XP: Extreme Programming & Agile Development

Plan, code, test, and repeat till the customer is satisfied

Sources: O'Reilly's Extreme Programming and general industry knowledge of XP & Agile Development

Major Features of XP

- Lightweight planning and tracking
- Agile for adaptability
- Unit testing along the way
- "A little bit of planning, a little bit of coding, and a little bit of testing let you decide if you're right or wrong while it's still cheap to change your mind."
- Freedom to Experiment – always asking if there is a better way to do something – Innovation Welcome!

XP's Core Values

- Communications
- Feedback
- Simplicity
- Courage

Communications

- Essential to any project
- Allows adjustment to change
- Developers know what to do & Customers know when it will be done
- Hidden or ignored information can sink a project

Feedback

- Ask questions and learn from the answers
- Only way to know if code works is to test it
- Only way to know what someone wants is to ask them
- Tests should ensure the user's needs are met
- Sooner you get feedback, the sooner you can make changes to accommodate it
- Only way to know if code works is to test it

Simplicity

- Build the system that really needs to be built
- Only solve today's problems today
- Complexity costs and predicting the future is difficult
- If you have good communications and feedback, you will better know what is really needed

Courage

- Make hard decisions when necessary
- Be honest when reporting progress
- If you need help ask for it
- If code needs improvement or fixing, just do it – it's a team effort
- Decide what you can deliver and do it

Scrum

- **Scrum** (rugby) "… is a method of re-starting play … utilised (sic) either after an accidental infringement or when the ball has gone out of play… The scrum then 'engages' with the opposition team so that the player's heads are interlocked with those of the other side's front row." Wikipedia
- **Scrum** (development) "Scrum is an iterative and incremental agile software development method for managing software projects and product or application development." Wikipedia

Roles

- Customers – control what stories to be worked on
- Developers – determine tasks and make estimates for overall stories
- Scrum Master – tracks progress on active tasks leading up to story completion, along with backlog of uncompleted tasks
- Coach – mentor to assist teams in progressing

Sprints: the Iterations of Agile Development

- Planning – determine which Customer Stories & Developer Tasks to work on with estimates of time to accomplish
- Create Tests to effectively verify each Story
- Execution of Tasks - coding
- Integration & build of code
- Testing & demonstration to customer
- Then usually back to the top for additional Sprint iterations
- Once enough functionality is complete - Release!

Sprints

- Typically every 1-3 weeks in most companies – we will target 2 weeks?
- Each includes all the previously mentioned steps: Planning, Coding (completing tasks), Integrating, and Testing
- After a Sprint, apply what was learned and use that to better plan the next set of tasks to work on. For example, fix bugs found in testing and adapt features.

Sprint Planning

- Customer creates **Stories** to describe features & specifications
- Developers estimate time to implement a **Story** by…
- Breaking down **Stories** into **Tasks** (steps) to accomplish them
- Developer estimates time for each **Task**
- Based on estimates, customer chooses which **Tasks** to be accomplished within the Sprint and they are divided among developers

