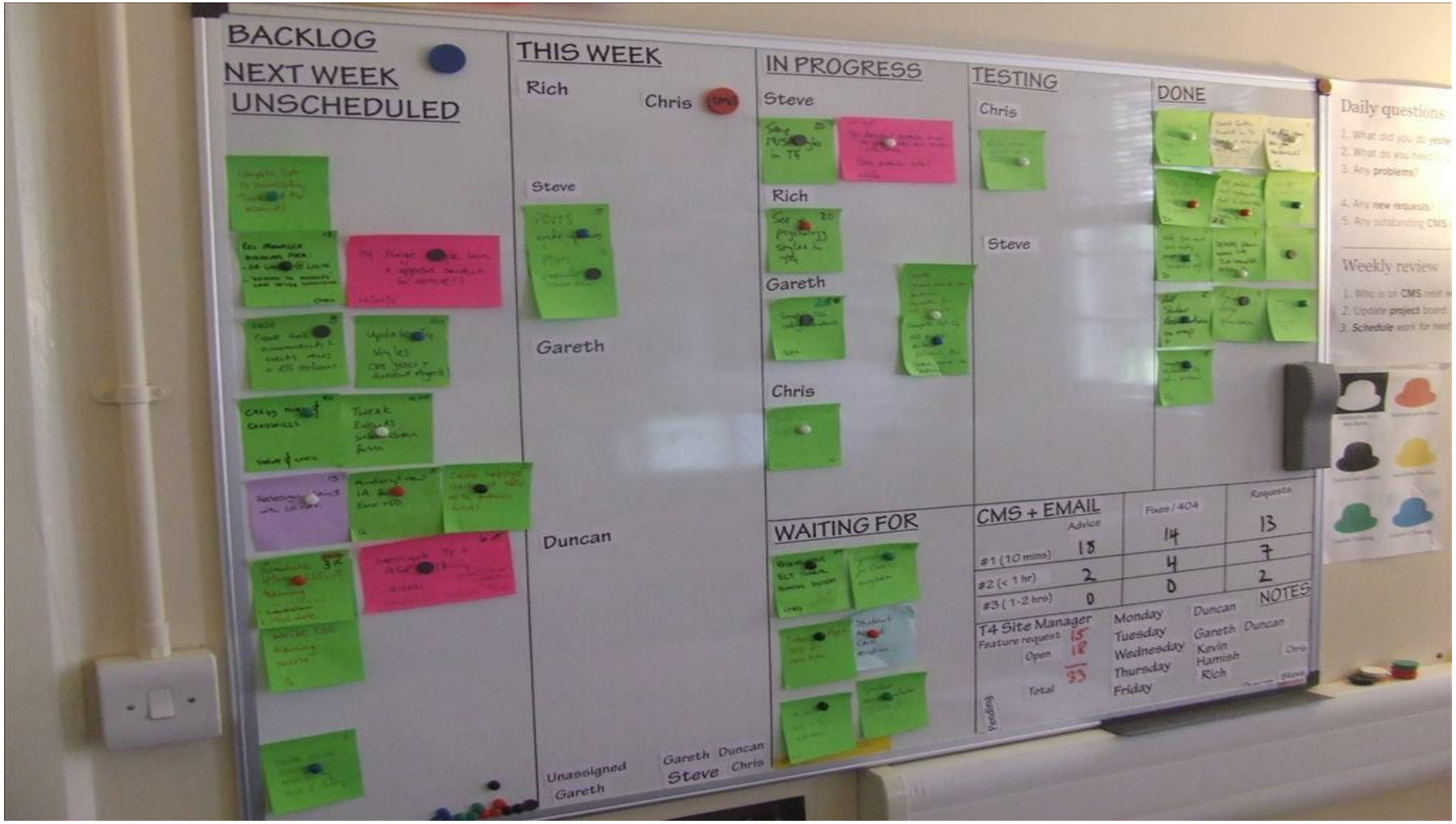
Used to see and change the state of the tasks of the current sprint: “to do”, “in progress”, “done”.



Benefits:

- Transparency
- Collaboration
- Prioritization
- Focus
- Self-organization
- Empiricism
- “Humility”
- Morale

Task board, story board



Task board in Jira

Agile ugly

- Rejection of upfront tasks
  - Particularly: no upfront requirements
  - Dismissal of a priori architecture work
- User stories as a replacement for abstract requirements
- Tests as a replacement for specifications
- Feature-based development & ignorance of dependencies
- Method keeper (e.g. Scrum Master) as a separate role
- Test-driven development (but not the rest of agile's emphasis on tests)
- Dismissal of traditional manager tasks
- Dismissal of auxiliary products and non-shippable artifacts
- Dismissal of a priori concern for extendibility
- Dismissal of a priori concern for reusability

Agile good

- Acceptance of change
- Iterative development
- Emphasis on working code
- Tests as one of the key resources of the project
- Constant test regression analysis
- Notion of velocity
- No branching
- Product (but not user stories!) burndown chart
- Daily meeting