Agile Introduction

— Nanda

Agenda

Introductory topics:

- What is Agile?
- Why is Agile important?
- How hard is it to be Agile?

Iterative development

Overview of Scrum (one popular Agile method)

Agile Estimation exercise

Agile Basics

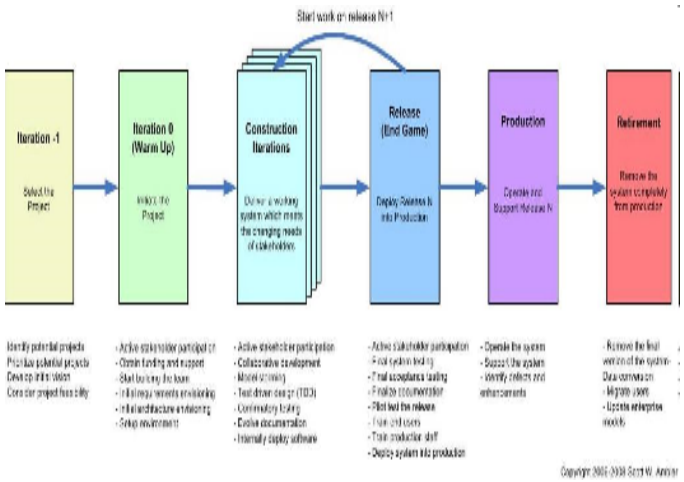
Agile Development – What does it mean?

Agile is a set of ***practices***, ***values***, and ***principles*** for software product development.

In software product development, we think about "methodologies," "activities," "interactions," "results, work products or artifacts;" we think about "processes" that we use to organize the work:

- documents
- meetings and reviews
- diagrams and models
- coding and user documentation standards

So will Agile Development define a new set of process activities? Not necessarily.



What is Agile? (Agile vs. Sequential Models or Frameworks)

Many of us are familiar with the Waterfall Model – it is a "framework" for the software development process

- Waterfall Model talks about "development activities through time"
- Waterfall Model talks about "teams of people"

Development activities	**Teams**
Divide the work into stages	A separate team of specialists for each stage
At each stage, the work is passed from one team to another	Some coordination is required for the handoff from team to team – using "documents"
At the end of all of the stages, you have a software product ready to ship	As each team finishes, they are assigned to a new product

What is Agile? (continued)

The core ideas in Agile Development:

- Adaptive
- Iterative/incremental
- People-oriented

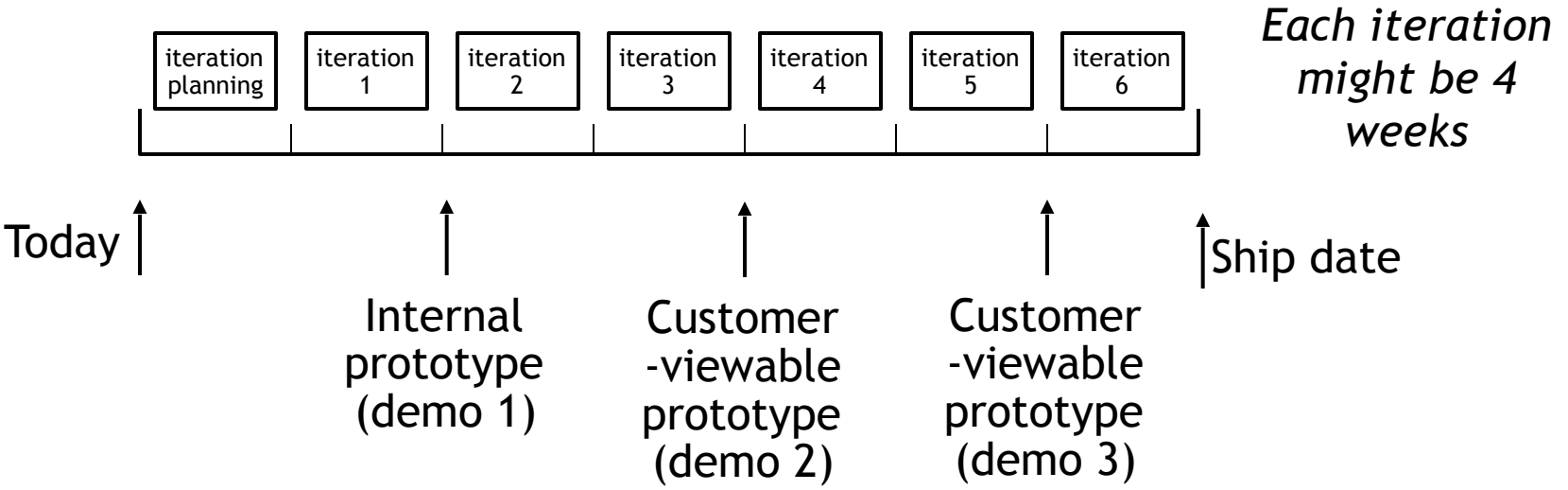
Adaptive means that the teams and the process should be flexible in the presence of "rapid-fire change".

