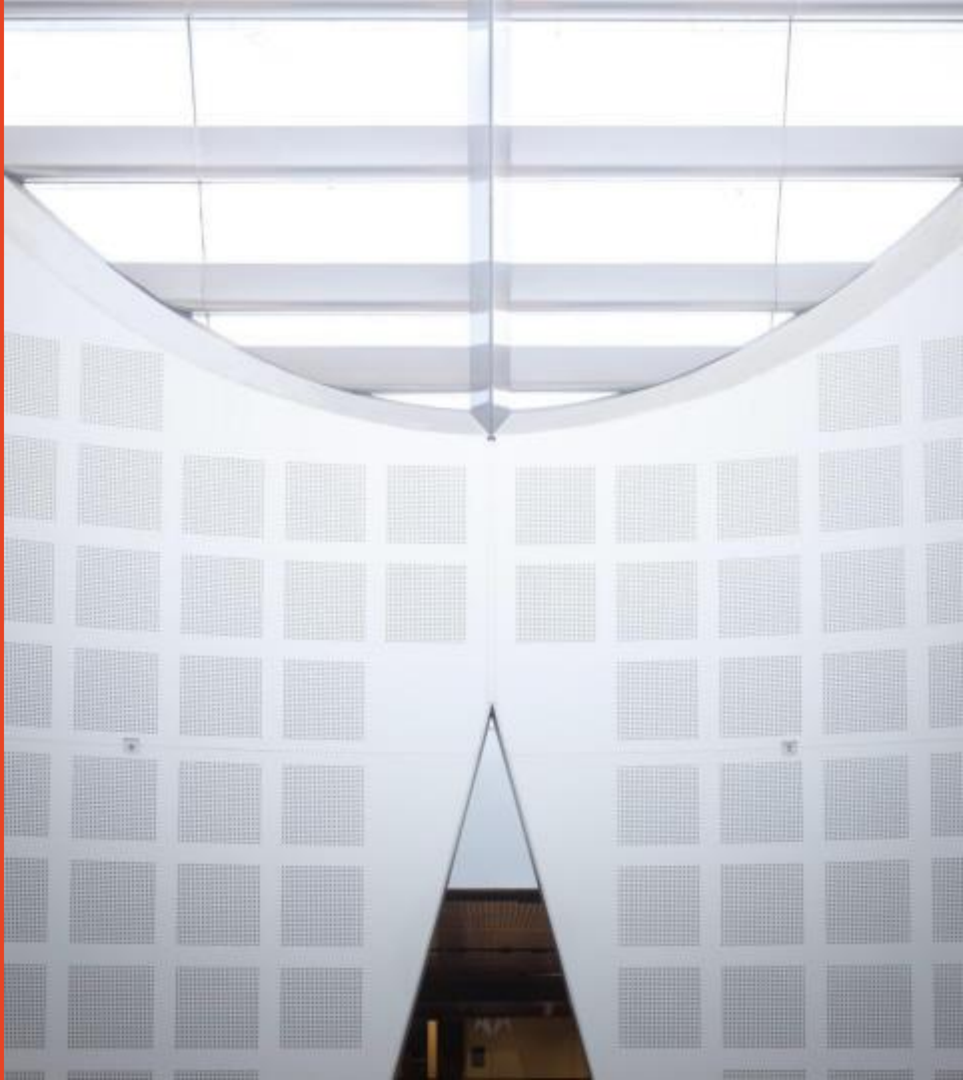
Agile Software Development Practices (SOFT2412/COMP9412)

Estimation and Its Challenges; Tools and Technologies for Tracking Progress

Xinyi Sheng

School of Computer Science

Slides based on Dr. Basem Suleiman and Dr. Farshid Hajati



Agenda

- Planning in Traditional Software Development
- Planning in Agile Development

Part 1

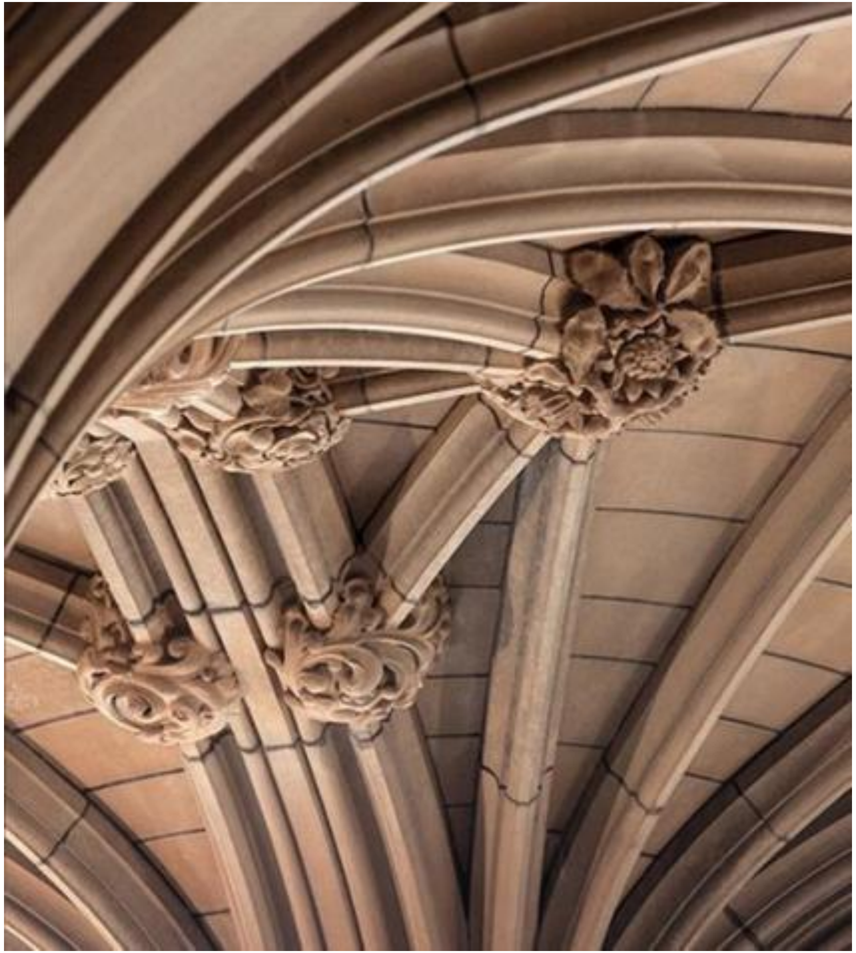
- Estimation in Agile Development
  - ① Story Points, ② Ideal/Elapsed Time
  - ③ Velocity

Part 2

- Tracking Project Progress
  - Burndown charts, Velocity Charts
- Tools for tracking progress
  - JIRA Agile

