

# CE29x Team-Project Challenge

## Keeping your Project on Track, with Agile

Professor Anthony Vickers

Room: INW.3.17

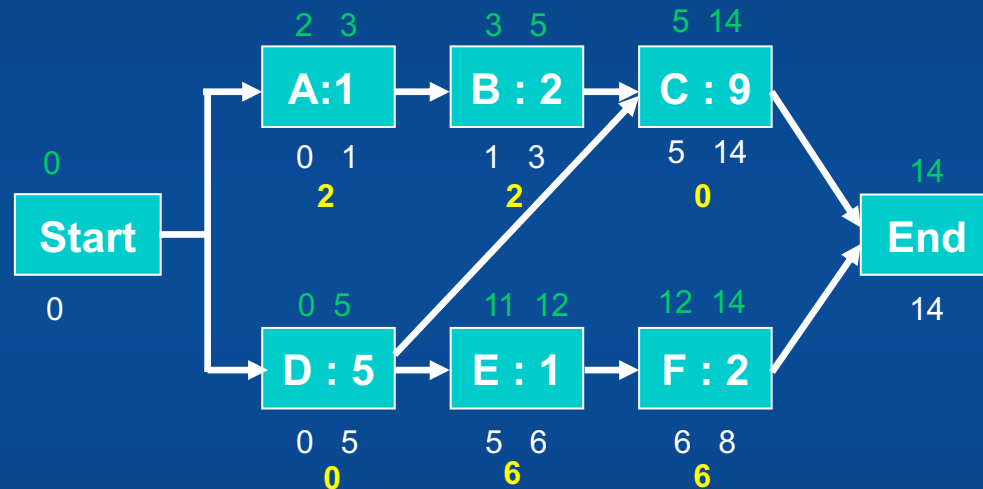
e-mail: vicka

Thanks to previous work by Michael Fairbank

# Scrum-meeting organisation Tips:

- Reminder:
  - Assign a scrum master each week
  - Scrum master responsible for
    - clicking “complete sprint” + “start sprint”
    - Holding the sprint retrospective
    - Leading the discussion on building the next sprint
  - Otherwise it’s chaotic!
  - Arrive on time
- Try to ensure you just have one scrum board in Jira
  - don’t want things getting cluttered