Iterative and incremental means that Agile Development produces working products in stages – a growing set of "completed and working software".

People-oriented means the team organization and processes will support good people, who are the most important ingredient to project success.

Iterative development

One way to organize agile development is using short iterations:



Each iteration step:

- has some analysis, some design, some coding, some integration and testing
- executed by a cross-functional team
- delivers some kind of internally or externally usable functionality – intermediate demos or deliveries are possible!

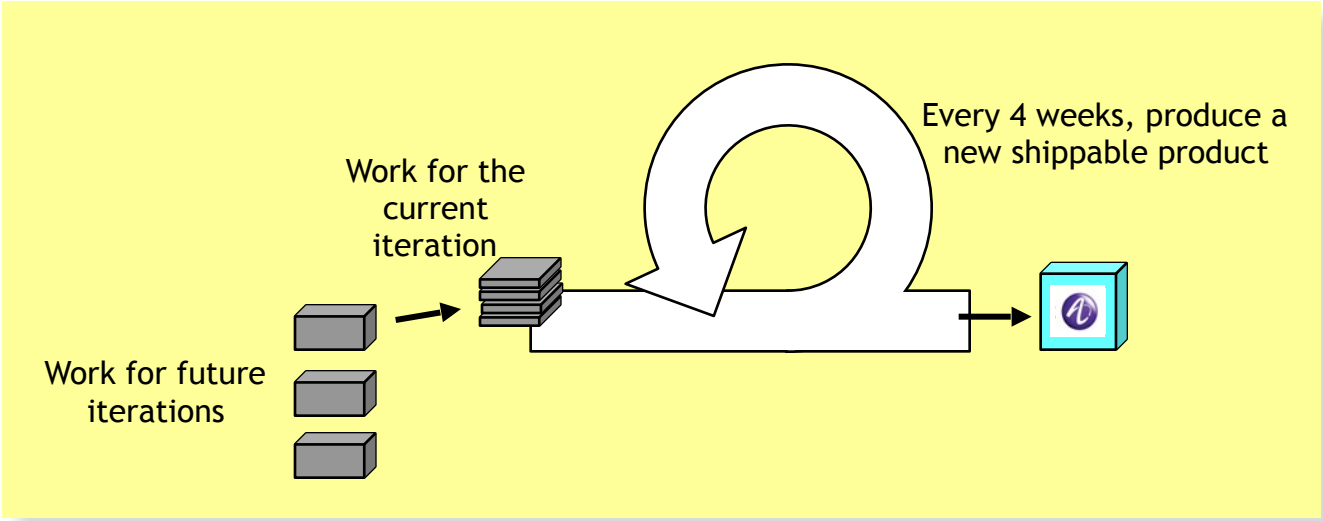
Question:

- Could we do a "demo" every iteration?
- Absolutely yes! The team gets practice at doing system integration

Main characteristics of Agile Development

Agile Development as a “software development framework” says:

- keep things small
- deliver partially-completed software frequently
- talk to the customer often
- write more code than documentation
- everyone on the team learns together



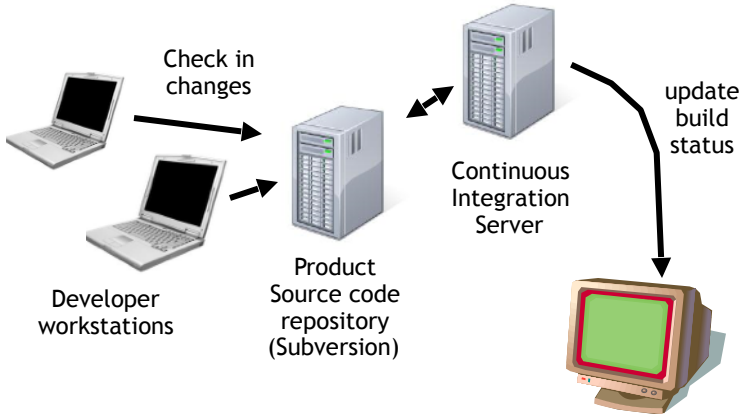
Agile Practices

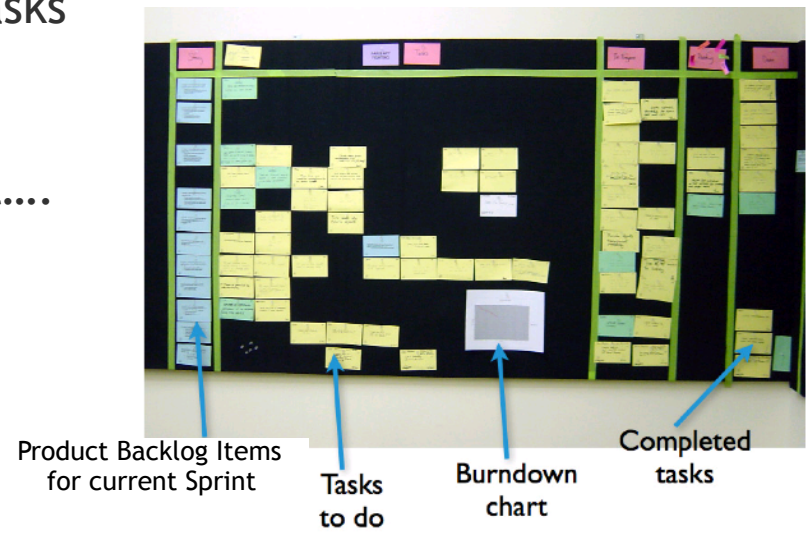
There are many Agile practices:

- short timeboxed iterations
- continuous integration
- daily unit testing
- regular retrospectives
- direct communication between developers and the customer or a customer surrogate
- a single list of features and tasks
- short-term estimation of development tasks
- information radiators
- refactoring

Will you use every Agile practice? Maybe not…. they are not all required.

What *is* required? Agile values…





Agile principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	9. Continuous attention to technical excellence and good design enhances agility.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	10. Simplicity--the art of maximizing the amount of work not done--is essential.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	7. Working software is the primary measure of progress.	11. The best architectures, requirements, and designs emerge from self-organizing teams.
4. Business people and developers must work together daily throughout the project.	8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Requirements process

There is no "standard way" to do requirements in Agile development

- Could be a normal "Software Requirements Document"
- But it is better to be more lightweight

One way to do requirements:

Start with a much slimmer "initial requirements document" at the beginning of the iterations…

- Initial list of overall "systems capabilities" – written in the form of User Stories
- Plus a section containing "global non-functional requirements" (security, reliability, performance, usability, etc.)

The list of system capabilities and global non functional requirements will be the first draft of the SRD.

In each iteration, elaborate a small set of the functional requirements (the high-priority behavior)

- This avoids creating a big requirements document too soon
- A good strategy is to delay writing most of the "fine details" in the requirements until the iteration when they will be implemented
- Why? Because you will have learned more about the problem…

For some key requirements, create some acceptance tests at the same time as you write the requirements

Agile – questions and challenges?

- **Documentation** – it is still important in an Agile project.
  - If it is the only kind of communication in your project, it isn't good
  - Real working code is more valuable than documents – less ambiguous
  - Documents – easy to leave something out, easy to misinterpret
- **Development plans** – also important in an Agile project
  - the format of an Agile development schedule is a bit different from a conventional project plan.
  - Development plan includes "iterations"
  - Each iteration gives the team has a chance to incorporate what they learn, rather than just following a non-adaptive plan
- **Contracts** – we expect to have contracts, but we need to talk with the customers as well.
  - Customer collaboration is one way to reduce development costs
  - Do you want to deliver "everything" the customer asked for in the original contract? No – if the customer no longer needs it, the extra code will increase maintenance costs
  - Always ask: Who needs this feature and how does it contribute to the value of the product?

Why is Agile Development important?

The world is a lot different today. A large feature set might only increase costs for the customer.

- There is a constant introduction of new technology
- New players enter the market,
- New requirements are added
- “Small is Beautiful”
- If we are listening to the customer, we will reduce our chances of being “blindsided” by a smaller, more flexible competitor
- Anything that helps reduce maintenance costs will contribute to the bottom line

How hard is it to be Agile?

"Don't do Agile, be Agile"

- Just doing "development in iterations" isn't enough

Agile Development is about:

- Keeping the process lightweight
- Making real progress in each iteration
- Communicating – face-to-face when possible
- Actively gathering customer input – early and often
- Being willing to make minor changes to your process

Agile Methods: Scrum

Agile methodologies

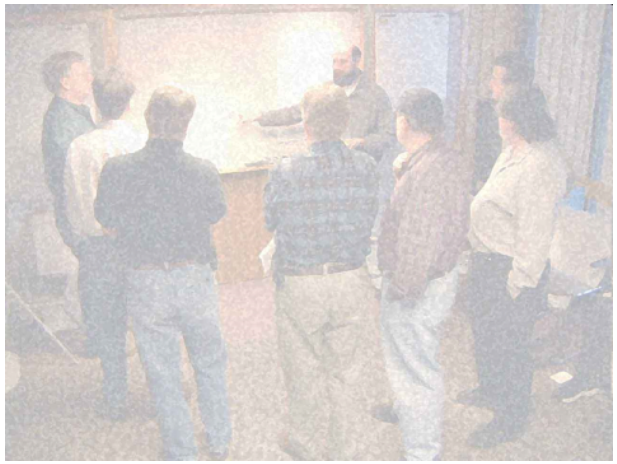
In this course, we will discuss the Scrum methodology

- Scrum has been around since the early 1990s
- The structure of Scrum is very simple (3 roles, 3 meetings)
- Scrum is not as “extreme” as some other methodologies

What is a Scrum?

