Agile Models

What is Agile Software Development?

- Agile: Easily moved, light, nimble, active software processes
- How agility achieved?
  - Fitting the process to the project
  - Avoidance of things that waste time

Agile Model

- To overcome the shortcomings of the waterfall model of development.
  - Proposed in mid-1990s
- The agile model was primarily designed:
  - To help projects to adapt to change requests
- In the agile model:
  - The requirements are decomposed into many small incremental parts that can be developed over one to four weeks each.

Ideology: Agile Manifesto

- Individuals and interactions *over*
  - process and tools

http://www.agilemanifesto.org

- Working Software *over*
  - comprehensive documentation
- Customer collaboration *over*
  - contract negotiation
- Responding to change *over*
  - following a plan

Agile Methodologies

- XP
- Scrum
- Unified process
- Crystal
- DSDM
- Lean

Agile Model: Principal Techniques

- **User stories:**
  - Simpler than use cases.
- **Metaphors:**
  - Based on user stories, developers propose a common vision of what is required.
- **Spike:**
  - Simple program to explore potential solutions.
- **Refactor:**
  - Restructure code without affecting behavior, improve efficiency, structure, etc.

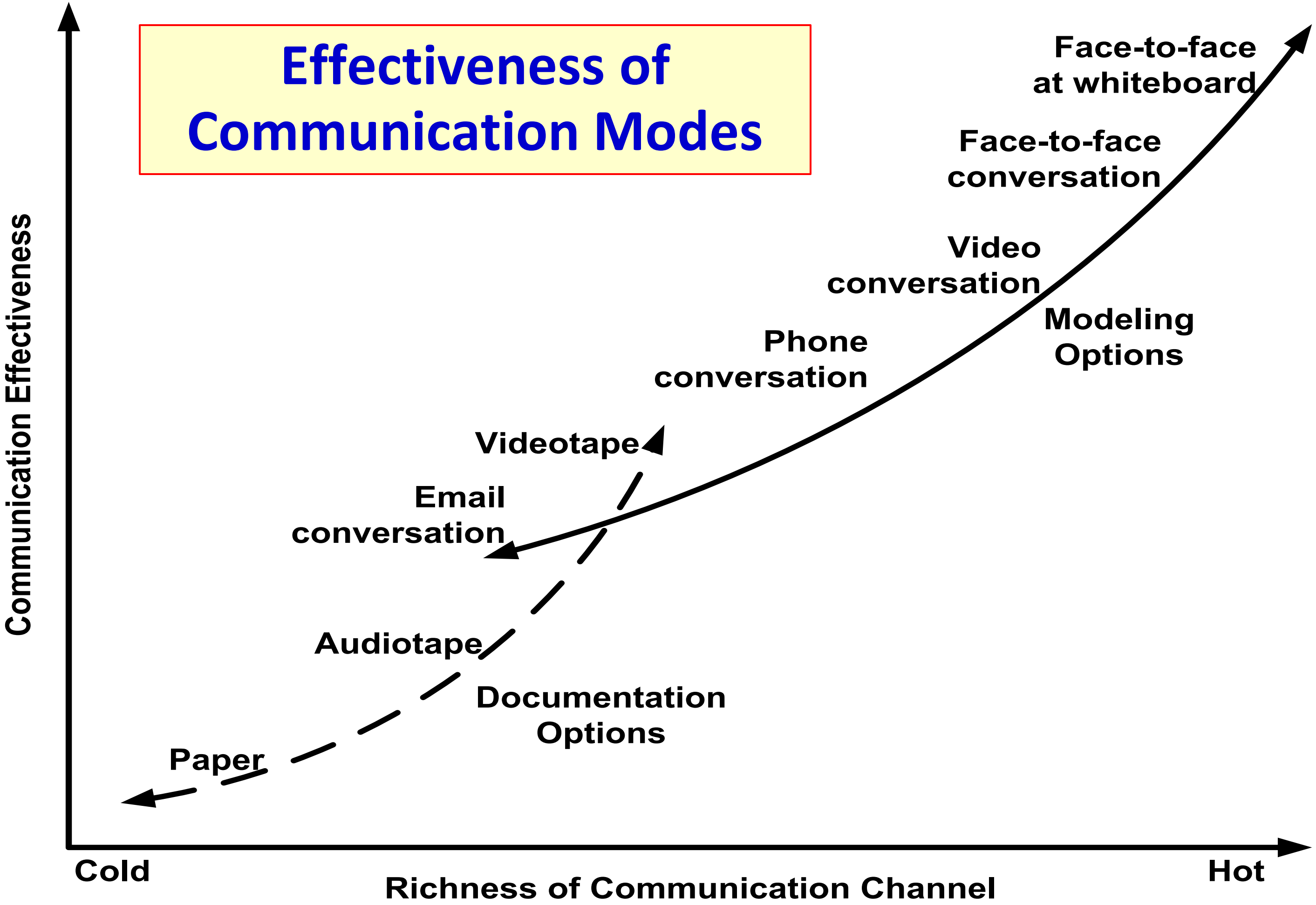
Agile Model: Nitty Gritty

- At a time, only one increment is planned, developed, deployed at the customer site.
  - **No long-term plans are made.**
- An iteration may not add significant functionality,
  - But still a new release is invariably made at the end of each iteration
  - Delivered to the customer for regular use.

Methodology

- **Face-to-face communication favoured over written documents.**
- To facilitate face-to-face communication,
  - Development team to share a single office space.
  - Team size is deliberately kept small (5-9 people)
  - This makes the agile model most suited to the development of small projects.

Effectiveness of Communication Modes



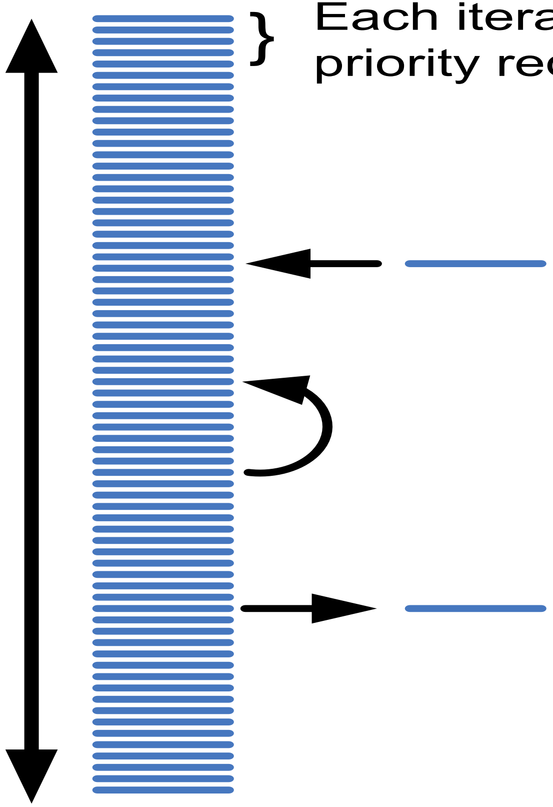
Agile Model: Principles

- The primary measure of progress:
  - **Incremental release of working software**
- Important principles behind agile model:
  - Frequent delivery of versions --- once every few weeks.
  - Requirements change requests are easily accommodated.
  - Close cooperation between customers and developers.
  - Face-to-face communication among team members.