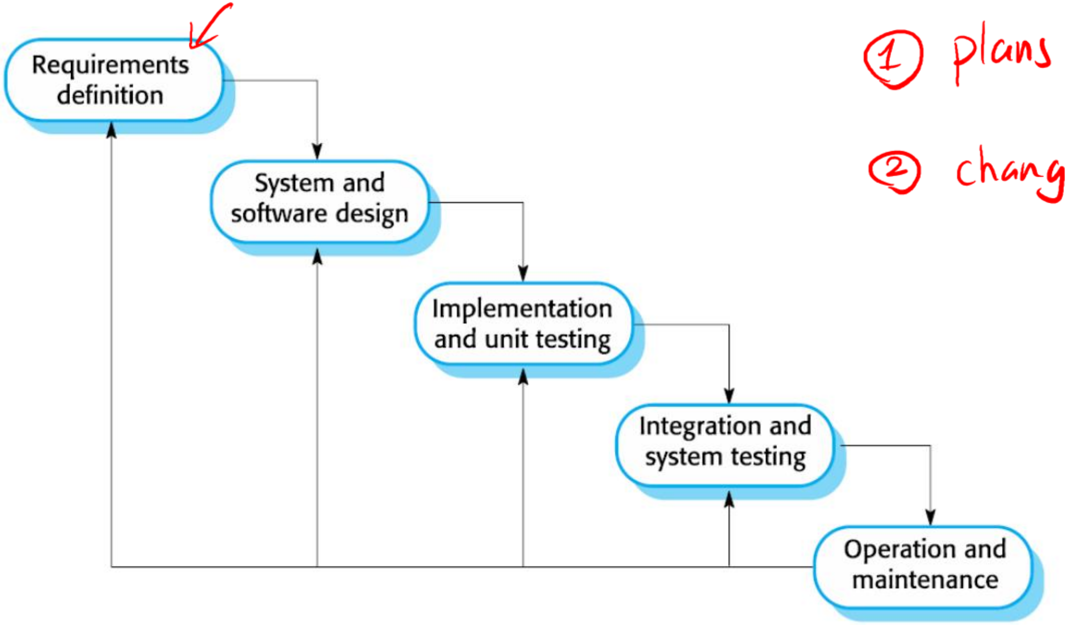
Part 3

Plan-and-document Software Development



Plan-and-Document Software Methodologies

– Typical phases in traditional software development methodologies


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

① plans are made in advance

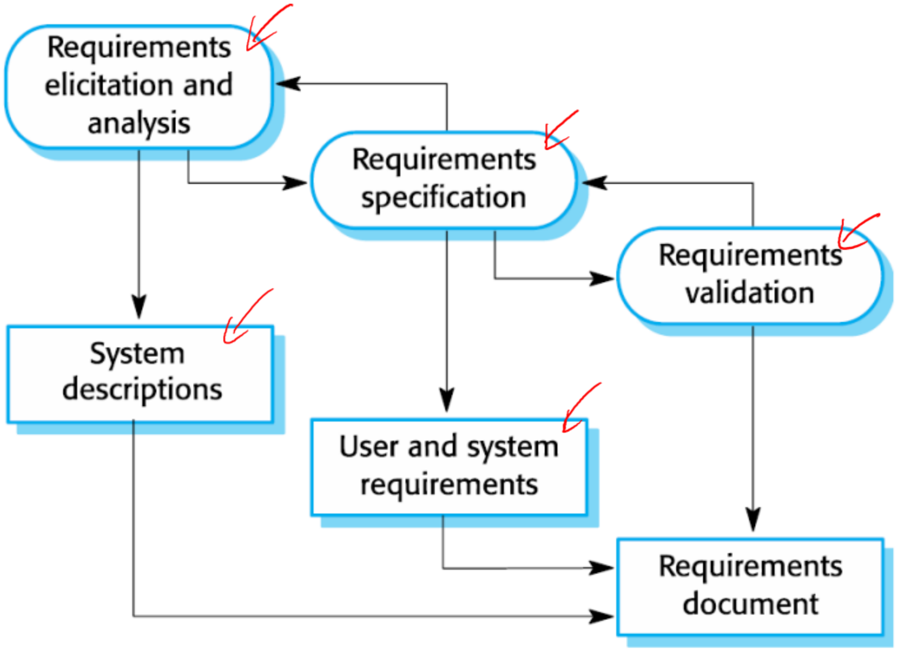
② changes are very expensive

Ian Sommerville. 2016. Software Engineering (10th ed. Global Edition). Pearson

Plan-and-Document Software Methodologies

- Goal is to make Software Engineering predicable in budget and schedule
  - Requirements elicitation
  - Requirements documentation
  - Cost estimation
  - Scheduling and monitoring schedule
  - Change management for requirements, cost and schedule
  - Ensuring implementation matches requirement features
  - Risk analysis and management

Plan-and-Document - Requirements Engineering Process


Requirements elicitation and analysis ✓
Requirements specification ✓
Requirements validation ✓
System descriptions ✓
User and system requirements ✓
Requirements document

Ian Sommerville. 2016. Software Engineering (10th ed. Global Edition). Pearson

Requirements Documentation

- Software Requirements Specifications (SRS) process
  - 100s of pages, IEEE 830-1998 standard recommended practice for SRS

- Stakeholders to read SRS document, build basic prototype, or generate test cases to check:
  - Validity: are all requirements necessary?
  - Consistency: do requirements conflict?
  - Completeness: are all requirements and constraints included?
  - Feasibility: can requirements be implemented?

- Estimate budget and schedule based on the SRS

Why Software Projects Fail?

- **Over-budget, over-time**
- Hard to maintain and evolve
- Useless (unwanted) product features
  - Standish group’s CHAOS report:
    - 45% of features never used, 19% rarely used
  - Development teams would build software, and *throw it over the wall* to their users, and hope some of what they build would stick

https://www.projectsmart.co.uk/white-papers/chaos-report.pdf

Software Failures - Budget, Schedule, Requirements

Project	Duration	Cost	Failure/Status
e-borders (UK Advanced passenger Information System Programme)	**2007 - 2014**	Over £ 412m (expected), £742m (actual)	Permanent failure - **cancelled after a series of delays**
Pust Siebel - Swedish Police case management (Swedish Police)	**2011 - 2014**	$53m (actual)	Permanent failure – scraped due to poor functioning, inefficient in work environments
US Federal Government Health Care Exchange Web application	**2013 - ongoing**	$93.7m (expected), $1.5bn (actual)	Ongoing problems - too slow, poor performance, people get stuck in the application process (frustrated users)
Australian Taxation Office's Standard Business Reporting	**2010 - ongoing**	~$1bn (to-date), ongoing	Significant spending on contracting fees (IBM & Fjitsu), significant scope creep and confused objectives

https://en.wikipedia.org/wiki/List_of_failed_and_overbudget_custom_software_projects

Planning in Agile Software Development

Agile Manifesto - Planning

- "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**
- The items on the left are more valued than those at the right

Agile Manifesto: http://agilemanifesto.org/

Agile Principles - Planning and Estimation?

Which Principles relate to planning and estimation in Agile Software Development ?

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 Alliance: http://www.agilealliance.org

Agile Principles - Planning

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. Businesspeople 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.**

Agile Alliance: http://www.agilealliance.org

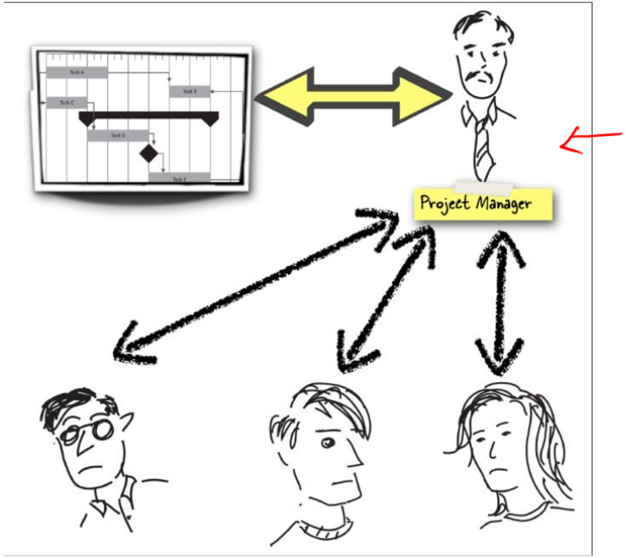
Planning in Software Development

- **Plan-driven (plan-and-document or heavy-weight)**
  - All of the process activities are planned in advance and progress is measured against this plan
  - Plan drives everything and change is expensive

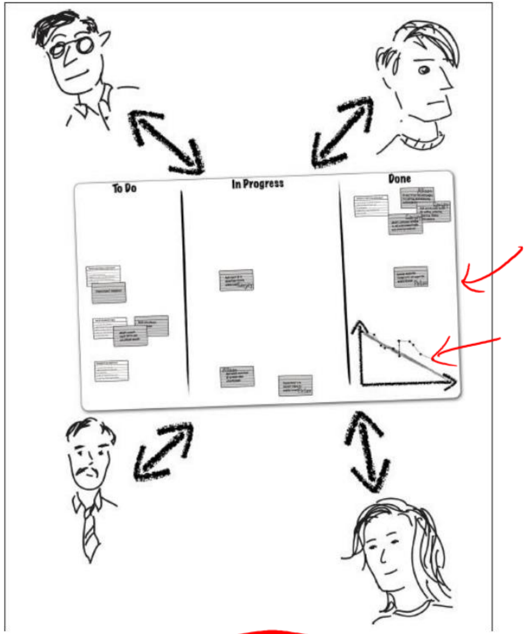
- **Agile processes (light-weight)**
  - Planning is *incremental and continual* as the software is developed
  - Easier to change to reflect changing requirements

Scrum - Team Structure

Which team structure better reflect planning and estimation in Agile development?