- It is a meeting with attitude – good teamwork is necessary

a software scrum



a rugby scrum

Let’s Scrum

Scrum overview

The Scrum presentation is short and simple:

- Scrum iteration process
- Product Backlog
- Roles: Team Member, Product Owner, and Scrum Master
- Project estimation and iteration estimation
- Daily Scrum Meeting
- Management
- Retrospectives

Scrum iteration process

Scrum is designed to organize the work of a single cross-functional team

The team will do software product development this way:

1. Iteration planning – create a plan for one iteration
   - Select next features or sub-features to deliver (choose from highest priority items), define and estimate tasks, negotiate scope of the delivered product
2. Iteration execution – implement the items in the plan
   - Fill in missing requirements, design, code, integrate/build, and test the modules needed in the plan
3. Deliver the results of the iteration – give a demo

- Steps 1 – 3 will be executed many times – based on the Release Plan
- Each cycle is a fixed-length timebox:
  - Always end each iteration on schedule, even if it isn't complete
    - (Don't say – "we can finish everything in this iteration in 2 more days". Just deliver and run the next iteration planning meeting.)
  - The team learns to make good short-term estimates – so over time, most of the iterations will deliver as expected

Scrum Elements

THREE Roles

- Product Owner
- Scrum Master
- Team Member

THREE Meetings

- Planning (Release & Sprint)
- Daily Scrum
- Sprint Review

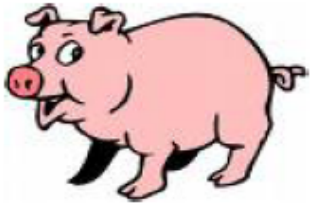
THREE Lists

- Product Backlog
- Spring Backlog
- Impediments List

For details, see
Scrum Guide: http://www.scrum.org/scrumguides

More on Scrum Introductopn

- Product Manager - acts in the role of the customer, adding new features to backlog, prioritizing work on the backlog
- Developers - estimate work items on backlog, develop product using highest priority items from backlog
- Scrum Master - keeps the team on track and removes obstacles. This is a damping capacitor, not an amplifier! Protect the developers from all external distractions.



http://www.youtube.com/watch?v=vmGMpME_phg

Scrum iteration process

The ***Product Backlog*** is the set of all features and sub-features that you know you need to do to build the product

- This is the "plan" for multiple iterations
- The items in the Product Backlog is ordered by priority – value to the customer
  - you want to deliver some value to the customer in each iteration, so you put the most important things early
- It is OK to add things to the Product Backlog at any time

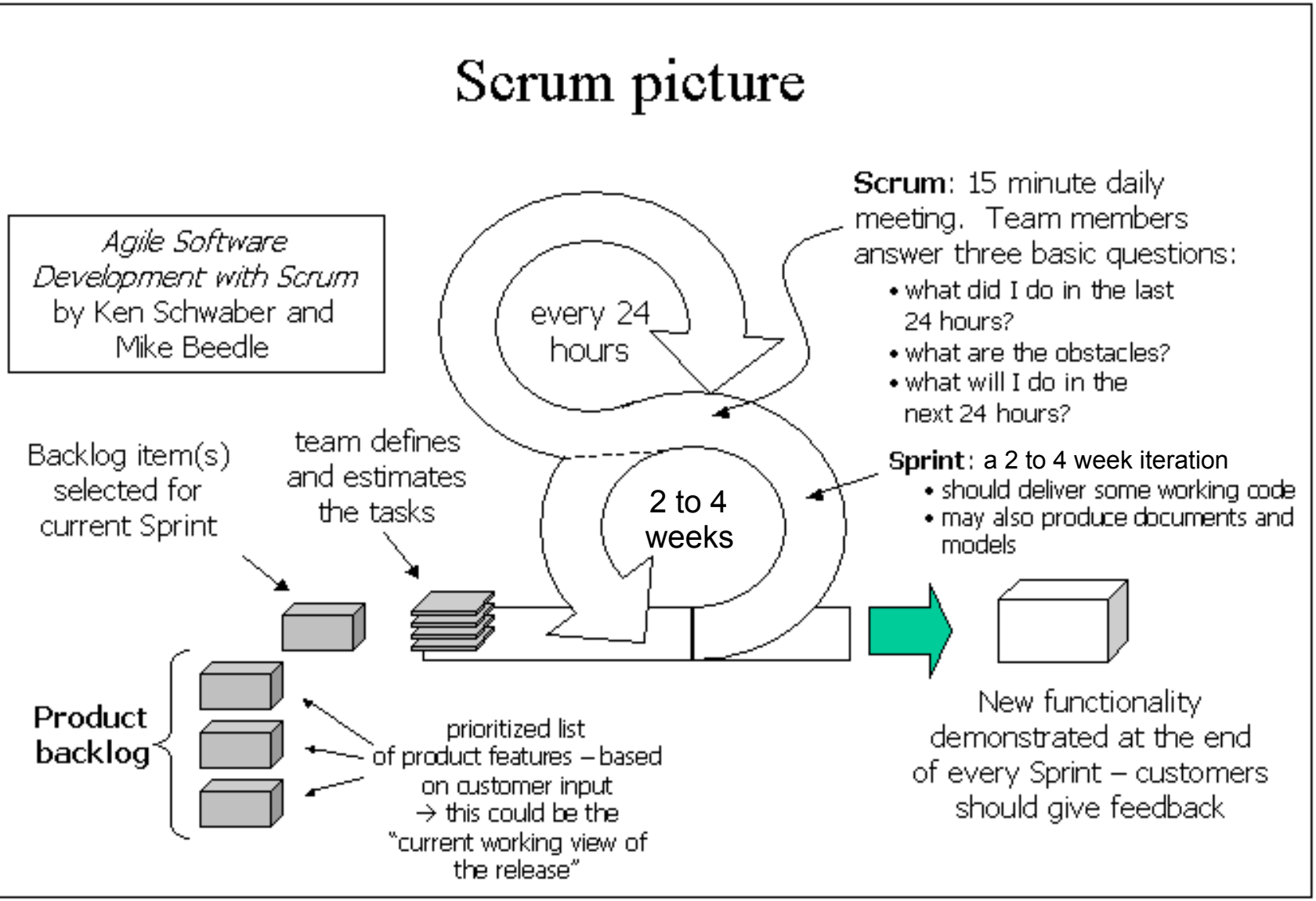
Backlog item	Prio	Size
Subfeature 1	1	5
Subfeature 2	2	8
Subfeature 5	3	13
Subfeature 4	4	1
Subfeature 3	5	2
...		