Agile Documentation

- Travel light:
  - You need far less documentation than you think.
- Agile documents:
  - Are concise
  - Describe information that is less likely to change
  - Describe "good things to know"
  - Are sufficiently accurate, consistent, and detailed
- Valid reasons to document:
  - Project stakeholders require it
  - To define a contract model
  - To support communication with an external grou
  - To think something through

Agile Software Requirements Management



High Priority

Low Priority

Requirements

Each iteration implement the highest-priority requirements

Each new requirement is prioritized and added to the stack

Requirements may be reprioritized at any time

Requirements may be removed at any time

Adoption Detractors

- Sketchy definitions, make it possible to have
  - Inconsistent and diverse definitions
- High quality people skills required
- Short iterations inhibit long-term perspective
- Higher risks due to feature creep:
  - Harder to manage feature creep and customer expectations

Agile Model Shortcomings

- Derives agility through developing tacit knowledge within the team, rather than any formal document:
  - Can be misinterpreted…
  - External review difficult to get…
  - When project is complete, and team disperses, maintenance becomes difficult…

Agile Model versus Waterfall Model

- Steps of Waterfall model are a planned sequence:
  - Requirements-capture, analysis, design, coding, and testing .
- Progress is measured in terms of delivered artefacts:
  - Requirement specifications, design documents, test plans, code reviews, etc.
- In contrast agile model sequences:
  - Delivery of working versions of a product in several increments.

Agile Model versus Iterative Waterfall Model

- As regards to similarity:
  - We can say that Agile teams use the waterfall model on a small scale.

Agile versus RAD Model

- Agile model does not recommend developing prototypes:
  - Systematic development of each incremental feature is emphasized.
- In contrast:
  - RAD is based on designing quick-and-dirty prototypes, which are then refined into production quality code.

Agile versus exploratory programming

- Similarity:
  - Frequent re-evaluation of plans,
  - Emphasis on face-to-face communication,
  - Relatively sparse use of documents.
- Agile teams, however, do follow defined and disciplined processes and carry out rigorous designs:
  - This is in contrast to chaotic coding in exploratory programming.

Extreme Programming (XP)

Extreme Programming Model

- Extreme programming (XP) was proposed by Kent Beck in 1999.
- The methodology got its name from the fact that:
  - **Recommends taking the best practices to extreme levels.**
  - If something is good, why not do it all the time.

Taking Good Practices to Extreme

- **If code review is good:**
  - Always review --- **pair programming**
- **If testing is good:**
  - Continually write and execute test cases --- **test-driven development**
- **If incremental development is good:**
  - Come up with new increments every few days
- **If simplicity is good:**
  - Create the simplest design that will support only the currently required functionality.

Taking to Extreme

- **If design is good,**
  - everybody will design daily (refactoring)
- **If architecture is important,**
  - everybody will work at defining and refining the architecture (metaphor)
- **If integration testing is important,**
  - build and integrate test several times a day (continuous integration)

4 Values

- **Communication**:
  - Enhance communication among team members and with the customers.
- **Simplicity**:
  - Build something simple that will work today rather than something that takes time and yet never used
  - May not pay attention for tomorrow
- **Feedback**:
  - System staying out of users is trouble waiting to happen
- **Courage**:
  - Don’t hesitate to discard code

Best Practices

- **Coding**:
  - without code it is not possible to have a working system.
  - Utmost attention needs to be placed on coding.
- **Testing**:
  - Testing is the primary means for developing a fault-free product.
- **Listening**:
  - Careful listening to the customers is essential to develop a good quality product.

Best Practices

- **Designing**:
  - Without proper design, a system implementation becomes too complex
  - The dependencies within the system become too numerous to comprehend.
- **Feedback**:
  - Feedback is important in learning customer requirements.