# Review of previous lecture

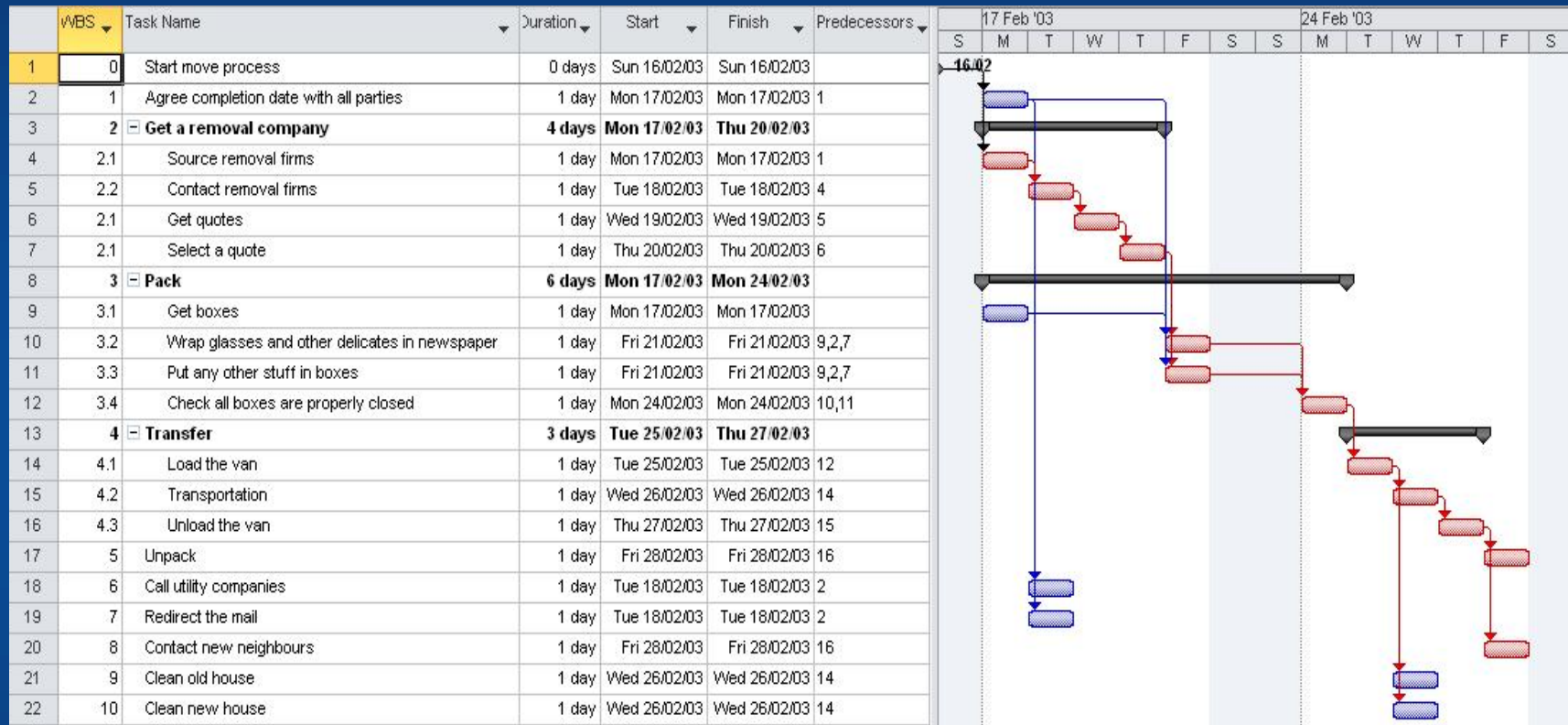


Critical path is “Start-D-C-End”

# Today's Lecture

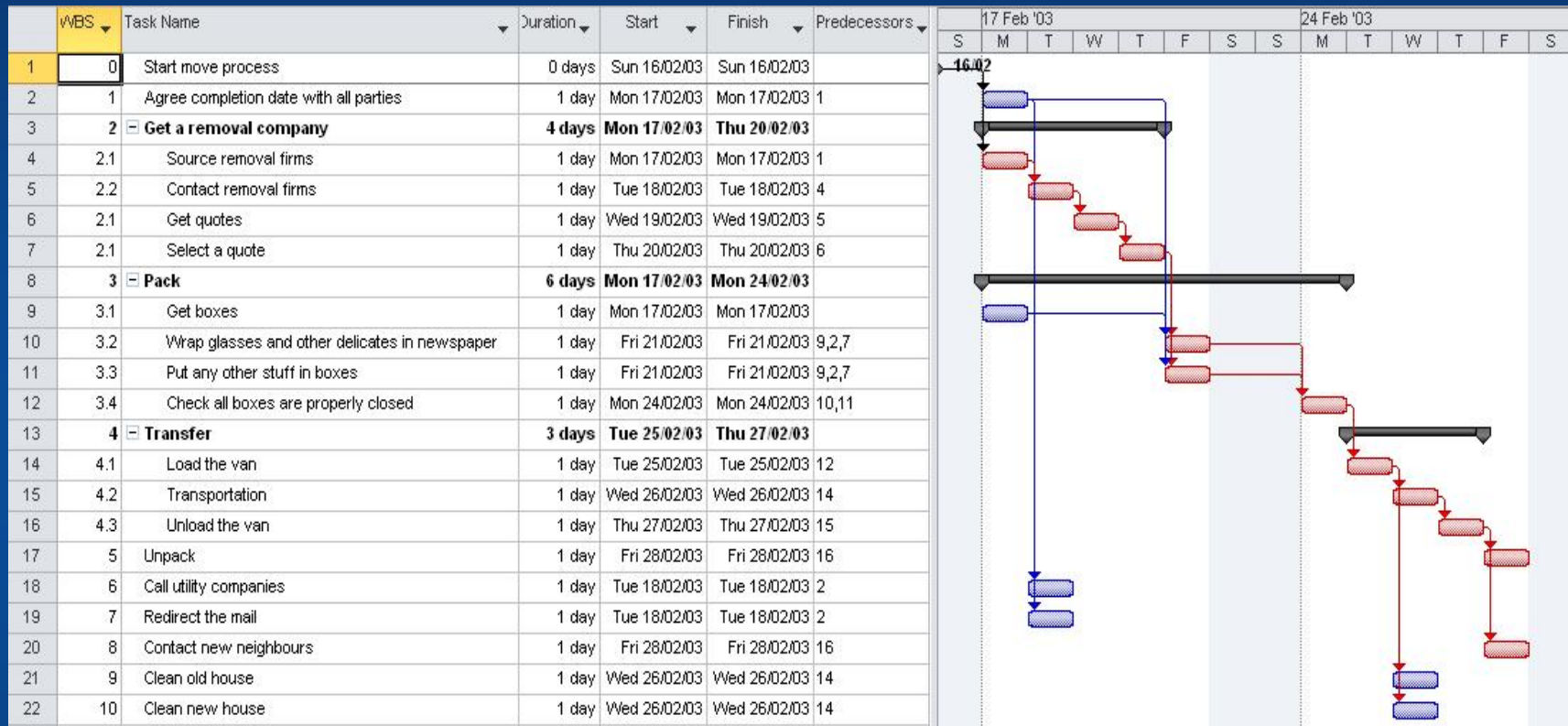
- Introduce Gantt Charts
  - Speeding a project up
  - Resource levelling
- Keeping your project on track in Agile:
  - Stand-up meetings
  - Progress reporting
  - Epics + Versions + Roadmaps
  - Burndown Charts
  - Continuous Integration