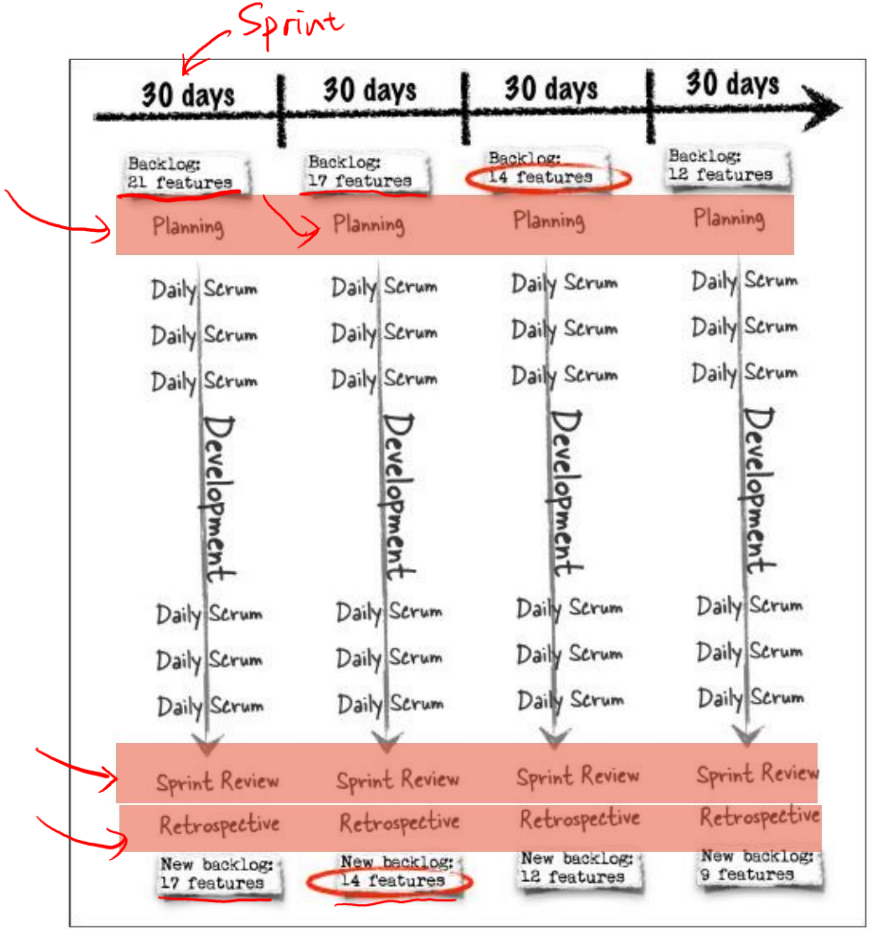


Team A



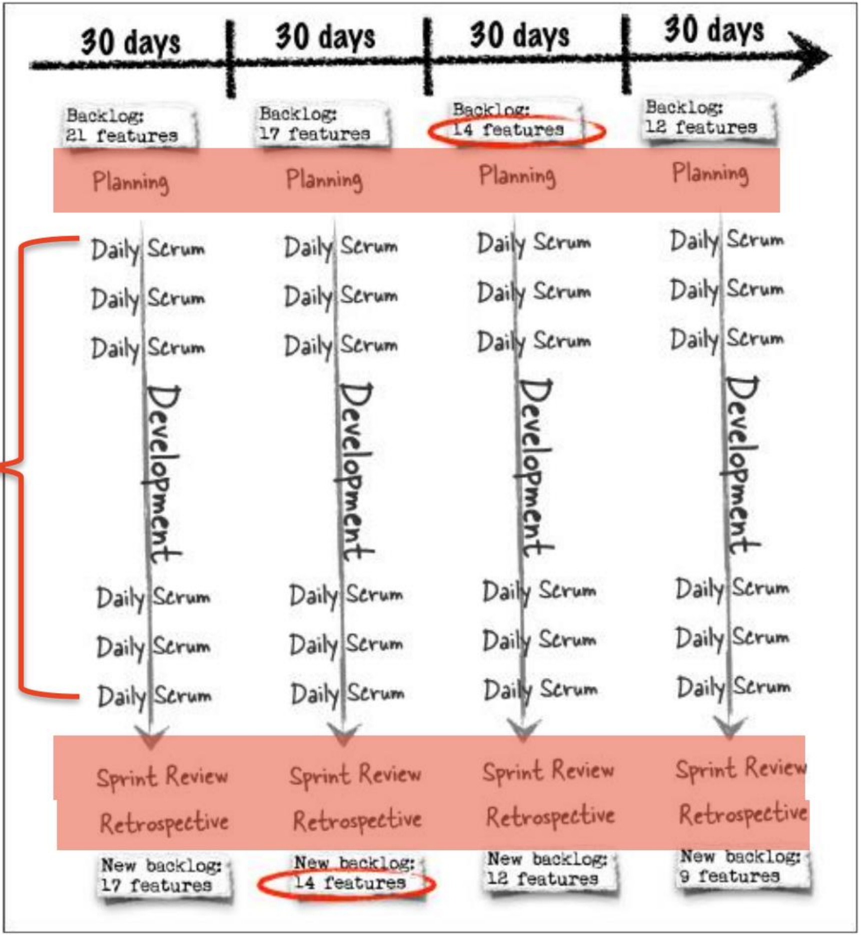
Team B

Scrum - Planning



Scrum - Planning

Can planning be changed (e.g., adding/removing user stories to a Sprint) during development?



Agile Methods for Estimating Size

①Story Points and ②ideal Days

Estimating Size in Agile Development

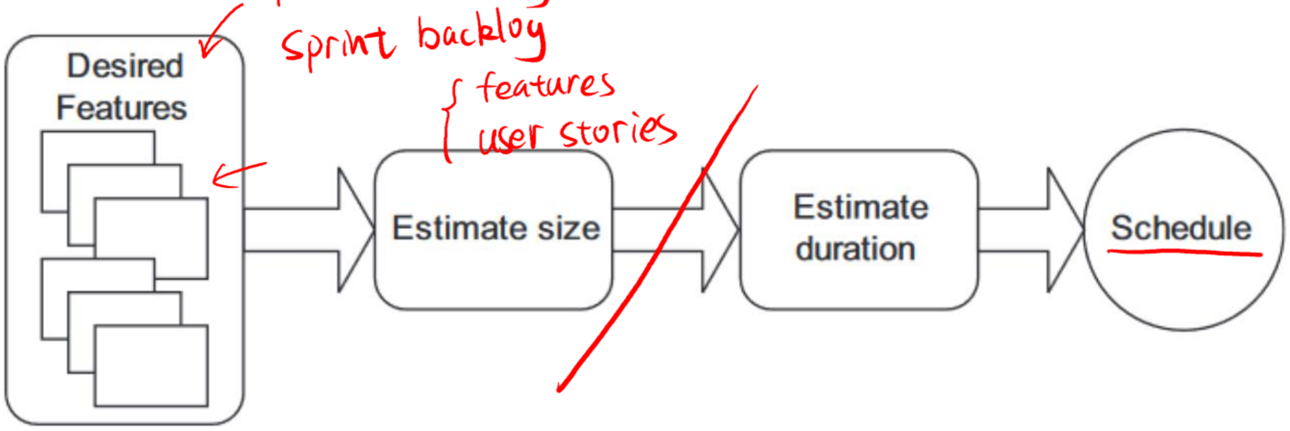
– Agile teams separate estimates of size from estimates of duration

product backlog

sprint backlog

features

user stories



Estimating the duration of a project begins with estimating its size.

Requirements in Agile Software Development

- Requirements are also *crucial for planning* in Agile development
- Work continuously and closely with stakeholders to develop requirements and tests
- Iterative development

  Sprints in Scrum

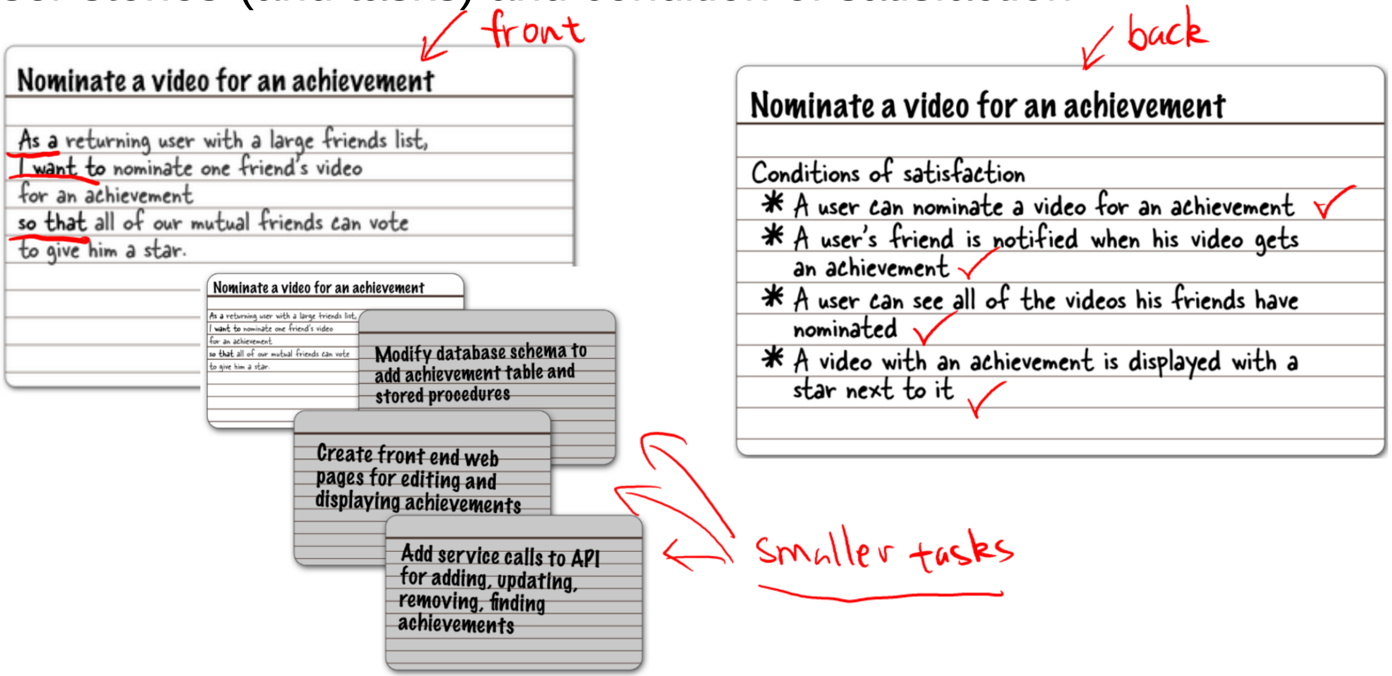
  - Short iterations 2-4 weeks each focused on core features
  - Maintain working prototype while adding new features

    Scrum: Sprint Review

  - Check with stakeholders what's next to validate building the right software
  - Initial planning and estimation and adapt during development

User Stories - Base for Better Estimation and Monitoring

– User stories (and tasks) and condition of satisfaction



front

Nominate a video for an achievement

As a returning user with a large friends list,
I want to nominate one friend's video
for an achievement
so that all of our mutual friends can vote
to give him a star.

