

Agile Software Development

Credits to: Aaron Ciaghi, Pietro Molini

Some Material from Mike Cohn and Mountain Goat Software

Outline

- A short talk about waste
- Intro to Agile Software Development
- Examples of Agile Development Processes
 - DSDM (overview)
 - eXtreme Programming (more detailed)
 - Scrum (in depth)

A short talk about waste

Lean Manufacturing

- Lean manufacturing is a production practice that considers the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful, and thus a target for elimination.
- Value = what the user is willing to pay for
- Derived mostly from the Toyota Production System (hence the term Toyotism is also prevalent) and identified as "Lean" only in the 1990s
- Economical motivations in adapting production to demand, high cost of land (storage), economies of scale to improve competitiveness

無駄

(muda - unnecessary)

無理

(muri – work that can
be avoided)

斑

(mura – unevenness)

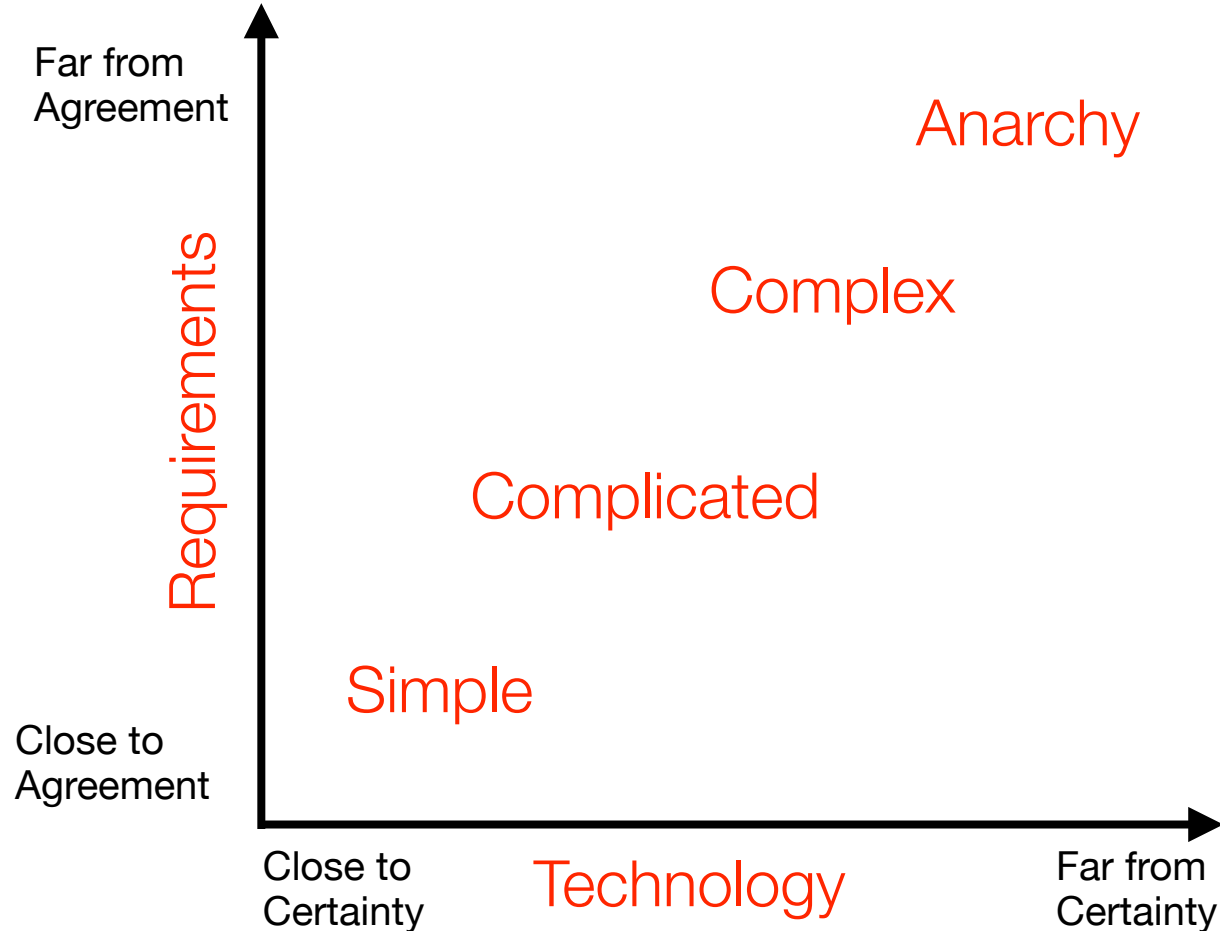
TIMWOOD – types of 無駄

- **Transportation:** moving products that is not actually required to perform the processing
- **Inventory:** all components, work in progress, and finished product not being processed
- **Motion:** people or equipment moving or walking more than is required to perform the processing
- **Waiting:** waiting for the next production step
- **Overproduction:** production ahead of demand
- **Over Processing:** resulting from poor tool or product design creating activity
- **Defects:** the effort involved in inspecting for and fixing defects