Story Cards

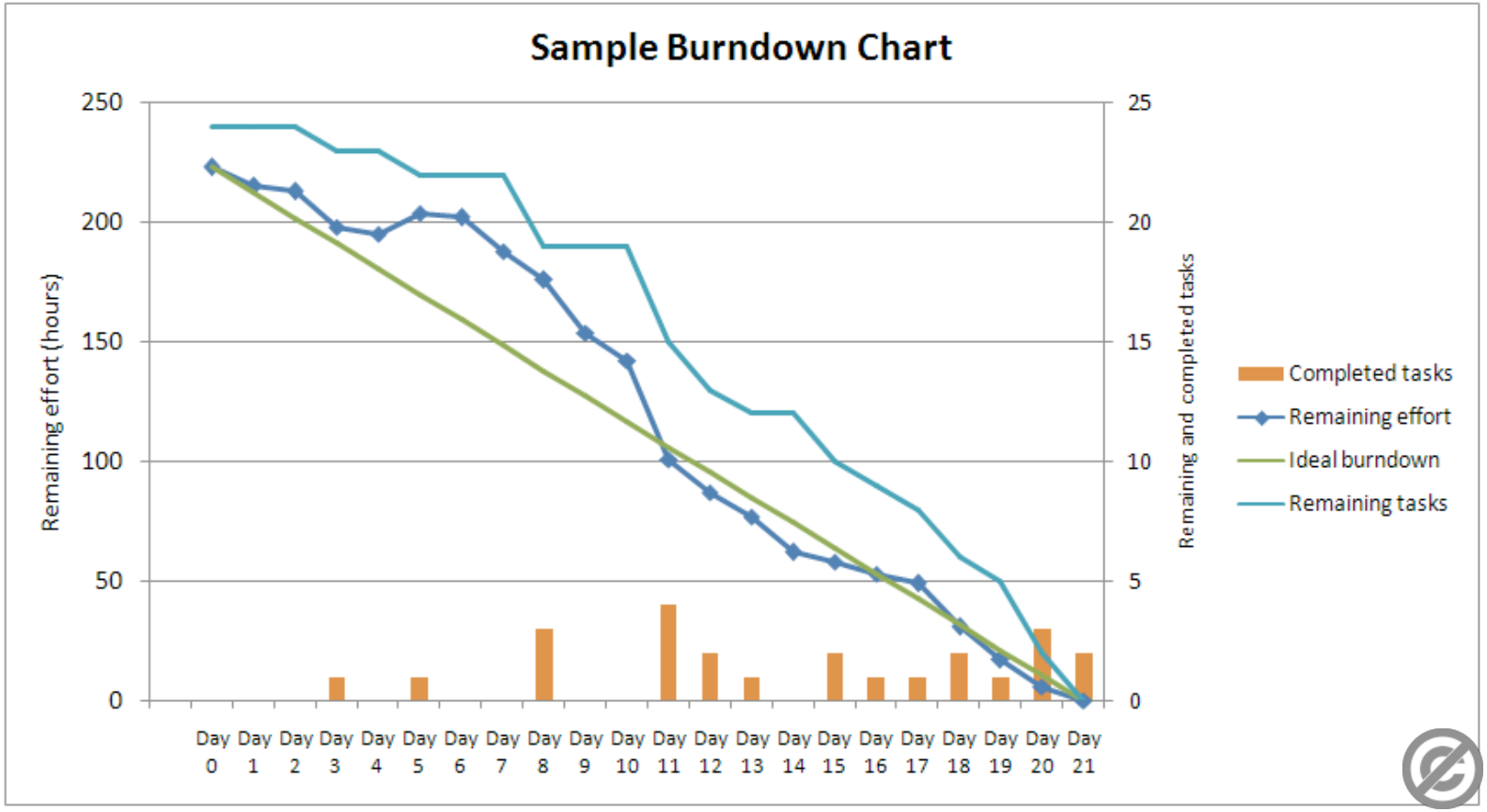
- Created by Customers to communicate what needs to be done
- Each describes a single feature in a story form – a sentence or two from the customers' perspective
- All features start with Story Cards and Developers can suggest stories
- Each story should stand on its own and be testable once completed
- Some stories may never be built – it's better to have just written it down than to have coded it

Task Cards (Post-Its)

- Created by Developers to describe how stories should be accomplished
- Given a story, Developers break it into tasks, sketching out its implementation details
- Every task is related to a story
- Tasks should be small, ideally a few hours to complete
- Every story requires a task of writing its acceptance tests with the customer

Tasks assembled in a backlog

- Task completion allows the "Burndown" of the backlog – till done



Special cases:
Spike Solution / Proof of Concept

If a story is too hard to break into tasks easily or if a task is too hard to implement, then a "spike solution" or "proof of concept" experiment is done - a pair of developers write a portion of code to explore the problem and learn enough so it can be broken down into tasks or determined unfeasible.

Special cases:
The First Iteration

- Should choose several small tasks to lay out the basic architecture of the project, end to end as much as possible
- Gathers real data on how to proceed
- Building this skeleton helps fine tune your overall processes like source control, build & release before things get too complicated to change
- Also identify any possible show-stopper risks, so those can be addressed and solved upfront. Better to know their cost earlier than later.