Nominate a video for an achievement

As a returning user with a large friends list,
I want to nominate one friend's video
for an achievement
so that all of our mutual friends can vote
to give him a star.

Modify database schema to add achievement table and stored procedures

Create front end web pages for editing and displaying achievements

Add service calls to API for adding, updating, removing, finding achievements

smaller tasks

back

Nominate a video for an achievement

Conditions of satisfaction
* A user can nominate a video for an achievement ✓
* A user's friend is notified when his video gets an achievement ✓
* A user can see all of the videos his friends have nominated ✓
* A video with an achievement is displayed with a star next to it ✓

Estimation - Story Points

- Metric for the overall size of a user story, feature, or other piece of work
- A point value to each item is assigned
- No. of **story points** ~ the **overall size** of the story
- Estimation scale:
  - 1 *Fibonacci series*; 1, 2, 3, 5, 8, … ✓
  - 2 *Subsequent number as twice the number that precedes it*: 1, 2, 4, 8, …
- Why?
  - Measure of the user story only
  - No emotional measurements
  - Team velocity is considered
  - Team members focus on solving problems based on difficulty not time spent

Estimation - Staring Story Points

- Two approaches:
  - ① Smallest story (expected) is estimated at 1 story point
  - ② Medium-sized story assigned a number somewhere in the middle of the range you expect to use

story points range: 1, 2, 3, 5, 8

- Estimating in story points completely separates the estimation of effort from the estimation of duration

Sprint Planning Session using Story Points

- Start with the most valuable user stories from the product backlog

- Take a story in that list (ideally the smallest one)

- Discuss with the team whether that estimate is accurate

- Keep going through the stories until the team have accumulated enough points to fill the Sprint

3 sprint 30 SPs

30/3 SPs for 1 Sprint

Why Story Points Work?

- Simple, not magic
- The team is in control of them
- They get your team talking about estimates
- Developers are not scared of them
- They help the team discover exactly what a story means
- They help everyone on the team become genuinely committed

Ideal Days vs Elapsed Days

Ideal time   elapsed time

- **Ideal time** and **elapsed time** are different
  - American football game is 60 minutes; however, 3 or more hours will typically pass on a clock before a 60 minutes game is finished

- Time to do a development task without any interruptions is?

Ideal days

- Time to do a development task with work interruptions is?

Elapsed days

Estimating in Ideal Days

- Ideal days
  - The amount of time a user story will take to develop

- Elapsed days
  - Requires considering all interruptions that might occur while working on a user story

- Ideal days only is an estimate of size

Estimating in Ideal Days

- Associate a single estimate with each user story
  - Not in four programmer days, two tester days, and three product owner days, etc.
- Express the estimate as a whole (i.e., nine ideal days)

Estimation Techniques

- Expert opinion (estimates based on opinion)
- Analogy: compare with other stories
- Disaggregation: splitting a story into smaller tasks
- Planning Poker: based on expert opinion, analogy, disaggregation and fun

Estimation Technique - Planning Poker

- Gamified technique to estimate effort/relative size of development in Agile development

- The best way I've found for agile teams to estimate is by playing planning poker (Grenning 2002)

Agile Estimating and Planning: Planning Poker - Mike Cohn

Planning Poker

- Aims to avoid individual influence/bias (think independently)

- Team discussion with mediation to consider different views

- Team involvement in planning increases commitment

- It combines expert opinion, analogy, and disaggregation into an enjoyable approach to estimating that results in quick but reliable estimates

Considering Story Points over Ideal Days

- Story points help drive cross-functional behavior
- Story points are a pure measure of size
- Estimating in story points is typically faster
- My ideal days are not your ideal days

Re-Estimating

- Story points and ideal days helps you know when to re-estimate

- Re-estimate only when your opinion of the relative size of one or more stories has changed

- Do not re-estimate solely because progress is not coming as rapidly as you'd expected

- Let velocity take care of most estimation inaccuracies

Considering Ideal Days over Story Points

- Ideal days are easier to explain outside the team
- Ideal days are easier to estimate at first
- Ideal days make velocity predictions easier

Estimating Progress

Velocity

Estimating Progress - Velocity

- A measure of a team's rate of progress

- Sum the number of story points assigned to each user story that the team completed during the iteration

- Example:
  - A team completed 3 stories, 5 SPs each ➔ velocity = 15

5 + 5 + 5

Estimation - of Number of Sprints

- Size estimate of the project: sum the SP estimates for all desired features

- Divide size by velocity to arrive at an estimated number of Sprints (iterations)

Velocity, Duration and Schedule - Exercise

- Sum of all user stories is 100 SPs

- Team's velocity, based on past experience, is 11 SP per two- week Sprint

- Work out the following estimates:
  - How many iterations/Sprints ?
  - Estimate of duration is?
  - Project schedule?

Velocity, Duration and Schedule - Example

$\frac{100}{11} \approx 9.1$

- 100 SPs
- Team's velocity 11 SP per two-week iteration
- 9.1 iterations (either 10 iterations or find one point to remove)
  - Let's assume we go with10 iterations

$2 \times 10 = 20$

- 1 iteration = 2 weeks ➔ estimate of duration is 20 weeks
- Count forward 20 weeks on the calendar and that becomes our schedule

Estimating Velocity

- Use Historical values
- Run an Sprint (iteration)
- Make a forecast

Consideration for Historical Values

- Is the technology the same?
- Is the domain the same?
- Is the team the same?
- Is the product owner the same?
- Are the tools the same?
- Is the working environment the same?
- Were the estimates made by the same people

Run a Sprint

- Run an iteration (or two or three) and then estimate velocity from the observed velocity during the one to three iterations

Velocity - Forecasting

- Estimate the number of hours that each person will be available to work on the project each day.
- Determine the total number of hours that will be spent on the project during the iteration.
- Somewhat randomly select stories and expand them into their basic tasks.
- Repeat until you have identified enough tasks to fill the number of hours in the iteration.
- Convert the velocity determined in the prior step into a range.

Velocity - Planning and Estimation

- Inconsistent velocity over long time
  - Hidden challenges not counted?
  - Outside business/stakeholders' pressure?

- Observe team velocity throughout the Sprints and investigate decrease in average velocity (e.g., inefficiency)
  - Discuss in the retrospective meetings

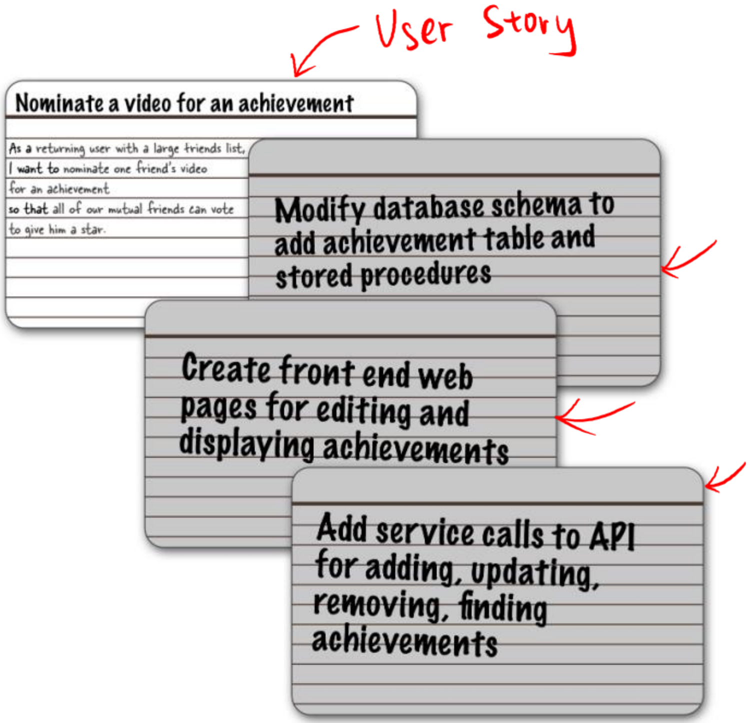
Tracking Project Progress

Burndown Chart, Task Board

Planning and Running a Sprint Using Stories, Points, Tasks, and a Task Board - Revisit

- First half of the Sprint planning
  - Story points and velocity to figure out what will go into the Sprint
- Second half of the Sprint planning
  - Plan out the actual work for the team is to add cards for individual tasks
  - Tasks can be written code, create design and architecture, and all of those other things that teams actually do every day to build and release software

Tasks - Sprint Planning (Revisit)



Nominate a video for an achievement

As a returning user with a large friends list,
I want to nominate one friend's video
for an achievement
so that all of our mutual friends can vote
to give him a star.