Causes of Waste in Software Development

- Accurate estimation, effective planning, management and control, and deterministic delivery of quality software has always been subject of great debate
- Some of the causes include:
 - Immateriality: software is difficult to measure
 - Flexibility: “why don’t we add this nice feature?”
 - Complexity
 - New: maturity of engineering techniques

Software Development Complexity



Source: *Strategic Management and Organizational Dynamics* by Ralph Stacey in *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

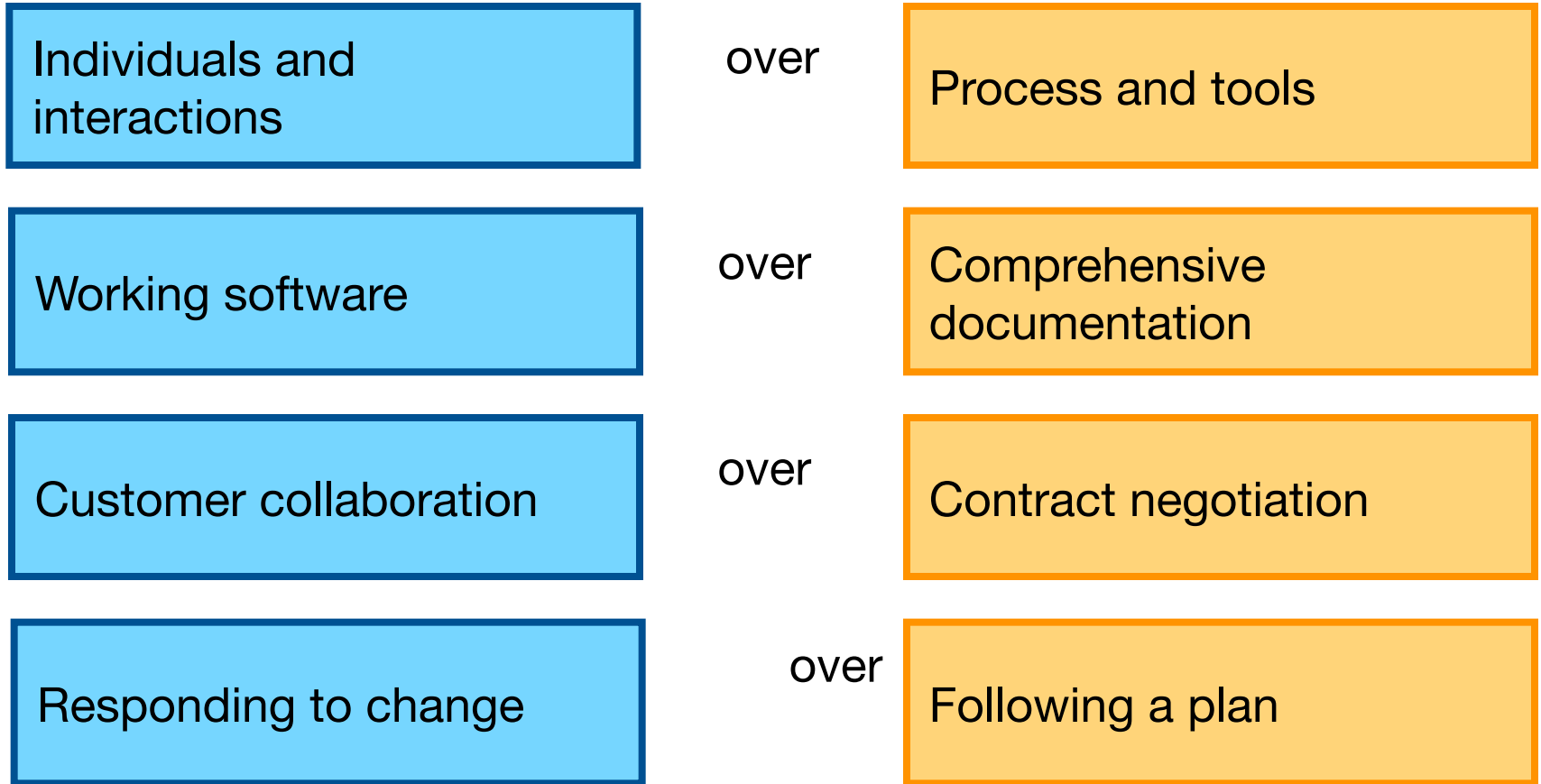
Motivation for Agile (1/2)

- Traditional approaches to control complexity in software development rely on superimposing structure on development
- A natural trend from the Waterfall model that risks to end up in a bureaucratic structure

Motivation for Agile (2/2)

- **Agile Software Development:** A group of SW development methodologies based on similar principles:
 - A management process that encourages frequent inspection and adaptation
 - Team work, self organization, accountability
 - Rapid delivery of high-quality software
 - A business approach that aligns development with customer needs and company goals