Extreme Programming Activities

- **XP Planning**
  - Begins with the creation of “user stories”
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
- **XP Design**
  - Follows the KIS principle
  - Encourage the use of CRC cards
  - For difficult design problems, suggests the creation of “spike solutions”—a design prototype
  - Encourages “refactoring”—an iterative refinement of the internal program design

Extreme Programming Activities

- **XP Coding**
  - Recommends the construction of unit test cases *before* coding commences (test-driven development)
  - Encourages “pair programming”
- **XP Testing**
  - All unit tests are executed daily
  - “Acceptance tests” are defined by the customer and executed to assess customer visible functionalities

Full List of XP Practices

1. **Planning** – determine scope of the next release by combining business priorities and technical estimates
2. **Small releases** – put a simple system into production, then release new versions in very short cycles
3. **Metaphor** – all development is guided by a simple shared story of how the whole system works
4. **Simple design** – system is to be designed as simple as possible
5. **Testing** – programmers continuously write and execute unit tests

Full List of XP Practices

7. **Refactoring** – programmers continuously restructure the system without changing its behavior to remove duplication and simplify

8. **Pair-programming** -- all production code is written with two programmers at one machine

9. **Collective ownership** – anyone can change any code anywhere in the system at any time.

10. **Continuous integration** – integrate and build the system many times a day – every time a task is completed.

Full List of XP Practices

11. **40-hour week** – work no more than 40 hours a week as a rule

12. **On-site customer** – a user is a part of the team and available full-time to answer questions

13. **Coding standards** – programmers write all code in accordance with rules emphasizing communication through the code

Emphasizes Test-Driven Development (TDD)

- Based on user story develop test cases
- Implement a quick and dirty feature every couple of days:
  - **Get customer feedback**
  - **Alter if necessary**
  - **Refactor**
- Take up next feature

Project Characteristics that Suggest Suitability of Extreme Programming

- Projects involving new technology or research projects.
  - In this case, the requirements change rapidly and unforeseen technical problems need to be resolved.
- Small projects:
  - These are easily developed using extreme programming.

Life Cycle Models: Scrum

Practice Questions

- What are the stages of iterative waterfall model?
- What are the disadvantages of the iterative waterfall model?
- Why has agile model become so popular?
- What difficulties might be faced if no life cycle model is followed for a certain large project?

Suggest Suitable Life Cycle Model

- A software for an academic institution to automate its:
  - Course registration and grading
  - Fee collection
  - Staff salary
  - Purchase and store inventory
- The software would be developed by tailoring a similar software that was developed for another educational institution:
  - 70% reuse
  - 10% new code and 20% modification

Practice Questions

- Which types of risks can be better handled using the spiral model compared to the prototyping model?
- Which type of process model is suitable for the following projects:
  - A customization software
  - A payroll software for contract employees that would be add on to an existing payroll software