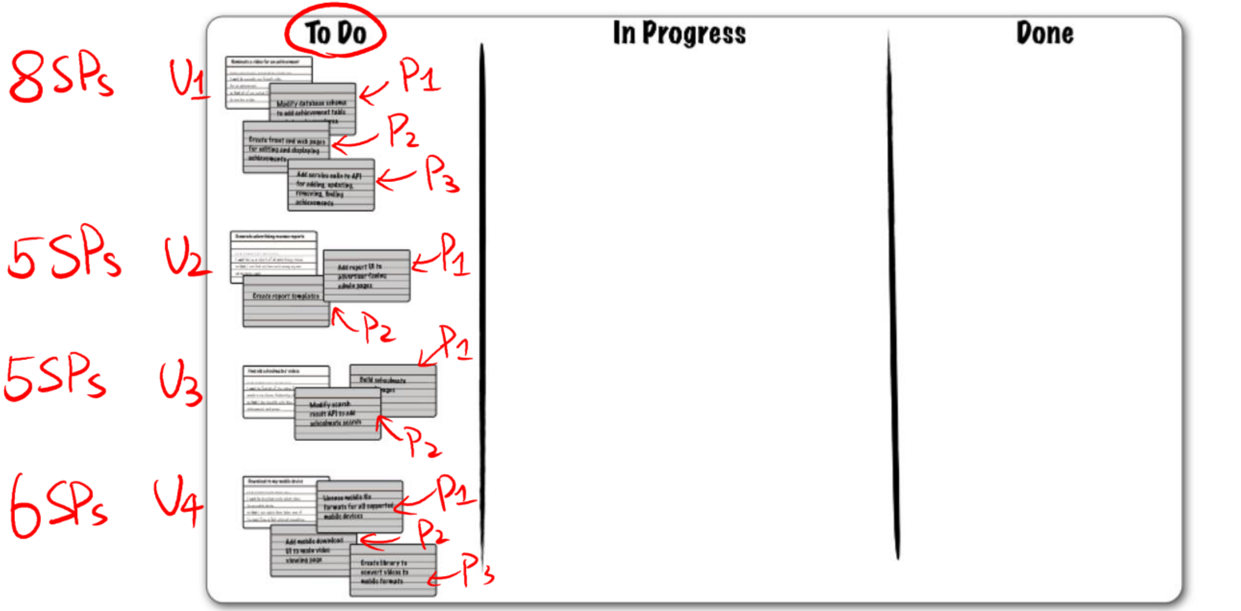
Modify database schema to add achievement table and stored procedures

Create front end web pages for editing and displaying achievements

Add service calls to API for adding, updating, removing, finding achievements

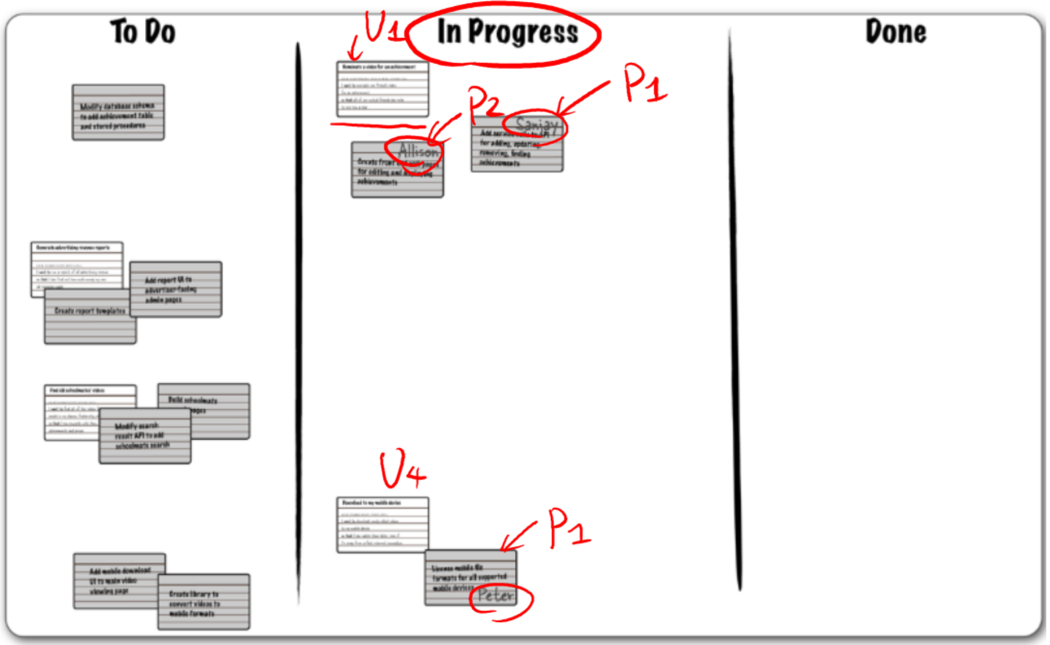
The second half of the sprint planning session is all about breaking the stories down into tasks that the team members will do during the sprint

Task Board (1) - Revisit


To Do
In Progress
Done
8SPs
U1
Nominate a video for an achievement
Modify database schema to add achievement table
Create front end web pages for editing and displaying achievements
Add service calls to API for adding, updating, removing, finding achievements
P1
P2
P3
5SPs
U2
Generate advertising revenue reports
Add report UI to advertiser-facing admin pages
Create report templates
P1
P2
5SPs
U3
Find old schoolmates' videos
Build schoolmate pages
Modify search result API to add schoolmate search
P1
P2
6SPs
U4
Download to my mobile device
Choose mobile file formats for all supported mobile devices
Add mobile download UI to web video viewing page
Create library to convert videos to mobile formats
P1
P2
P3

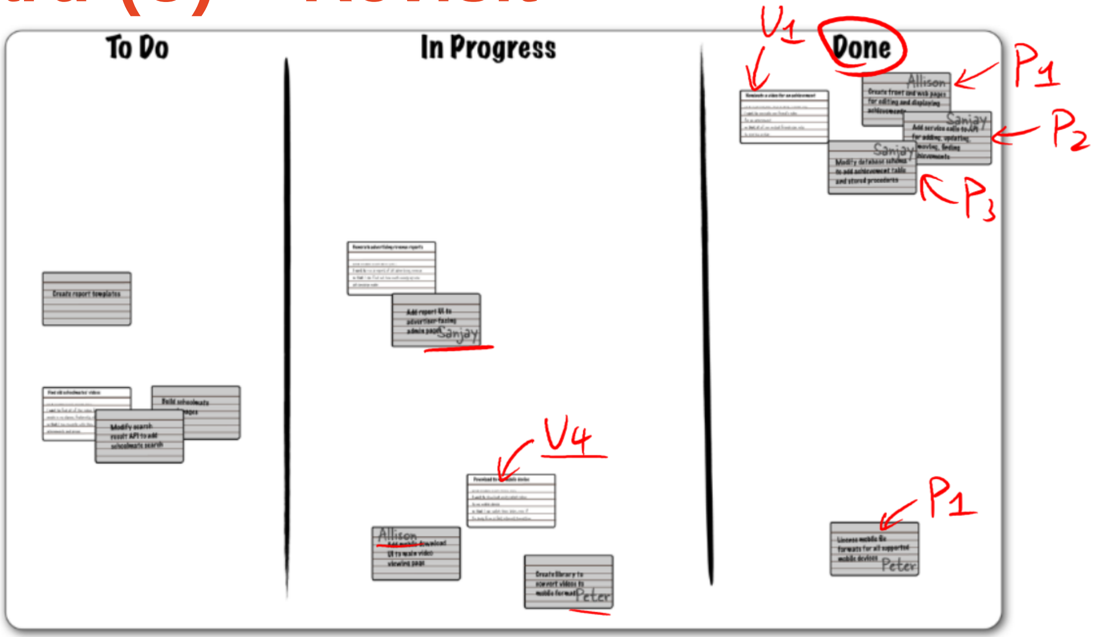
Each story card in a sprint is grouped together with the cards for its tasks and added to the "To Do" section of the task board.