The Agile Manifesto



Source: www.agilemanifesto.org

Some Consequences

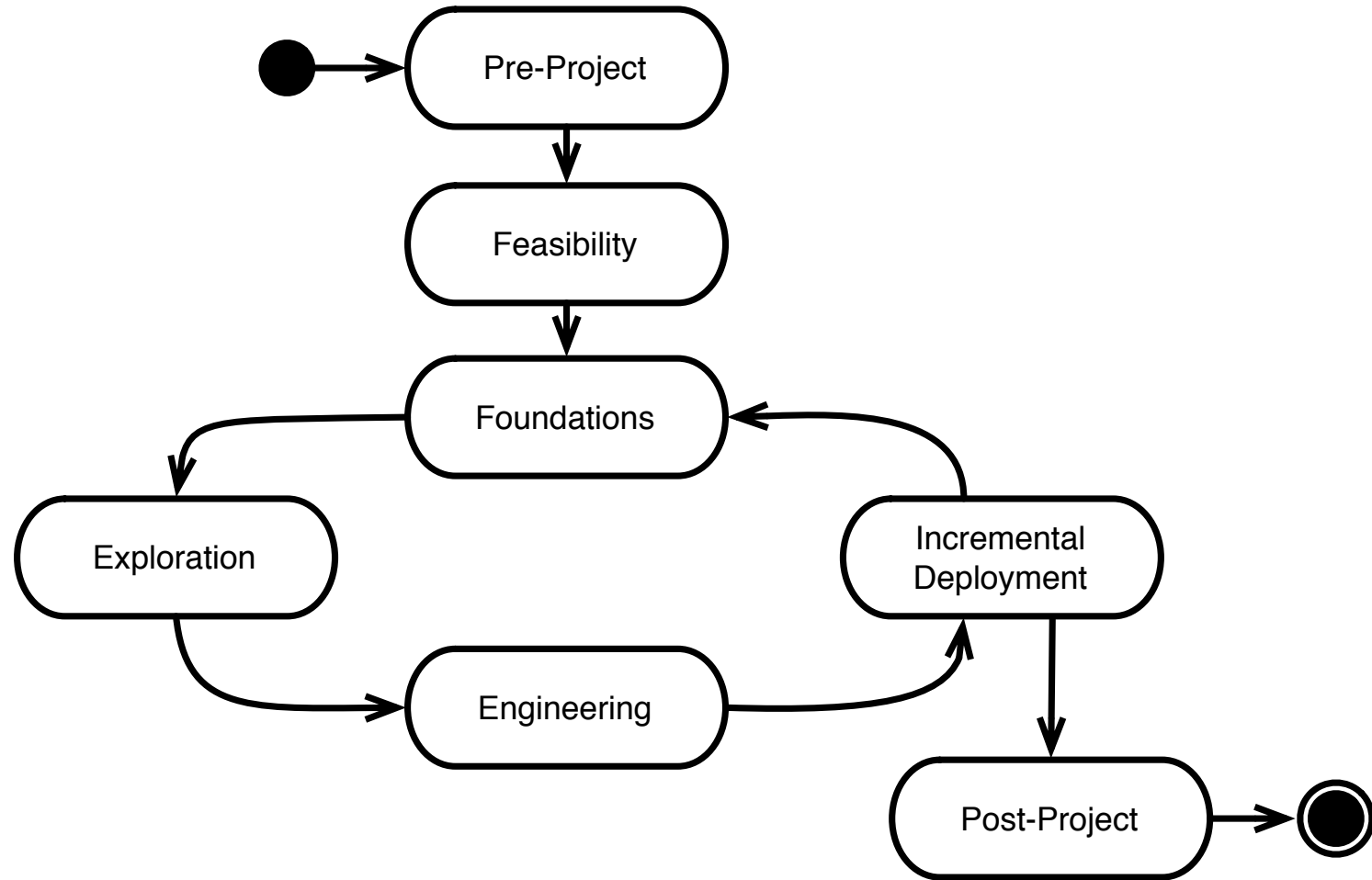
- Customer satisfaction
- Working software delivered frequently
- Working software is the principal measure of progress
- Late changes in requirements = no problem
- Close, daily cooperation
- Face2Face & Co-location
- Projects built around individuals
- Attention to technical excellence and good design
- Simplicity
- Adaptation

DSDM

Dynamic System Development Method

- It starts from the experience of RAD (Rapid Application Development)
- Key considerations:
 - people are the key to project success
 - change is inevitable
 - no software is perfect the first time it is released
- Standardized and supported by a consortium