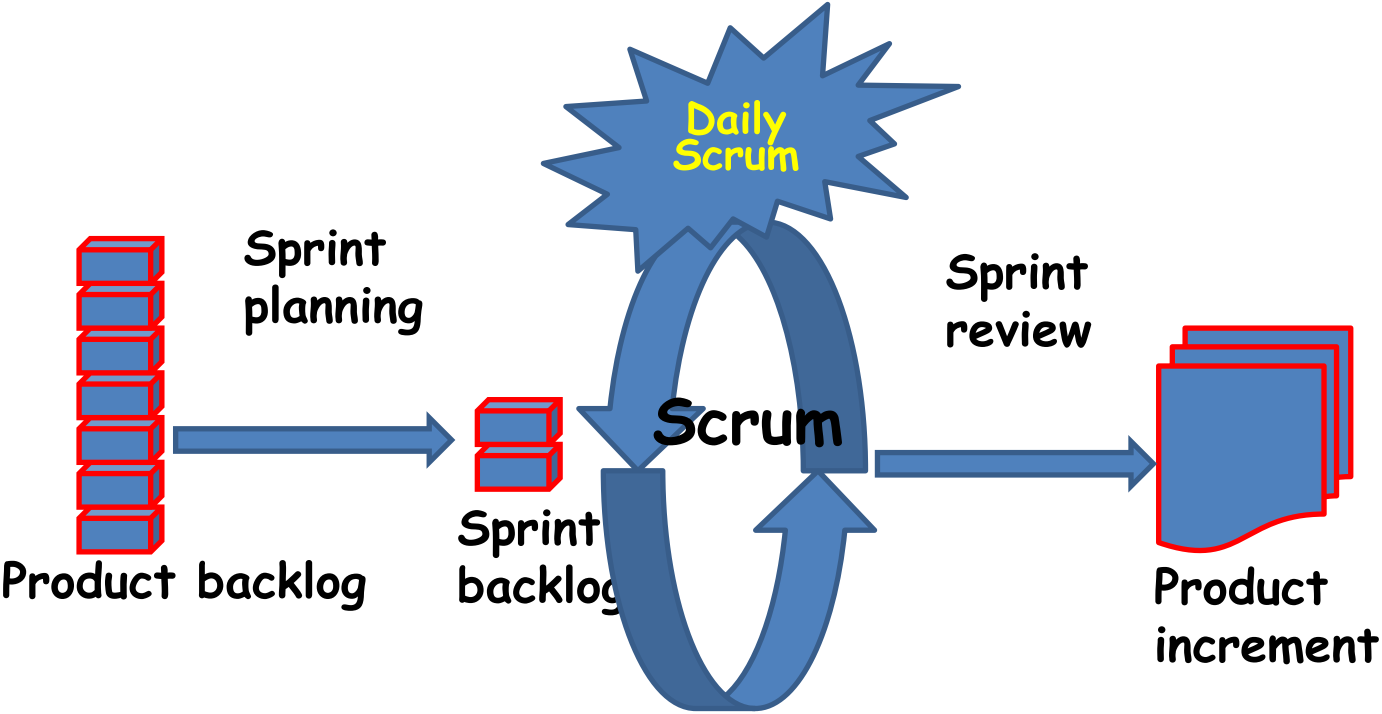
Practice Questions

- Which lifecycle model would you select for the following project which has been awarded to us by a mobile phone vendor:
  - A new mobile operating system by upgrading the existing operating system
  - Needs to work well efficiently with 4G systems
  - Power usage minimization
  - Directly upload backup data on a cloud infrastructure maintained by the mobile phone vendor

Scrum

Scrum: Characteristics

- Self-organizing teams
- Product progresses in a series of month-long **sprints**
- Requirements are captured as items in a list of **product backlog**
- One of the agile processes



Product backlog

Sprint planning

Sprint backlog

Daily Scrum

Scrum

Sprint review

Product increment

Sprint

- Scrum projects progress in a series of “sprints”
  - Analogous to XP iterations or time boxes
  - Target duration is one month
- **Software increment is designed, coded, and tested during the sprint**
- **No changes entertained during a sprint**

Scrum Framework

- **Roles :** Product Owner, Scrum Master, Team
- **Ceremonies :** Sprint Planning, Sprint Review, Sprint Retrospective, and Daily Scrum Meeting
- **Artifacts :** Product Backlog, Sprint Backlog, and Burndown Chart

Key Roles and Responsibilities in a Scrum Team

- **Product Owner**
  - Represents customers' views and interests.
- **Development Team**
  - Team of five-nine people with cross-functional skill sets.
- **Scrum Master (aka Project Manager)**
  - Facilitates scrum process and resolves impediments at the team and organization level by acting as a buffer between the team and outside interference.

Product Owner

- Defines the features of the product
- Decides on release date and content
- Prioritizes features according to usefullness
- Adjusts features and priority every iteration, as needed
- Accepts or reject work results.

The Scrum Master

- Represents management in the project
- Removes impediments
- Ensures that the team is fully functional and productive
- Enables close cooperation across all roles and functions
- Shields the team from external interferences

Scrum Team

- Typically 5-10 people
- Cross-functional
  - QA, Programmers, UI Designers, etc.
- Teams are self-organizing
- Membership can change only between sprints

Sprint

- Fundamental process flow of Scrum
- It is usually a month-long iteration:
  - during this time an incremental product functionality completed
- NO outside influence allowed to interfere with the Scrum team during the Sprint
- Each day begins with the Daily Scrum Meeting

Ceremonies

- Sprint Planning Meeting
- Daily Scrum
- Sprint Review Meeting

Sprint Planning

- Goal is to produce Sprint Backlog
- Product owner works with the Team to negotiate what Backlog Items
- Scrum Master ensures Team agrees to realistic goals

Daily Scrum

- Daily
- 15-minutes
- Stand-up meeting
- Not for problem solving
- Three questions:
  1. What did you do yesterday
  2. What will you do today?
  3. What obstacles are in your way?