A Scrum iteration (called a *Sprint*) contains a list of tasks and work product outputs that will be done in a 4-week* timebox

- At the beginning of the 4 weeks, each team member has a pretty good idea of what they will be working on
- Management should not add new work product outputs to the Sprint – any new items should be added to the Product Backlog instead
- If new work items are important enough, they will get done in the next 4 week iteration

* (30-day iteration in the original Scrum articles – most teams use a 2-week to 6-week iteration)

Scrum picture



Product Backlog

What does a Product Backlog look like?

- It is a simple spreadsheet
- All “Product Backlog Items (PBIs)” are in priority order
- Some PBIs are the names of “customer features”
  - Could be a user screen, an interaction scenario or use case, a new report, a new algorithm
  - Much, much smaller than a telecom system feature
- Some PBIs are internal tasks that contribute to the value of the product
  - Can a design document be a PBI? Maybe.
  - If it is a document that nobody reads, leave it out (because you are Agile)
  - Can an early GUI prototype be a PBI? Certainly.

Effort estimates – each PBI should have an “estimated effort” that is assigned by the team

- Should managers do the estimation of Product Backlog Items? No, never.
- Estimates must come from the team – and they should be realistic

Project estimation and iteration estimation

The Product Backlog – managers and customers use it to set the working agenda of the development team

- Managers and customers work with Product Owner to set the priority of each item
- Development team estimates the size/effort for each item
- Even if the managers and customers don't like the estimates, they are not allowed to change them

Backlog item	Prio	Size
Subfeature 1	1	5
Subfeature 2	2	8
Subfeature 5	3	13
Subfeature 4	4	1
Subfeature 3	5	2
…		

Within an iteration, the team divides the Product Backlog Items into individual tasks – the "task view" is only used within the iteration

- Development team defines tasks and the estimated effort
- The list of tasks is flexible – new items might be discovered during the iteration, some items might be combined or eliminated
- Development team tracks all "tasks" on a Task Board
- Development team tracks progress with a burndown chart

Roles on a Scrum team

Product Owner

- Responsible for the ROI
- Available for the Team during the whole product development period
- Gets answers to all requirements questions
- Talks with customers and understands their priorities
- Keeps the Product Backlog current

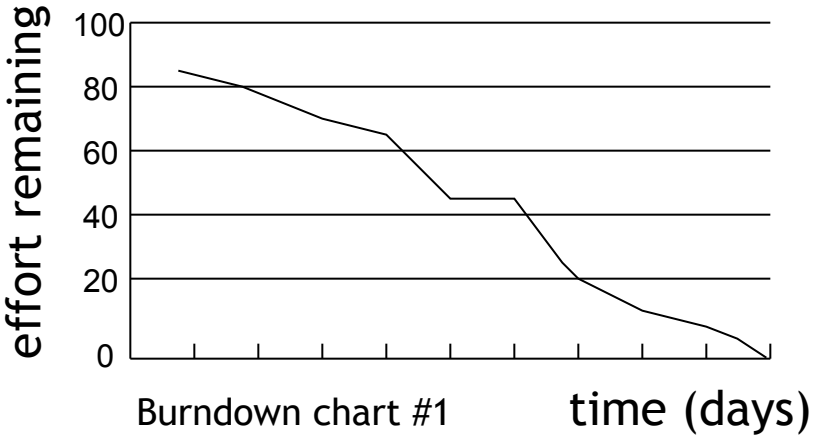
Scrum Master

- Scrum rules guardian
- Coach the team
- Removes impediments
- Prevents outside interference during an iteration
- Scrum Master is both a teacher and a referee