# Gantt Charts + Microsoft Project



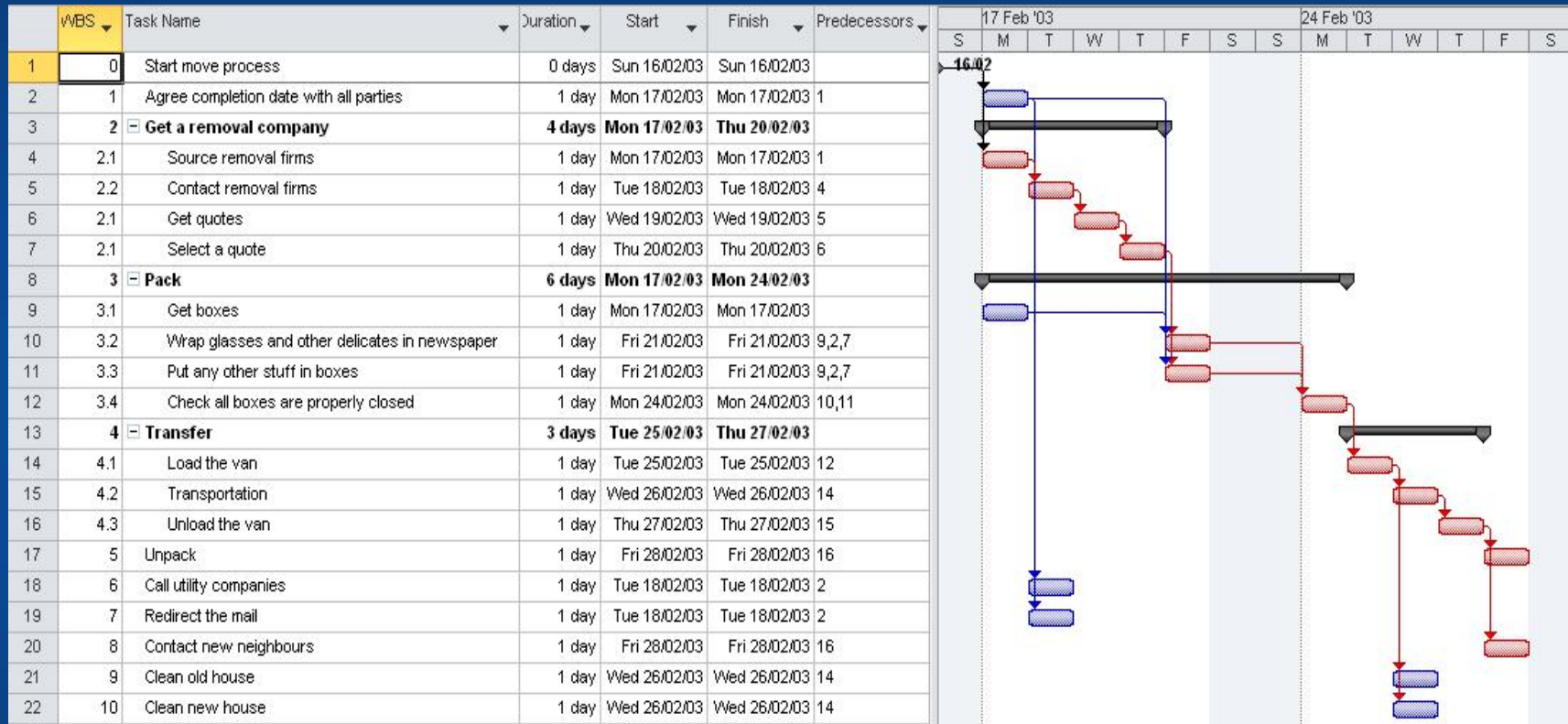
- \* Pink tasks on Gantt chart are critical tasks, here on Microsoft Project
- \* How many critical pathways are there here?