Daily Scrum

- Is NOT a problem solving session
- Is NOT a way to collect information about WHO is behind the schedule
- Is a meeting in which team members review what is done and make informal commitments to each other and to the Scrum Master
- Is a good way for a Scrum Master to track the progress of the Team

Sprint Review Meeting

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features
- Informal
  - 2-hour prep time rule
- Participants
  - Customers
  - Management
  - Product Owner
  - Other team members

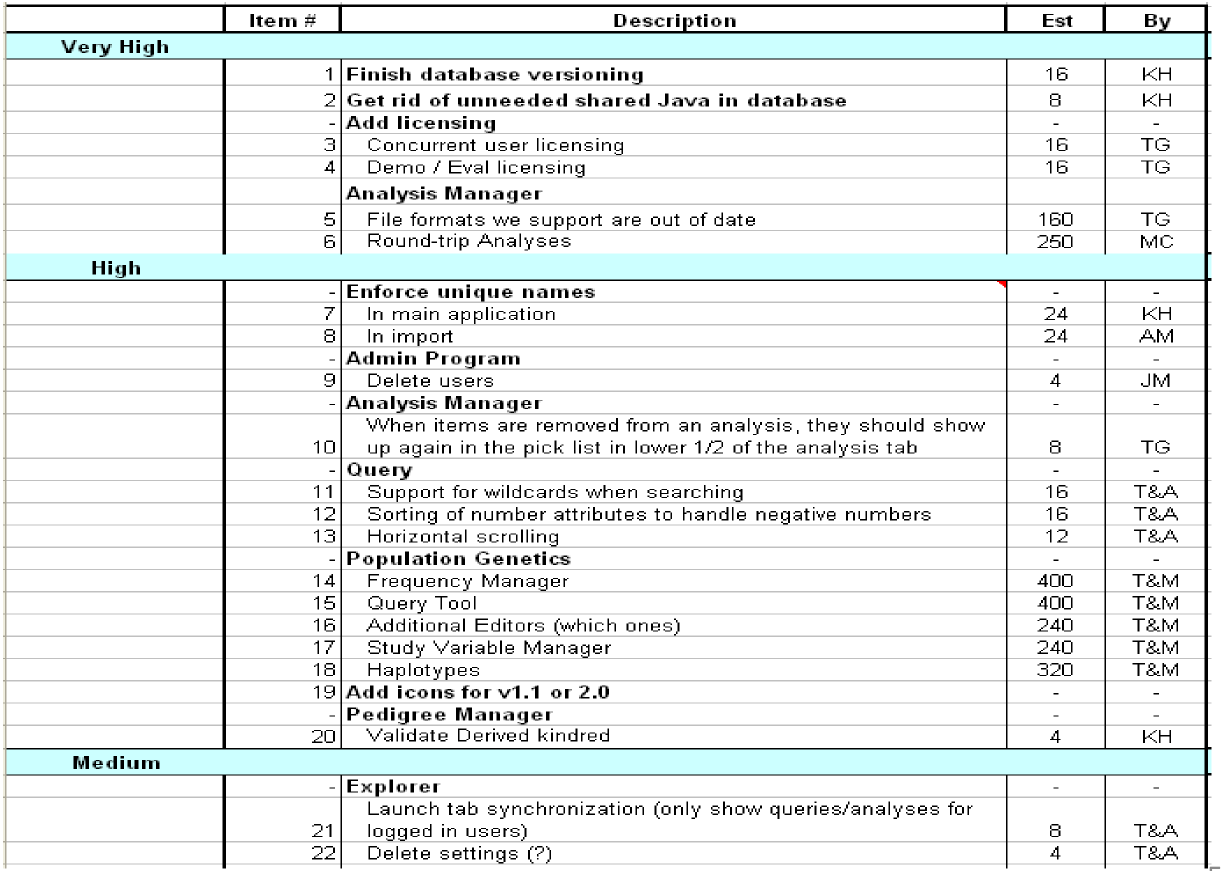
Product Backlog

- A list of all desired work on the project
  - Usually a combination of
    - story-based work (“let user search and replace”)
    - task-based work (“improve exception handling”)
- List is prioritized by the Product Owner
  - Typically a Product Manager, Marketing, Internal Customer, etc.

Product Backlog

- Requirements for a system, expressed as a prioritized list of Backlog Items
  - **Managed and owned by Product Owner**
  - **Spreadsheet (typically)**

Sample Product Backlog



	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	**Admin Program**	-	-
	9	Delete users	4	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	4	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

Sprint Backlog

- A subset of Product Backlog Items, which define the work for a Sprint
  - **Created by Team members**
  - **Each Item has it’s own status**
  - **Updated daily**

Sprint Backlog during the Sprint

- Changes occur:
  - Team adds new tasks whenever they need to in order to meet the Sprint Goal
  - Team can remove unnecessary tasks
  - But: Sprint Backlog can only be updated by the team
- Estimates are updated whenever there's new information