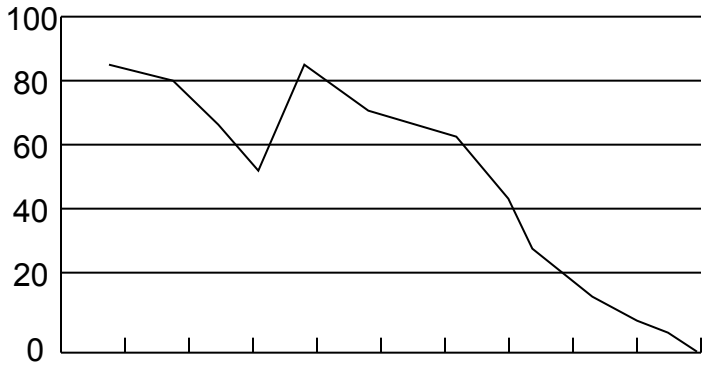
Burndown chart

Tracking an iteration:

- A burndown chart tracks the amount of estimated effort remaining in the current iteration
  - it should go down each day
  - but if you discover that something is missing, or you have mis-estimated a difficult task, it could go up
  - it's OK: better to acknowledge reality early
- Don't make your estimates too pessimistic
  - you will get a burndown chart that gets to zero well before the end of the iteration



Burndown chart #1



Burndown chart #2

Daily Scrum Meeting

The Scrum Team has two kinds of “once-per-iteration” meetings:

- An Iteration Planning meeting at the beginning of each Sprint
- A Sprint Review meeting at the end of each Sprint

In addition, the Scrum Team has one daily meeting: the Daily Scrum

- Daily Scrum is 15 minutes – no longer
- Everyone is supposed to speak:
  - “This is what I did yesterday.”
  - “Here is what I am planning to do today.”
  - “These are the obstacles in my way.”
- No problem solving in the meeting – everything is taken offline later.

What is the purpose of the Daily Scrum? To make sure that problems and obstacles are visible to the team

- Obstacles are valuable input for managers

Retrospectives

One important idea in Agile Development: take time to reflect and learn

- Iteration is good, because you have a natural breakpoint to apply some of what you have learned

In Scrum (and many other Agile methodologies), the team runs a Retrospective meeting at the end of each iteration

- A Retrospective is like a post-mortem, but it isn't dead yet
- An end-of-iteration retrospective meeting takes an hour or two

The end-of-iteration Retrospective meeting is a chance to learn what worked well, what should be changed

- don't use a Retrospective to blame team members or managers for all of the problems – focus on fixing the process

Retrospectives

The Retrospectives Prime Directive:

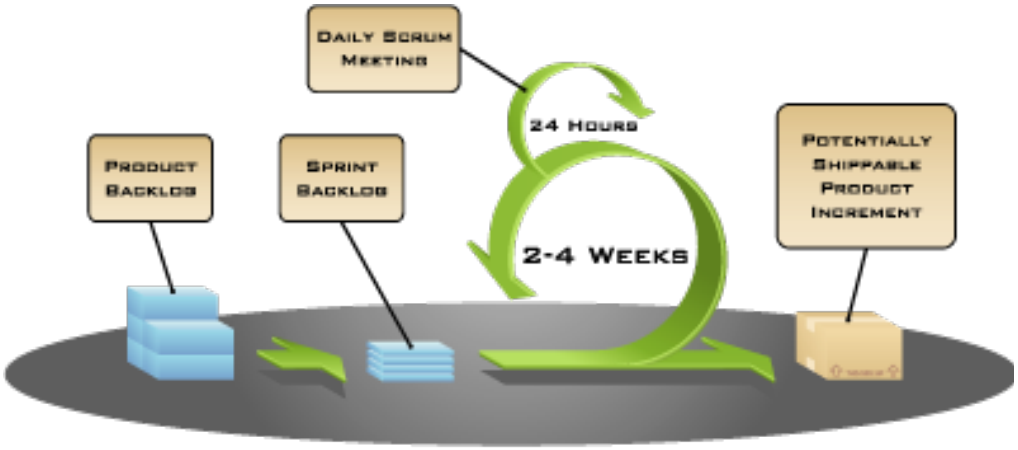
> *Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand.*