Working on Tasks during a Sprint

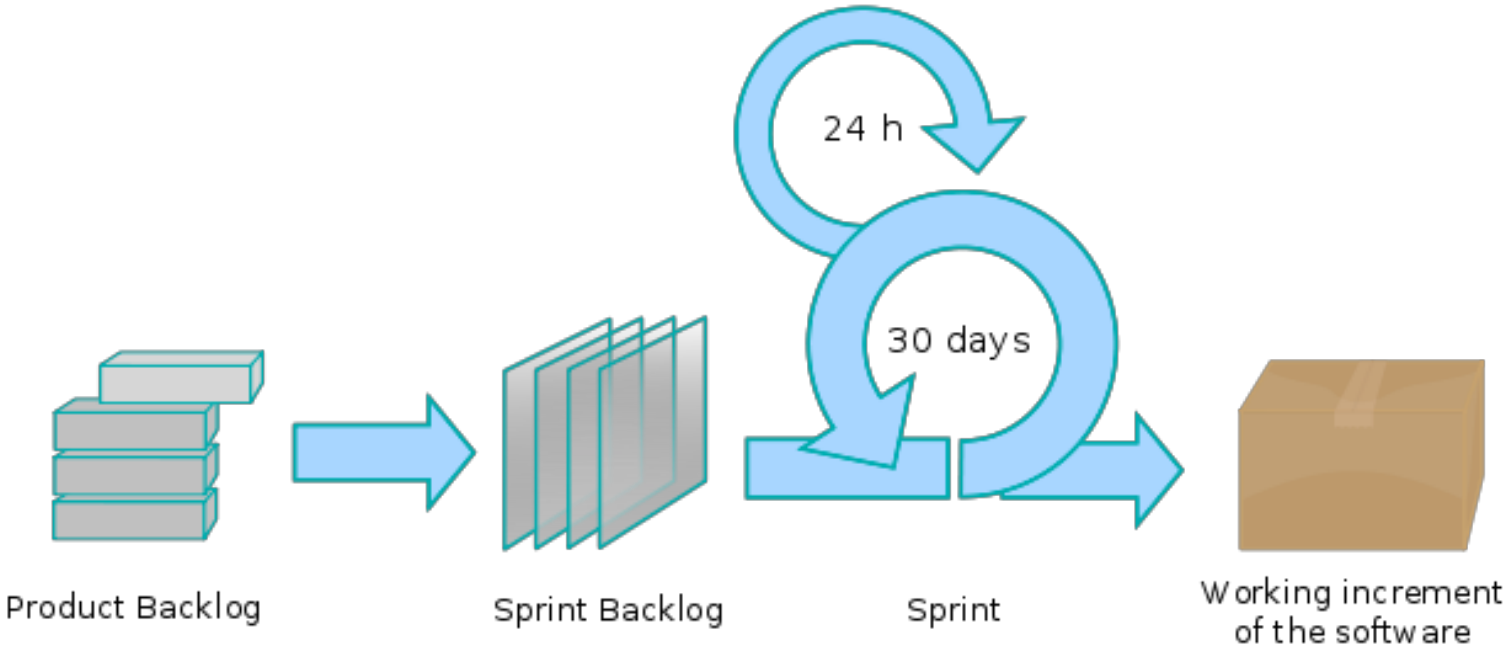
- Start with the riskiest and most important tasks at the start, leaving easier ones for later
- Work on only one task at a time, keep focused
- If you get stuck, swap tasks with another team member to get a fresh perspective
- Have regular "stand-up" meetings

3 Questions

Stand-Up (Daily Scrum) Meeting

- Meeting is kept short by requiring all attendees to stand, usually held daily
- Each team member answers 3 questions:
  - What I did yesterday?
  - What I am doing today?
  - Is there anything blocking me?
- This let's everyone know what is going on and allows suggestions to reuse code or help solve blocking issues

Scrum Process



By Lakeworks (Own work) [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC-BY-SA-3.0-2.5-2.0-1.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons from Wikimedia Commons

Sprints: the Iterations of Agile Development

- Planning – determine which Customer Stories & Developer Tasks to work on with estimates of time to accomplish
- Create Tests to effectively verify each Story
- Execution of Tasks - coding
- Integration & build of code
- Testing & demonstration to customer
- Then usually back to the top for additional Sprint iterations
- Once enough functionality is complete - Release!

Schedule for a 2 week Sprint

This is a proposal – any suggestions…

- 1st Monday: Sprint Planning meeting, determine Stories and Tasks
- 1st Thursday: Story test cases finalized
- 2nd Thursday: integration, build & testing
- 2nd Friday: demonstration of new features (stories) and identification of fixes & tasks to go into next Sprint - or are we ready to release?
- Stand-up meetings: 1st Tuesday & Thursday, 2nd Monday & Thursday

Next Activity: Start Sprint Planning:

- Write/Research the Customer Stories for our Projects
- Review those Stories with other teams for feedback and make them concise
- Finalize set of Stories to start with
- THEN we can start breaking them down into development tasks

Questions??