Task Board (2) - Revisit


To Do
In Progress
Done
U1
P2
P1
Sanjay
Allison
U4
P1
Peter
Modify database schema to add achievement table and stored procedures
Create report templates
Add report UI to advertiser-facing admin pages
Build schoolmate pages
Modify search result API to add schoolmate search
Add mobile download UI to main video viewing page
Create library to convert videos to mobile formats
Create front end pages for editing and displaying achievements
Add services for adding, updating, removing, finding achievements
License mobile file formats for all supported mobile devices

Each team member works on exactly one task at a time by writing his or her name on its card and moving it to the "In Progress" section of the task board.

Task Board (3) - Revisit



8 SPs

When a team member finishes a task, they move it to the "Done" section and claims another task—and if all of the story's conditions of satisfaction are met, the story card is moved to the "Done" section too

Burndown Charts

- Tracks the completion of development work throughout the Sprint;

- Should be visible to everyone in the team (e.g., whiteboard, wall chart, online tool)

- First half of the Sprint planning
  - Story points and velocity to figure out what will go into the Sprint

- Good estimation and planning should help the team to burn stories relatively with similar pace

Burndown Charts based on Story Points

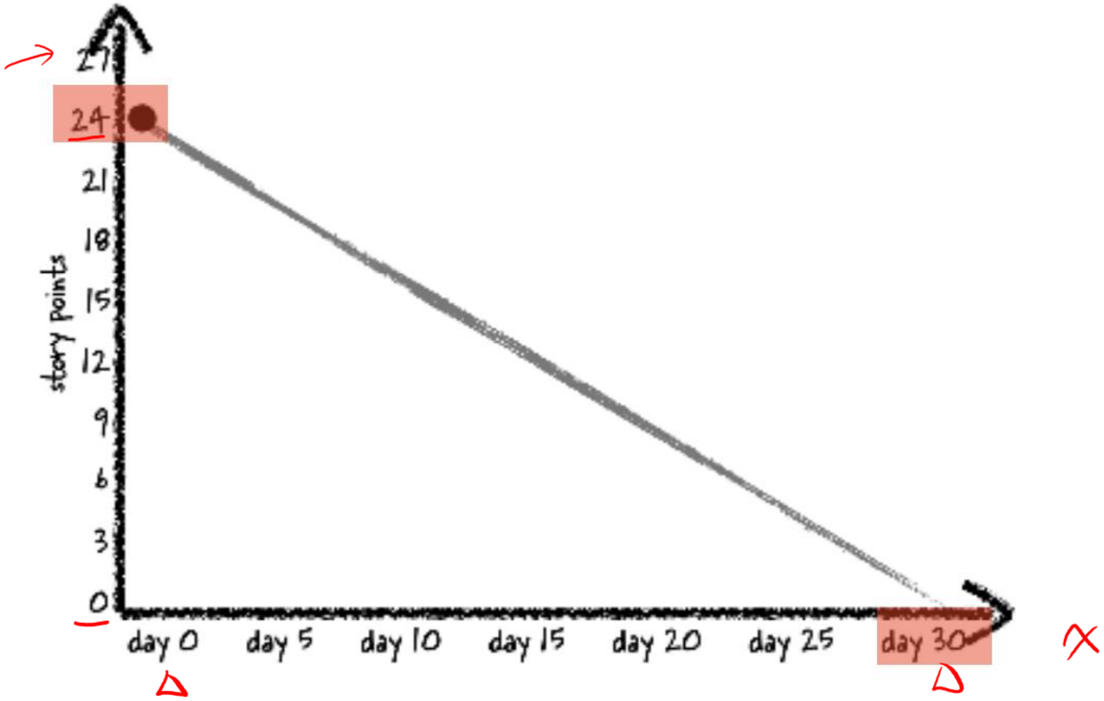
- x-axis: first and end dates of the sprint
- y-axis: story points (0 to 20% more than the total no. of points in the Sprint)

24 SPs $24 + 24 \cdot 20\%$

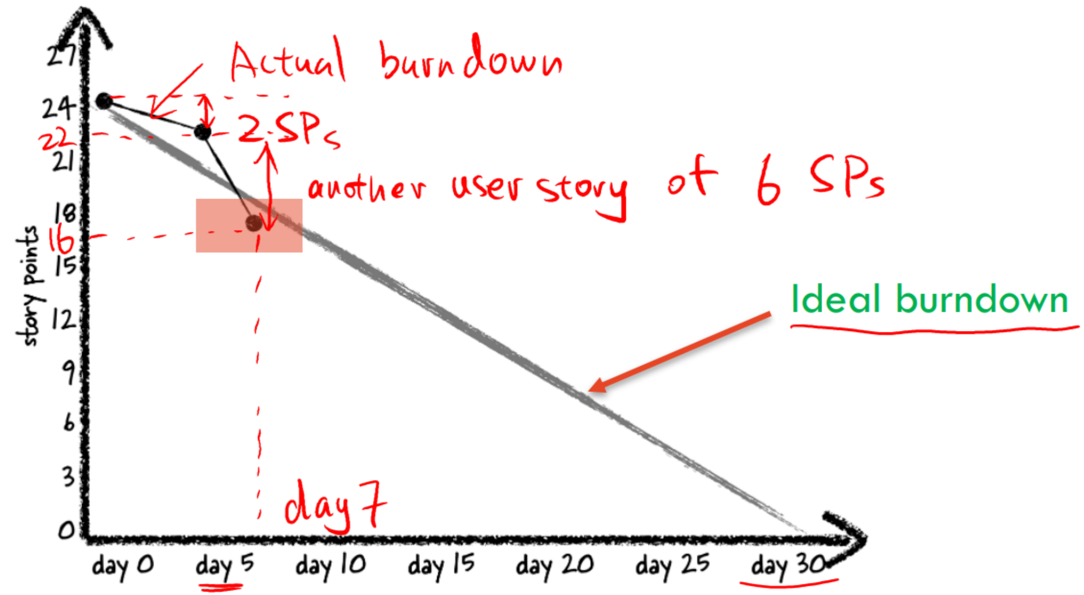
- The plot: a user story is “Done”, plot the number of points left in the sprint, at the current day
  - Fill in more days in the chart as more stories are finished
- More work needs to be added (discovered during daily scrum)
  - Estimate amount of points to remove to balance out the Sprint
  - Add card(s) to the task board, follow-up team meeting to estimate added work
  - Add notes to the chart
- As you get close to the end of the Sprint, more points burn off the chart

Burndown Charts

- Burndown chart when the Sprint starts

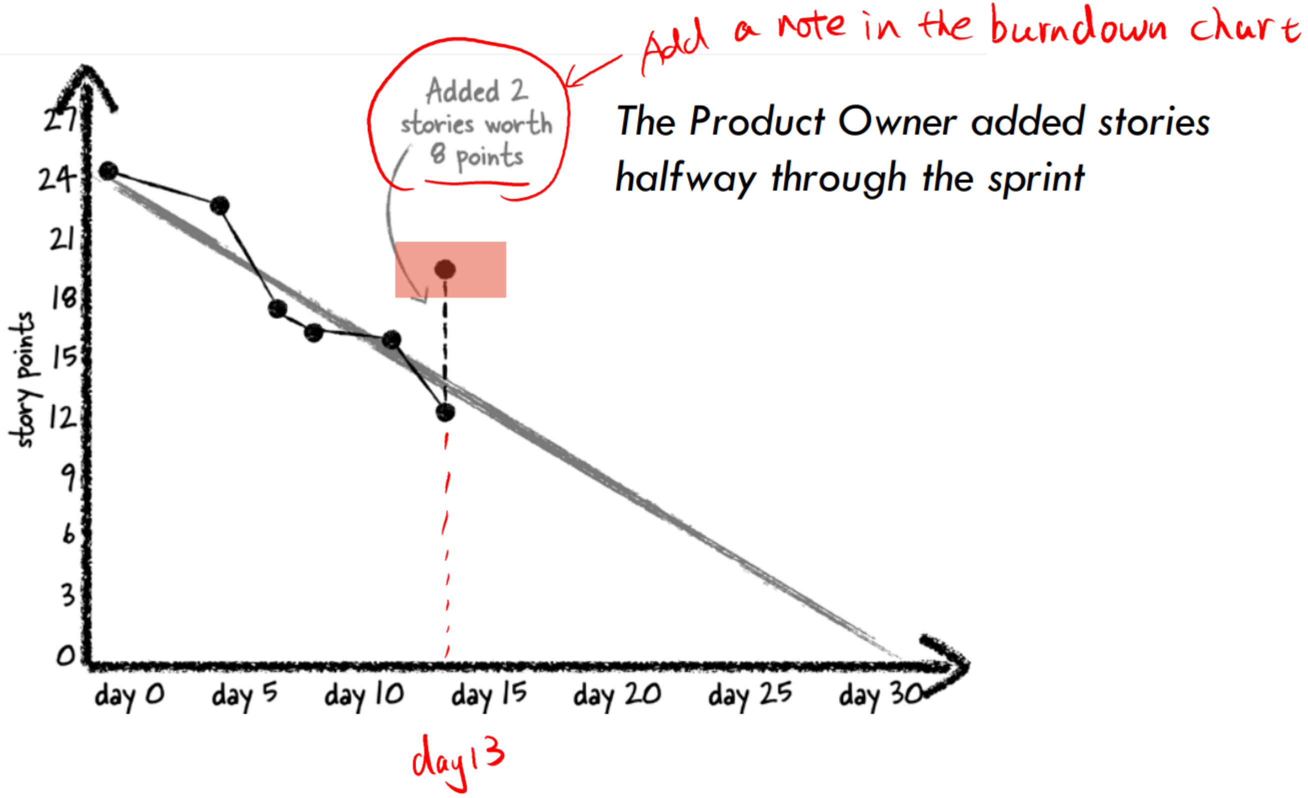


Burndown Charts based on Story Points (1)