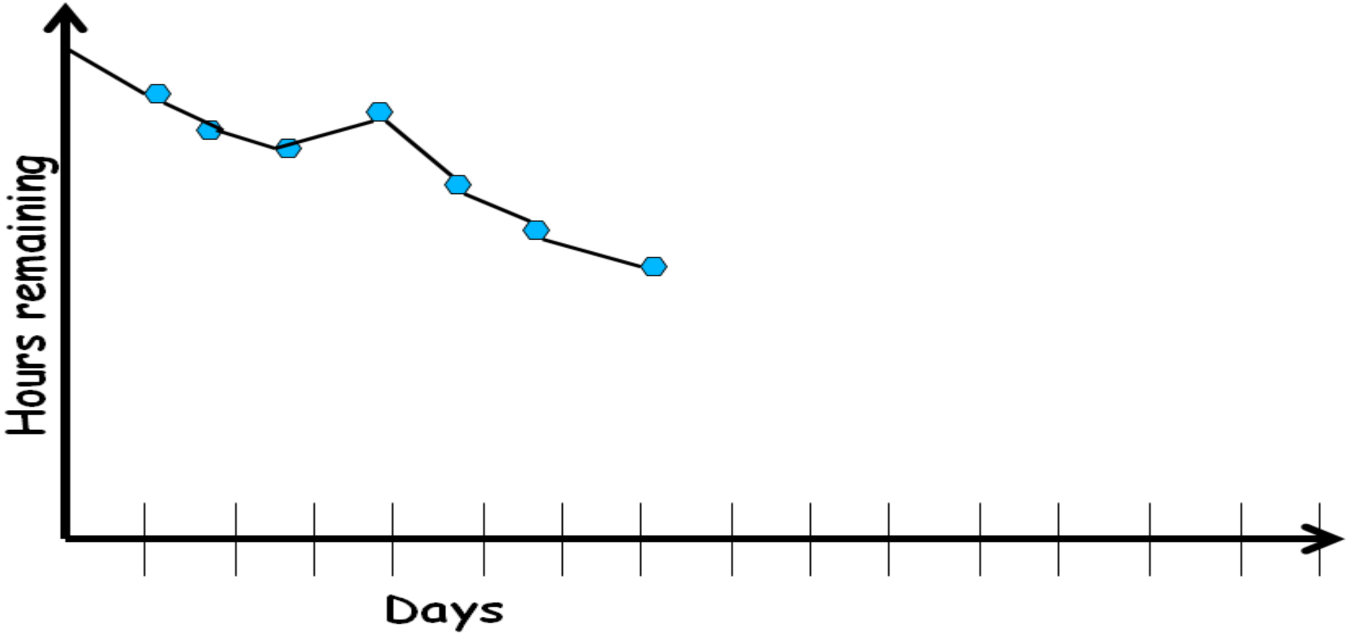
Burn down Charts

- Are used to represent “work done”.
- Are remarkably simple but effective Information disseminators
- 3 Types:
  - **Sprint Burn down Chart (progress of the Sprint)**
  - **Release Burn down Chart (progress of release)**
  - **Product Burn down chart (progress of the Product)**

Sprint Burn down Chart

- Depicts the total Sprint Backlog hours remaining per day
- Shows the estimated amount of time to complete
- Ideally should burn down to zero to the end of the Sprint