DSDM



Extreme Programming

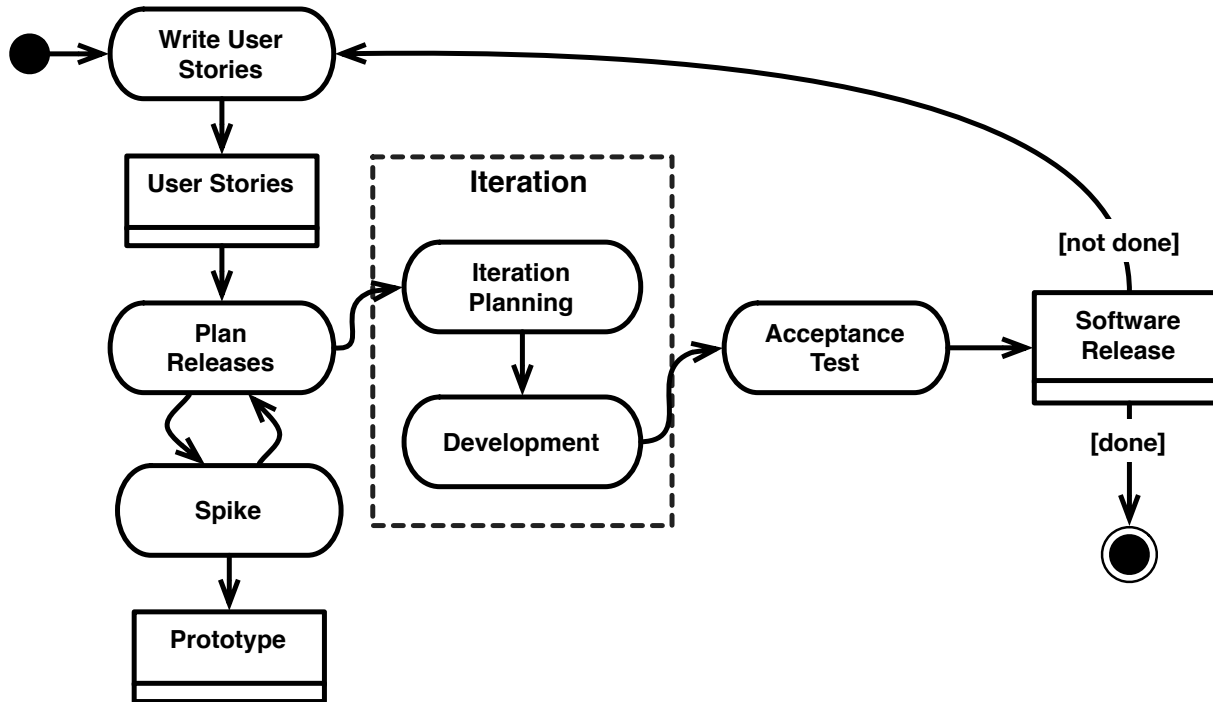
Extreme Programming

- XP takes proven practices to the extreme
 - If testing is good, let everybody test all the time
 - If code reviews are good, review all the time
 - If design is good, refactor all the time
 - If integration testing is good, integrate all the time
 - If simplicity is good, do the simplest thing that could possibly work
 - If short iterations are good, make them really, really short
- XP is based on:
 - values
 - practices
 - process

XP values

- **Communication:** open and honest
- **Feedback:** honest and rapid (close the loop quickly to make feedback effective)
- **Simplicity:** K.I.S.S.
- **Courage:** take the right decisions, even if they are difficult (if it can't be done, say it!)

XP Process



- Planning
- Management
- Design
- Coding
- Testing

XP Planning

- User stories are written
- Release planning creates the release schedule
- Make frequent small releases
- The project is divided into iterations
- Iteration planning starts each iteration

XP Management

- Give the team a dedicated open work space
- Set a sustainable pace
- A stand up meeting starts each day
- The Project Velocity is measured
- Move people around
- Fix XP when it breaks

XP Design

- Simplicity
- Choose a system metaphor
- Use CRC cards for design sessions
- Create spike solutions to reduce risk
- No functionality is added early
- Refactor whenever and wherever possible

XP Coding

- The customer is always available
- Code must be written to agreed standards
- Code the unit test first
- All production code is pair programmed
- Only one pair integrates code at a time
- Integrate often
- Set up a dedicated integration computer
- Use collective ownership

XP Testing

- All code must have unit tests
- All code must pass all unit tests before it can be released
- When a bug is found tests are created
- Acceptance tests are run often and the score is published

Pair Programming

- Two programmers work on the same screen/keyboard (the Driver and the Navigator)
- Some considerations:
 - About 15% less output than 2 solo programmers
 - Continuous code review: better design, fewer defects
 - Confidence to add to or change the system
 - Discipline to always test and refactor
 - Teach each other how the system works (reduced staffing risks)
 - Learn from partner's knowledge and experience (enhances technical skills)

Simple design

- Do the simplest thing that could possibly work
- Passes all the tests
- No duplicate code
- States every intention
- Fewest possible classes and methods

Refactoring

- Design becomes everybody's daily business
- Continuously improve quality of the code
- Unit Tests and Pair Programming give courage
- Consequences:
 - Fast development speed
 - Code becomes easy to change