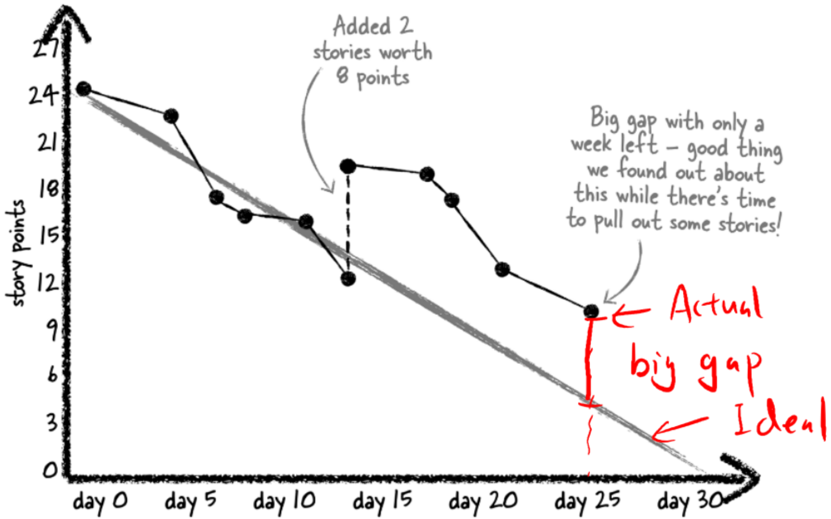
Two stories worth 7 points burned off

Burndown Charts based on Story Points (2)



The Product Owner added stories halfway through the sprint

Burndown Charts based on Story Points (3)



A gap between the burndown chart and the guideline tells you that there's a good chance you won't finish all of the stories by the end of the sprint.

Tracking Project Progress

Burndown Chart, Task Board - Tools

Tool Support for Agile SW Development

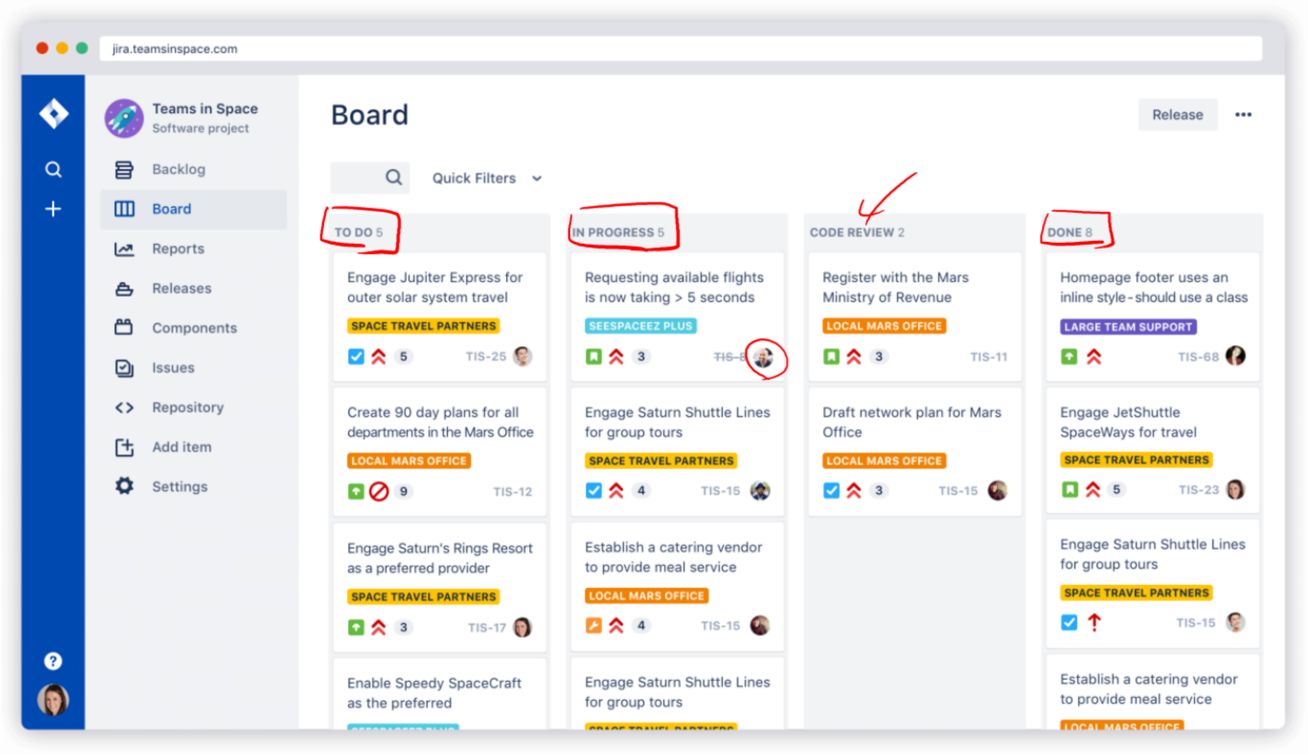
- JIRA Agile is a software tool for planning, tracking and managing software development projects
  - Supports different agile methodologies including Scrum and Kanban

- Jira Software supports Scrum Sprint planning, stand ups (daily scrums), Sprints and retrospectives
  - Including backlog management, project and issue tracking, agile reporting
    - E.g., Burndown and velocity charts, Sprint report
  - Scrum boards visualize all the work in a given Sprint

- Agile plugins such as GreenHopper, Agile Methods and Reports.

https://www.atlassian.com/software/jira/agile

Jira Agile - Scrum Board



jira.teamsinspace.com

Teams in Space
Software project

- Backlog
- Board
- Reports
- Releases
- Components
- Issues
- Repository
- Add item
- Settings

Board

Release •••

Quick Filters

TO DO 5

- Engage Jupiter Express for outer solar system travel — SPACE TRAVEL PARTNERS — 5 — TIS-25
- Create 90 day plans for all departments in the Mars Office — LOCAL MARS OFFICE — 9 — TIS-12
- Engage Saturn's Rings Resort as a preferred provider — SPACE TRAVEL PARTNERS — 3 — TIS-17
- Enable Speedy SpaceCraft as the preferred

IN PROGRESS 5

- Requesting available flights is now taking > 5 seconds — SEESPACEEZ PLUS — 3 — ~~TIS-8~~
- Engage Saturn Shuttle Lines for group tours — SPACE TRAVEL PARTNERS — 4 — TIS-15
- Establish a catering vendor to provide meal service — LOCAL MARS OFFICE — 4 — TIS-15
- Engage Saturn Shuttle Lines for group tours