- Actually is not a straight line

Sprint Burndown Chart



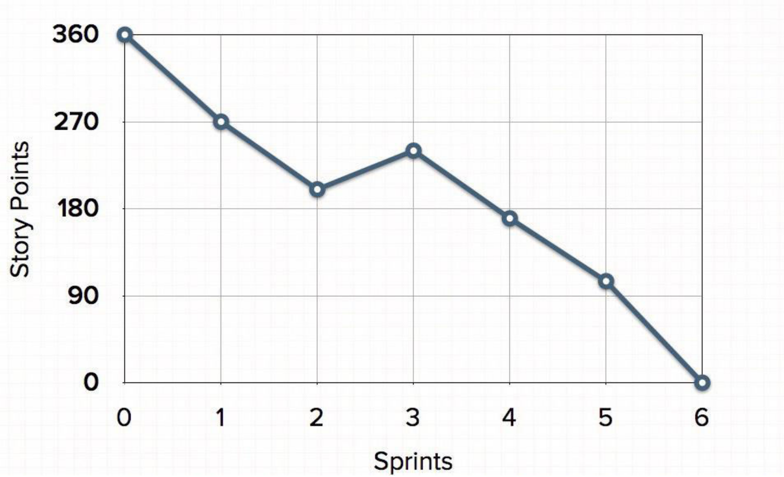
Release Burndown Chart

- Will the release be done on right time?

- How many more sprints?

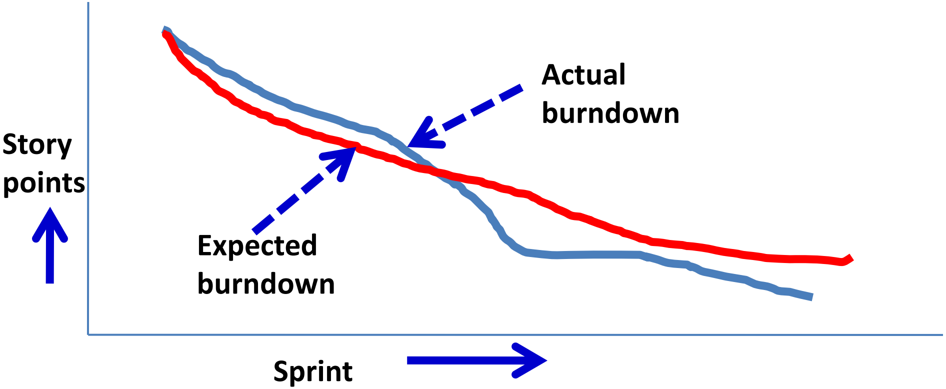
- X-axis: sprints

- Y-axis: amount of story points remaining



Product Burndown Chart

- Is a “big picture” view of project’s progress (all the releases)



Scalability of Scrum

- A typical Scrum team is 6-10 people
- Jeff Sutherland - up to over 800 people
- "Scrum of Scrums" or "Meta-Scrum“

Thank You!!