Why XP works

- Light-weight: discipline without bureaucracy
- Under stress, people do what is easiest
- All XP practices have short-term benefits as well as long-term benefits
- Development as a Conversation
- The code is the documentation
- XP is fun

XP Criticism

- Too many changes = higher costs
- “Scope creep” beyond what is agreed at the beginning of the project
- Requirements continuously updated
- No “Big Design” upfront
- Customer representative = single point of failure
- Generates stress
- Risk of non-technical representative dictating technical choices

Scrum

Scrum Approach

A team approach in which all players move together towards the same goal, setting the rhythm and adapting to change similar to what happens in rugby

Scrum Principles

- **Built-in instability:** broad goals and general strategic directions
- **Self-organizing project teams**
- **Overlapping development phases**
- **Multilearning**, so that the team can learn both from internal and external sources and adapt quickly to changing conditions and environments.
- **Subtle control:** steer the project without interfering too much (e.g., selecting the right people for the job, creating an open environment, tolerating and anticipating mistakes).
- **Organizational transfer of learning:** by ensuring that the know-how acquired in a project is transferred and reused in other projects.

Scrum has been used by:

- Microsoft
- Yahoo
- Google
- Electronic Arts
- IBM
- Lockheed Martin
- Philips
- Turner Broadcasting
- Océ
- Siemens
- Nokia
- Capital One
- Nielsen Media
- First American Real Estate
- BMC Software
- Ipswitch
- John Deere
- Lexis Nexis
- Intuit
- Sabre
- Salesforce.com
- Time Warner
- BBC

source



Scrum has been used for:

- Commercial software
- In-house development
- Contract development
- Fixed-price projects
- Financial applications
- ISO 9001-certified applications
- Embedded systems
- 24x7 systems with 99.999% uptime requirements
- the Joint Strike Fighter
- Video game development
- FDA-approved, life-critical systems
- Satellite-control software
- Websites
- Handheld software
- Mobile phones
- Network switching applications
- ISV applications
- Some of the largest applications in use

source



The Scrum Roles

Scrum Roles in One Slide

- Scrum Master
 - Facilitator, ensures Scrum is applied and the team can operate and work
 - Shields team from external interferences
- Product owner
 - Defines the features of the product, decides priorities, accepts or rejects work
- Team
 - Self organized, ideally full-time, small (5-9)