CODE REVIEW 2

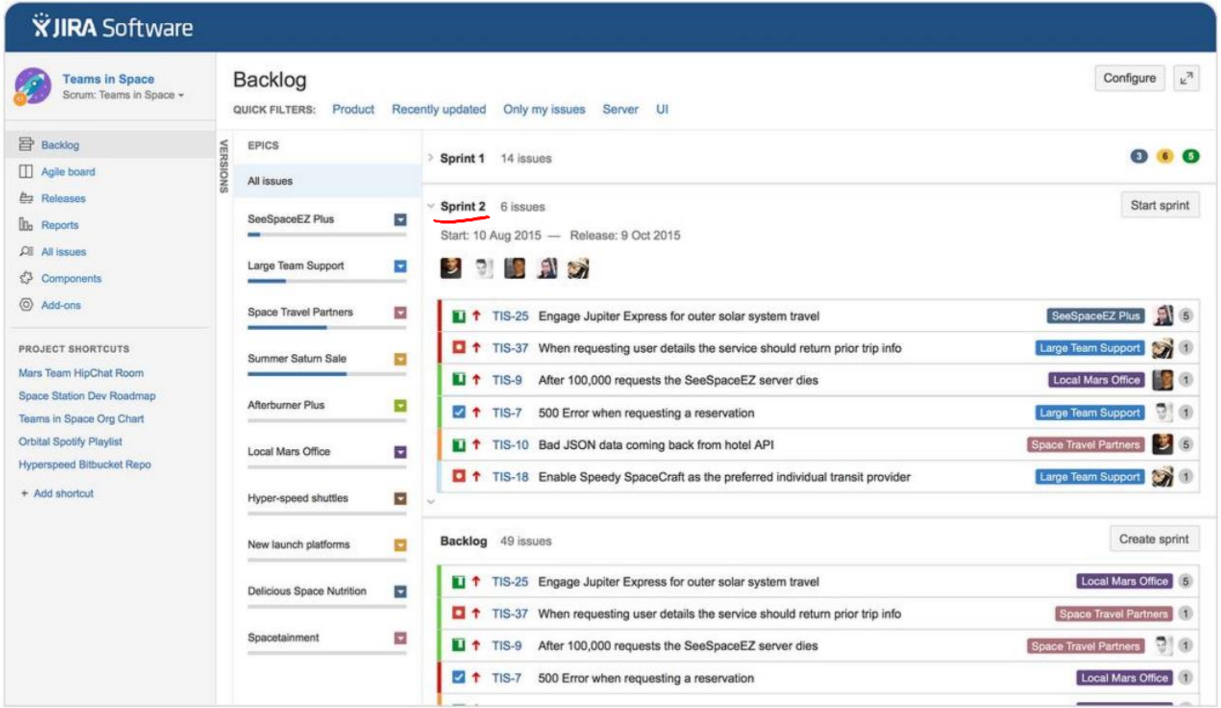
- Register with the Mars Ministry of Revenue — LOCAL MARS OFFICE — 3 — TIS-11
- Draft network plan for Mars Office — LOCAL MARS OFFICE — 3 — TIS-15

DONE 8

- Homepage footer uses an inline style - should use a class — LARGE TEAM SUPPORT — TIS-68
- Engage JetShuttle SpaceWays for travel — SPACE TRAVEL PARTNERS — 5 — TIS-23
- Engage Saturn Shuttle Lines for group tours — SPACE TRAVEL PARTNERS — TIS-15
- Establish a catering vendor to provide meal service — LOCAL MARS OFFICE

https://www.atlassian.com/software/jira/agile

Jira Agile - Sprint planning



Backlog

Sprint 1 14 issues 3 6 5

Sprint 2 6 issues

Start: 10 Aug 2015 — Release: 9 Oct 2015

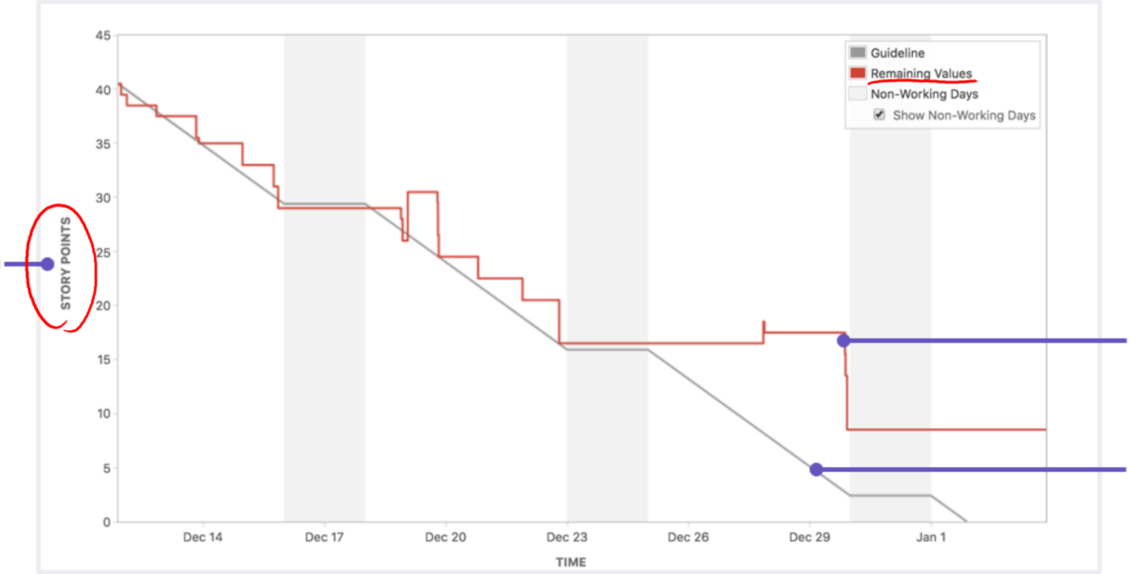
TIS-25	Engage Jupiter Express for outer solar system travel	SeeSpaceEZ Plus	5
TIS-37	When requesting user details the service should return prior trip info	Large Team Support	1
TIS-9	After 100,000 requests the SeeSpaceEZ server dies	Local Mars Office	1
TIS-7	500 Error when requesting a reservation	Large Team Support	1
TIS-10	Bad JSON data coming back from hotel API	Space Travel Partners	5
TIS-18	Enable Speedy SpaceCraft as the preferred individual transit provider	Large Team Support	1

Backlog 49 issues

TIS-25	Engage Jupiter Express for outer solar system travel	Local Mars Office	5
TIS-37	When requesting user details the service should return prior trip info	Space Travel Partners	1
TIS-9	After 100,000 requests the SeeSpaceEZ server dies	Space Travel Partners	1
TIS-7	500 Error when requesting a reservation	Local Mars Office	1

https://www.atlassian.com/software/jira/agile

Jira Agile - Burndown Charts



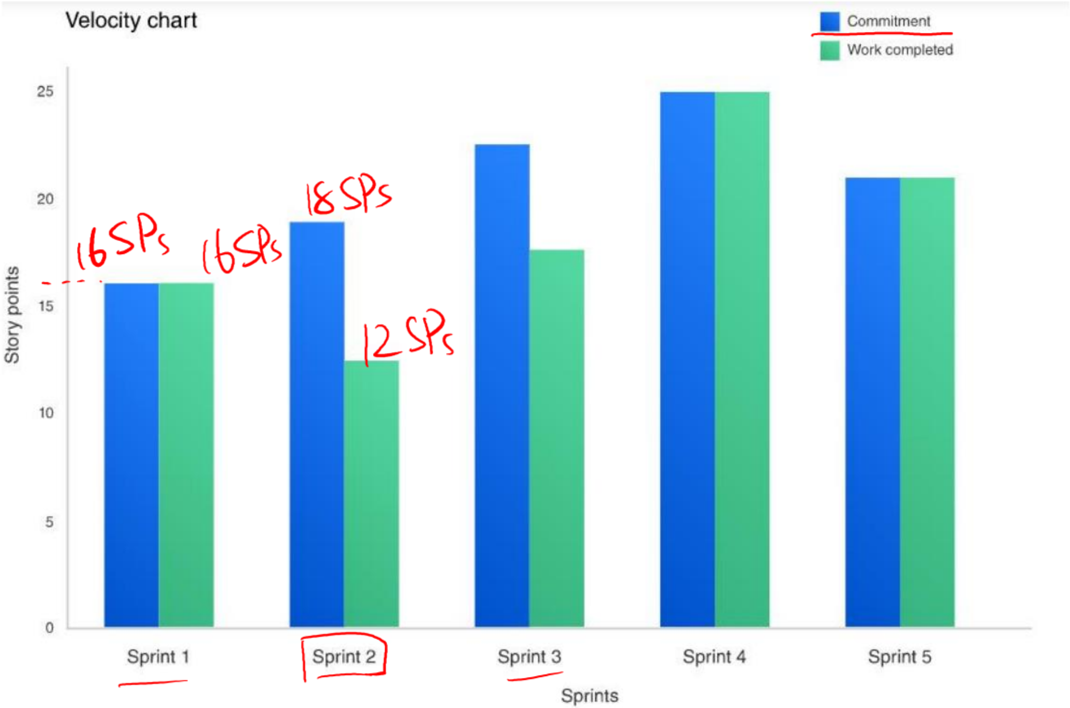
1 estimation in story points

2 amount of work left

3 Guideline; ideal progress

https://www.atlassian.com/agile/tutorials/burndown-charts

Jira Agile - Velocity Chart



https://confluence.atlassian.com/jirasoftwareserver/velocity-chart-938845700.html

References

- Andrew Stellman, Margaret C. L. Greene 2014. Learning Agile: Understanding Scrum, XP, Lean and Kanban (1st Edition). O’Reilly, CA, USA

- Mike Cohn. 2005. Agile Estimating and Planning. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Tutorial: Estimation & Tracking Progress (Story Points, Task Board, Burndown Chart) + Group Project 2 Sprint Demo 1

Next Lecture: Ethics, IP and Open Source Software

- Ethics and Professional Frameworks
- Intellectual Property
- Open-source Software