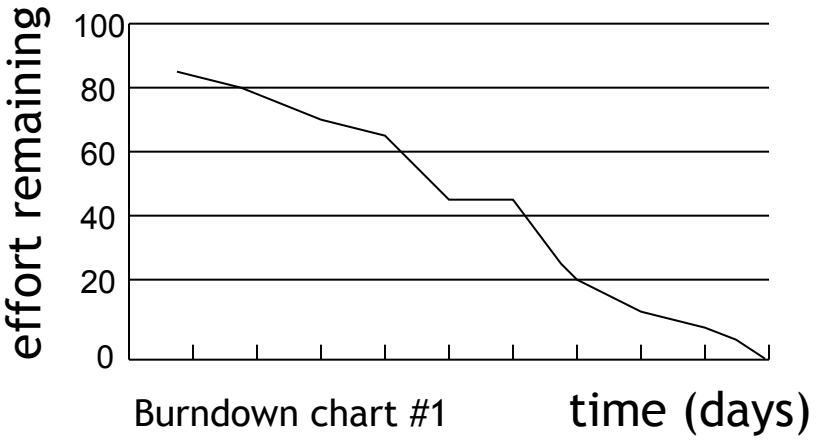
(From Norm Kerth's book on Project Retrospective
See also http://www.retrospectives.com)

Why this rule? The goal of a retrospective is to *improve the process*, not to assign blame for the problems

Scrum summary

- Scrum is a “team-oriented” Agile methodology
- Short timeboxed iterations
- Each iteration produces some real software that has value to the customer
- Each iteration has
  - iteration planning
  - development work
  - iteration review
- All estimation is done by the team
- Within a Sprint, the progress is tracked using a burndown chart
- Product Owner determines the priorities in the Product Backlog (list of things to build)
- Scrum Master helps enforce the rules of Scrum
- There is a 15-minute daily meeting to report what was done and identify obstacles





Scrum Adoption

This high-level presentation of Scrum has focused on the simple Scrum structure and rules.

But *running Scrum* and *getting the core benefits from it* is ***HARD***

- ***Cross-functionality and self-organization***
- ***Transparency, Inspect and Adapt***
- ***Continuous improvement***

More on Scrum

There are many more things to learn about Scrum. We will touch on some of those in the discussion:

- How to run big projects (more than one Scrum team)
- Can managers interfere with a Sprint in progress?
- Fixed schedule and fixed cost contracts
- Part-time team members (specialists)
- A single Scrum team with members in multiple locations
- Architecture
- CMMI / ISO9000 standards

Some references:

- Scrum Guide: **http://www.scrum.org/scrumguides**
- Scrum Primer: **http://scrumtraininginstitute.com/home/stream_download/scrumprimer**
- Craig Larman's books on Safari: **http://techbus.safaribooksonline.com/9780321685117**

Summary

Agile Development – it is a different way of organizing product development

- Emphasizes iterative development with small cross-functional teams instead of waterfall development in separate silos
- At the end of each iteration, there is some functionality that is "done"
- Better communication, faster feedback

Why do we need to be Agile?

- Products are different from 20 years ago
- Customers are in a changing environment – and our processes need to be working at the same pace
- Working in short cycles reduces time to market and time to quality

There are many ways to be Agile

- Scrum is the most popular Agile approach
- Scrum promotes some good Agile practices
  - Small cross-functional teams, short timeboxed iterations, adaptive planning, single product backlog, frequent integration, test automation

Agile Estimation

Estimation – Poker Planning

Scrum is fun, so estimation is a game.

- The Delphi technique brought up-to-date
- Visibility of differences
- Drives to consensus
- Breaks down linear thinking

References:

Mike Cohn, Agile Estimating and Planning, Prentice Hall PTR, Upper Saddle River, NJ, 2005. Read Chapter 6: Techniques for Estimating online.

J. W. Grenning, Planning Poker, 2002, http://www.objectmentor.com/resources/articles/PlanningPoker.zip

Units of Estimation

- PBI: Ideally, estimation/story points: relative estimation
- Alternatively, “ideal hours”
- For Sprints, use “ideal hours” and it eventually can come down to hours
- However, try to learn how many story points you can do per Sprint.
- This ratio is called the team’s velocity

Simple Rules of the Game

- Each participant gets a deck of estimation cards.
- The moderator (usually the product owner or an analyst), presents one user story at a time
- The product owner answers any questions the team might have.
- Each participant privately selects a card representing his or her estimate.
- When everybody is ready with an estimate, all cards are presented simultaneously.
- In the (very likely) event that the estimates differ, the high and low estimators defend their estimates.
- The group briefly debates the arguments.
- Make a new round of estimation.
- Continue until consensus has been reached.
- The moderator notes the estimate, and the group continues with the next user story.