The Scrum Master

- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences



Product owner

- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results



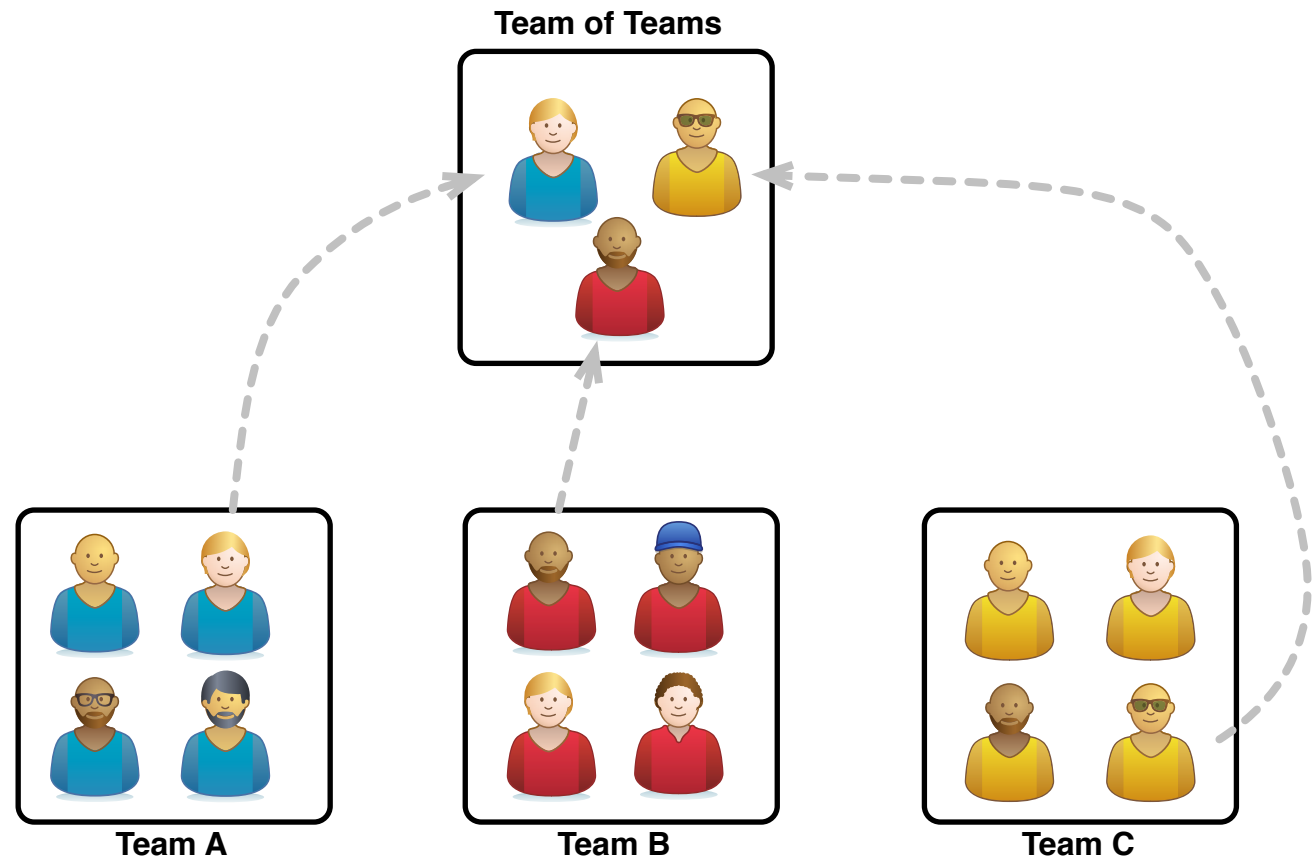
The team

- Typically 5-9 people
- Cross-functional:
 - Programmers, testers, user experience designers, etc.
- Members should be full-time
 - May be exceptions (e.g., database administrator)
- Teams are self-organizing
 - Ideally, no titles but rarely a possibility
- Membership should change only between sprints



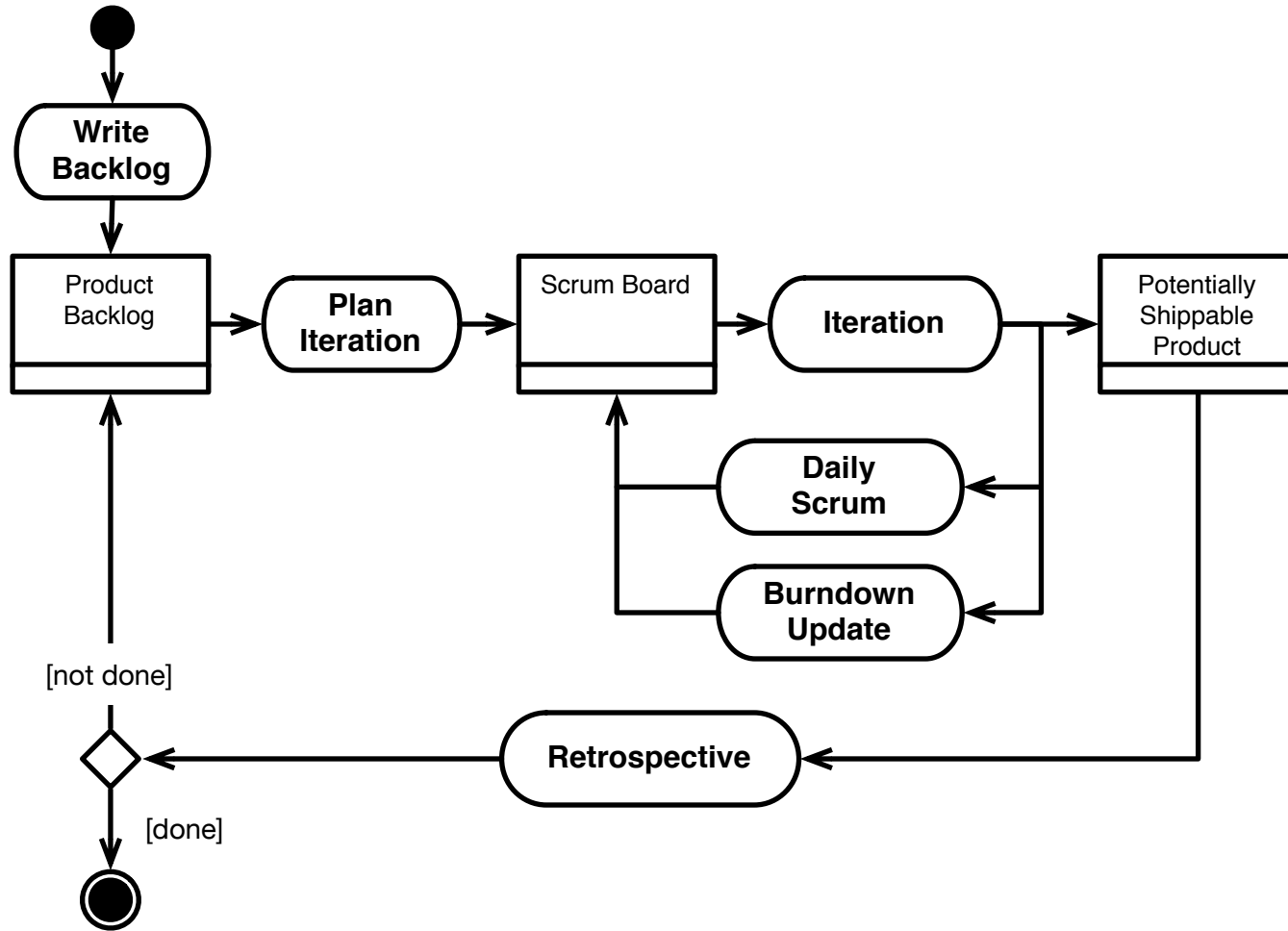
Team of Teams

- For large projects teams of teams can be organized:



The Scrum Process

The Scrum Process



The Product Backlog

User Stories

- The three Cs:
 - **Cards:** traditionally on note cards with annotations
 - **Conversation:** details behind the story come out during conversation with the product owner
 - **Confirmation:** details take the form acceptance tests confirm the story was understood right and coded correctly

As a Client of the Hotel I want to cancel a reservation

As a <role> I want to achieve <goal> (so that <reason>)



What Makes a Good Story

Independent	to simplify estimation
Negotiable	stories are not contracts; leave space for negotiation
Valuable	to the customer
Estimatable	they are the basis for planning
Sized Appropriately	break them into smaller ones, if necessary
Testable	

source



Some Advantages

- Easy to comprehend and manage
- Right size for planning (btw, similar approach to FP)
- Encourage iterative development
- They support opportunistic development (top-down and bottom-up approaches)
- Stories support participatory design
- Verifiable



The Product backlog

- The Product Backlog is a list of user stories, possibly annotated with additional information, such as:
 - Initiator
 - Priority
 - Cost
 - Owner
 - How to demo/test
 - ...

Sprint Planning

Sprints

- Scrum projects make progress in a series of “sprints” lasting between 2 and 4 weeks
- Emphasis on rhythm
- Product backlog items are allocated and developed during a sprint
- Once items are allocated, during a sprint no changes are possible
- Plan sprint durations around how long you can commit to keeping change out of the sprint

Sprint Planning

- Goal: collaboratively decide what stories end up in the sprint
- Activities involved
 - **Prioritize:** decide what the sprint will focus on (short narrative) and select stories accordingly
 - Break users stories into **tasks**
 - **Estimate** using planning poker (Delphi)
- Constraint: put all (and only) the stories the team can commit to

As a **Client of the Hotel** I want to **cancel a reservation**

Implement Reservation Class

Add view for cancelation

...

Estimating

- **Effort:**
 - simpler for people accustomed to traditional software development practices
- **Story points:**
 - abstract dimensionless measurements
 - provide a relative ordering of user stories
 - allows to compute velocity (how fast the team implements story points) and, therefore, to the number of story points allocatable in a sprint
 - defines an “algebra” of story points (10 story points require the same time as two 5 story points user stories)



Planning Poker® - an example



Estimator	Round 1	Round 2
Susan	3	5
Vadim	8	5
Ann	2	5
Chris	5	8



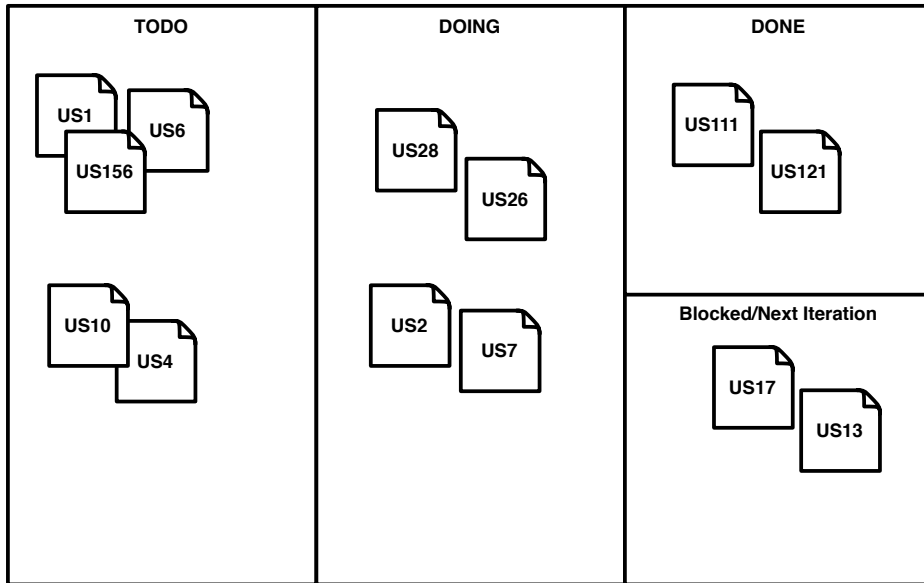
© Mountain Goat Software, LLC

<http://store.mountaingoatsoftware.com/products/planning-poker-cards>

Running the Sprint and Tracking

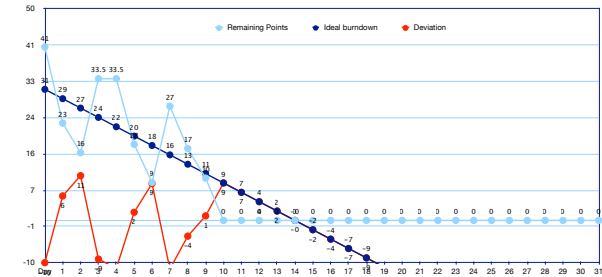
Scrum Tracking

- Two main goals:
 - Keep track of the work (todo, doing, done, blocked)
 - Measure the team “velocity” (sprints of fixed length means velocity is the important measure)



Sprint Detail

Sprint Number	1
Duration	14
Allocated Points	31
Ideal Average Point Burndown	2.21428571428571



TASK ID	RESP	TASK	Actual Points	Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
1	1	Migration of events data in production database	10	2	2	2	2	1	1	1	1	1	1																								
2	1	Table ordering through javascript/ajax	5	5	0	0	0	0	0	0	0	0	0																								
3	1	Views with pending admissions	3	3	3	3	3	3	0	0	0	0	0																								
4	1	Views with pending renewals	3	3	3	3	3	3	0	0	0	0	0																								
5	1	Views with inactive members	3	3	3	3	3	3	0	0	0	0	0																								
6	1	Export data to CSV for merge printing	3	3	1	1	1	1	1	1	1	1	1																								
7	1	"Approximate" bit on dates related to events (only)	2	2	2	2	2	2	2	2	2	2	2																								
8	1	Management of interest bits	1	1	1	1	1	1	1	1	1	1	1																								
9	1	Management of remarks field	1	1	1	1	1	1	1	1	1	1	1																								
10	1	User Interact/Layout	2	2	2	0	0	0	0	0	0	0	0																								
11	1	Table paging	5	5	5	0	0	0	0	0	0	0	0																								
12	1	Fix problem with search and locale	3	3	3	3	3	3	3	3	3	3	3																								
13	1	Make search box standard on all tables	7.5	7.5	7.5	0	0	0	0	0	0	0	0																								
14	1	Application title	2	2	0	0	0	0	0	0	0	0	0																								
15	5	Authentication - with roles and multiple users	15	15	7	7	7	7	7	7	7	7	7																								
16	5	Distinguish actions from links	3	3	3	3	3	3	3	3	3	3	3																								
Remaining Days				14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16	-17		
Remaining Points				41	23	16	33.5	33.5	18	9	27	17	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ideal Burndown				31	28.78	26.5	24.3	22.1	19.9	17.7	15.5	13.2	11.0	8.85	6.64	4.42	2.21	-0.0	-2.2	-4.4	-6.6	-8.8	-11.0	-13.1	-15.1	-17.1	-19.1	-22	-24	-26.1	-28.1	-31	-33.1	-35.1	-37		
Deviation				-10	5.28	10.5	9.2	11.1	1.96	8.11	1.1	-3.7	1.07	6.85	6.84	4.42	2.21	-0.0	-2.2	-4.4	-6.6	-8.8	-11.1	-13.1	-15.1	-17.1	-19.1	-22	-24	-26.1	-28.1	-31	-33.1	-35.1	-37		

Adapting/Computing Speed

- What if:
 - Speed is **higher** than expected: add stories
 - Speed is **lower** than expected: remove stories
- Speed is estimated according to historical data
- Velocity tends to converge after a few sprints
- Initial estimation is a guess or uses effort

Adapting/Computing Speed

- Between sprints, it is possible to accommodate for changes in effort, by inserting effort in the estimation:

$$\text{efficiency}_i = \frac{\text{actual velocity}_i}{\text{actual effort}_i}$$

$$\text{est. velocity}_{(i+1)} = \text{efficiency}_i \times \text{effort}_{(i+1)}$$

Managing the Scrum Process

The Daily Scrum

- A daily 15-minutes stand-up meeting for the whole team (including product owner)
- Everyone is invited (but only to listening)
- Emphasizes commitment
- Answer to three questions:
 - What did you do yesterday?
 - What will you do today?
 - What is blocking your work? (impediments)
- Impediments become the responsibility of the Scrum Master

The Sprint Review

- Each sprint ends with a demoable product (potentially shippable product): a piece of software, a mockup, a manual, ...
- The Sprint Review has the goal of demoing the output of the sprint
- Everyone participates
- Informal if focuses on demos (rather than slides)

The Sprint Retrospective

- After every sprint the team takes a look at what is and is not working
- Not very long: 15–30 minutes
- Whole team participates
- One approach is start-stop-continue:
 - What we need to start doing
 - What we should stop doing
 - What we should continue doing

Software Development Practices

- According to Cohn SCRUM entails the adoption of some practices typical of Agile processes such as
 - Pair programming
 - Test-first approach
 - Automated testing and integration
 - Refactoring
- ... they are not necessary, but make SCRUM really effective