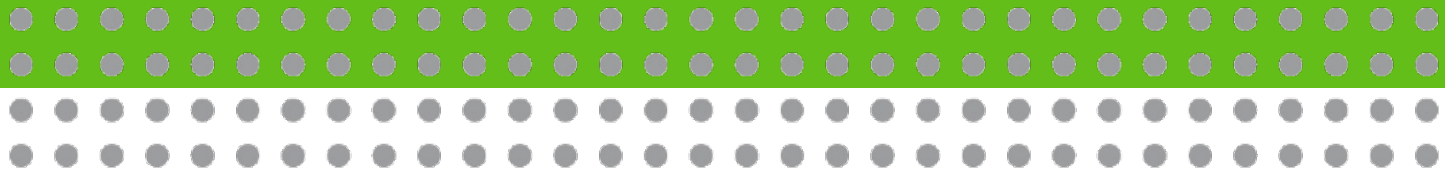
Planning Poker Tips

- Baseline the rating system by scanning all user stories and assigning the estimate value of "1" to the simplest story
- The cards reflect a moderately narrow range of numbers ranging about one order of magnitude.
- Overly large user estimates suggest that the PBI/task can be split into multiple items.
- Consider time boxing the debate period. The goal is that the technique be simple and lightweight
- The technique works best with three to five participants representing architecture, development, testing, and deployment.
- Work from a list that has already been sorted by business value, and avoid focusing on business value during the work queue priority ordering.
- Use the three-finger test as an audit on the confidence of whether the team will accomplish all Tasks during a sprint

Exercise: Kitchen Remodeling

- Install new hardwood floor
- Refinish (remove, sand, repaint) the cabinets
- Install granite countertop instead of tile
- Repaint entire kitchen
- Lay shelf paper
- Install recessed lighting
- Replace electric stove
- Install built-in refrigerator
- Install a new oven
- Plumb the island and add sink
- Replace simple window with a bay window

Backups & More Ressources



BACKUPS

What is Agile (Development)?

It comes down to a few basic concepts (not necessarily independent):

- Self-organization
- Constant feedback
- Ability to respond to change
- Respect
- Communication
- Rhythm (e.g., time boxing)
- Flow value at the pull of the customer
- Push decisions as close to the work as possible
- Make decisions as late as possible

As typically used, “Agile” refers to a set of development methodologies and frameworks (software development methodologies) based on and implementing the above principles.

It is a light weight and at the same time very structured development framework – marking the current step in software development models (from waterfall/sequential models – to V-shaped – to incremental – to spiral - to iterative - to “*agile*”). **Note, that agile is more than a process model.**

Requirements process

There is no “standard way” to do requirements in Agile development

- Could be a normal “Software Requirements Document”
- But it is better to be more lightweight

One way to do requirements:

- Initial list of overall “system capabilities” – written in the form of “User Stories” - http://www.agilemodeling.com/artifacts/userStory.htm
- Plus a document of “global non-functional requirements” (security, reliability, performance, usability, etc.)
- In each iteration, some of the user stories are elaborated – don’t create a lot of requirements detail too soon
  - Delay writing the requirements details – especially if they might change
  - You will have learned more about the system as you go along – late requirements are often better
- For some key requirements, create some acceptance tests at the same time as you write the requirements
  - The tests will be useful in the continuous integration process

Roles on a Scrum team

There are three important roles:

- **Team Member** – the people who do all of the hard work.
  - The team will have a range of experience and skills
    - one or two team members with good architecture/design experience
    - one or two team members who know a lot about test strategies
  - But every team member is prepared to help do activities outside of their main area of expertise
    - when testing that needs to be done before the end of the iteration or new user screens to be designed, anyone in the team could jump in to contribute to the work
  - Normal team: 5 to 9 members, all working in a single location
- **Product Owner** – the single person who interacts with customers and product management (a very difficult job)
- **Scrum Master** – the single person who enforces the Scrum process rules

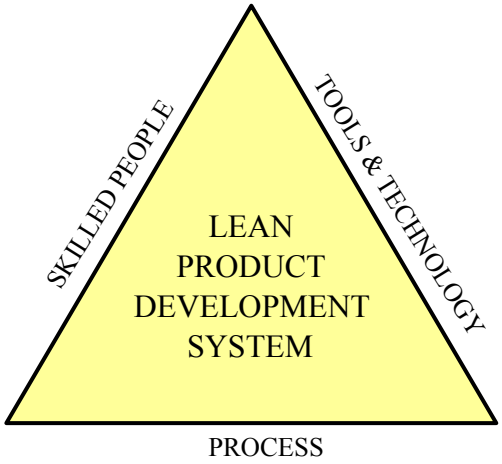
Example of self-organization

The Nokia Test and the Key Agile Concepts

- Iterations
- Expanding scope of Done to deployment
- Up-front specifications w/User Stories
- Product owner who plans
- Up-front Product Backlog
- Up-front estimates
- Business-oriented burndown chart
- Team disruption

What is Lean?

- Lean is a more complex (production) system aimed at adding value for the end user and customer
- To do that,
  - We eliminate waste
  - We eliminate inconsistency
  - We smooth and optimize the production flow
  - We are constantly improving: Kaizen



More Resources

Scrum video:
http://www.youtube.com/watch?v=vmGMpME_phg

Scrum resources:
http://www.scrumalliance.org/resources

Scrum user guide:
http://www.scrum.org/storage/scrumguides/Scrum%20Guide.pdf

Key Scrum Concepts:
http://www.scrumalliance.org/pages/what_is_scrum

More videos:
http://www.infoq.com/presentations/Agile-Management-Google-Jeff-Sutherland

http://video.google.com/videoplay?docid=-7230144396191025011#