# Gantt Charts + Microsoft Project



- \* The above Gantt chart shows project schedule
- \* Project has a scheduled end date of Friday 28<sup>th</sup> Feb
- \* Is client happy with that? What if not?

# Moving home example



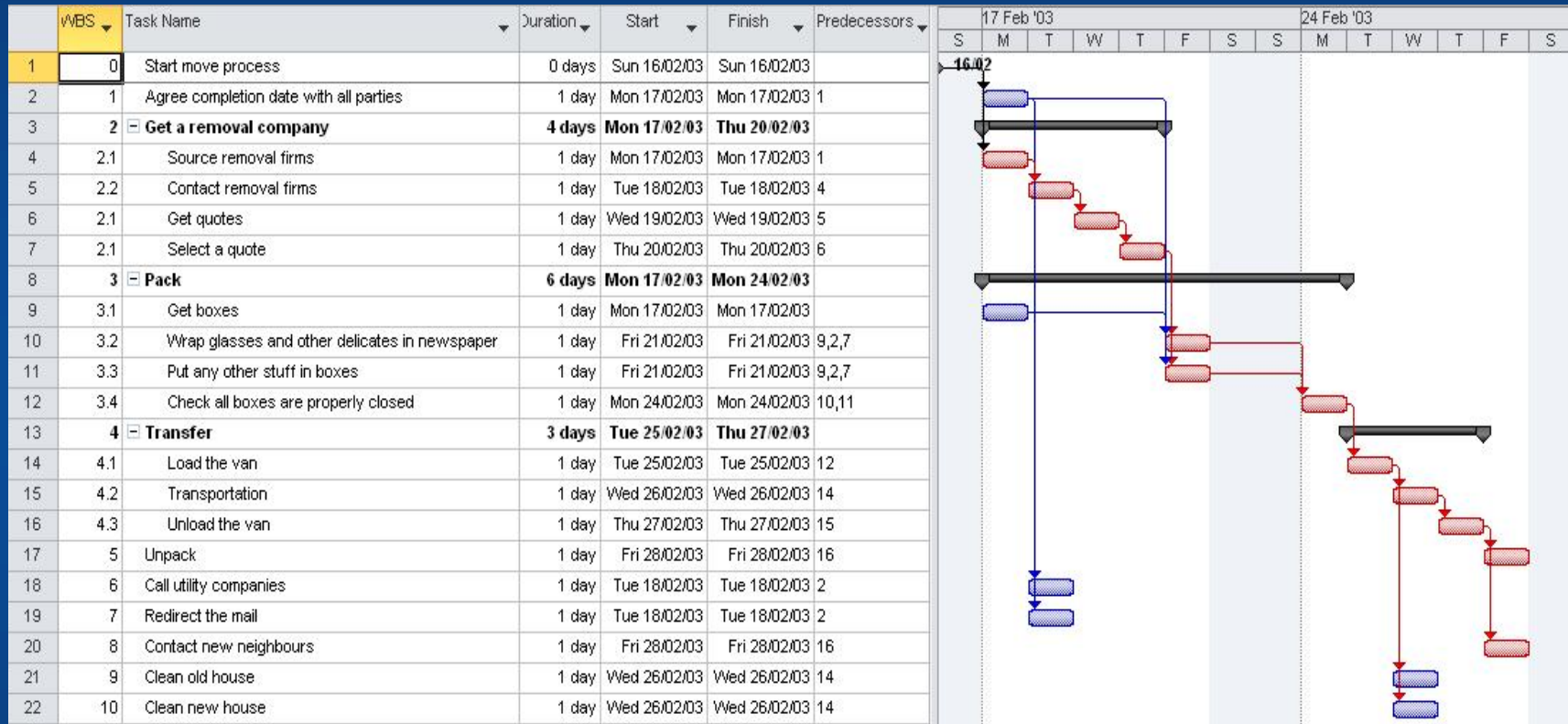
WBS

Dependencies

Gantt Chart

Tasks

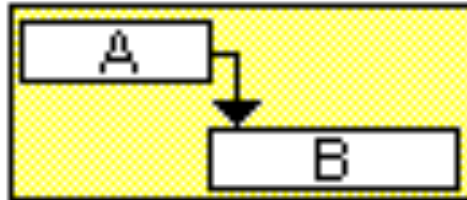
# Moving home example



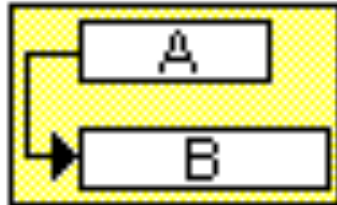


# Gantt Chart Task dependencies

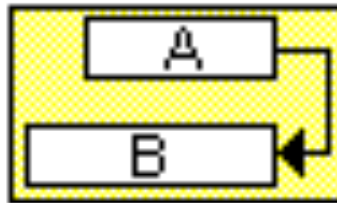
Finish-to-start  
(FS)



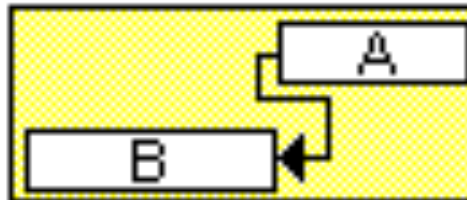
Start-to-start  
(SS)



Finish-to-finish  
(FF)



Start-to-finish  
(SF)



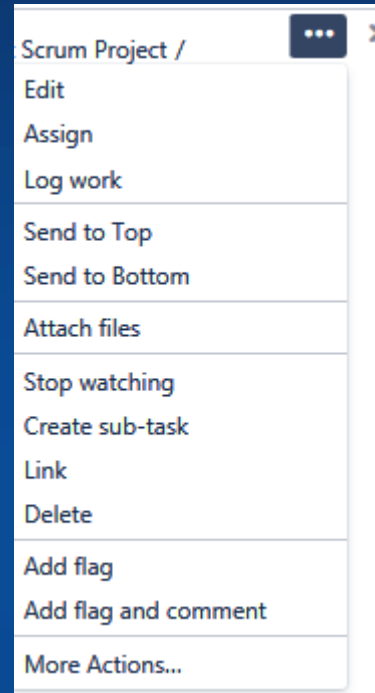
- \* B cannot start until A finishes.  
You can't get cash out of a bank until you have created an account.
- \* B cannot start until A starts.  
Once money starts to be paid in you can withdraw cash (including an overdraft).
- \* B cannot finish until A finishes.  
You can't close the account until you have paid back any overdraft.
- \* B cannot finish until A starts.  
You cannot close the account until you transfer the overdraft to another bank.

- \* Try to use FS whenever possible, if possible, always. Keeps life simple.
- \* See [https://en.wikipedia.org/wiki/Dependency\\_\(project\\_management\)](https://en.wikipedia.org/wiki/Dependency_(project_management))

# Task Dependencies in Agile

For a task-dependency in Jira, create a “link” to on an issue.

- \* To say a task can't be done until another issue is resolved, selects “is blocked by”
- \* Jira does not enforce these rules:
  - \* These are just *notes* for you to consider when you choose which issues to work on next



Link

Jira Issue

Web Link

Select a Jira issue to link this issue to

This issue

Issue

Attachment

Comment

blocks

is blocked by

clones

is cloned by

duplicates

is duplicated by

relates to

Drop files to attach, or browse.

Style

B I U A

+

# Speeding a project up

# Techniques for Compression

- \* What can we do if the client is not happy with the scheduled finishing time? Schedule-compression techniques:
- \* Fast Tracking: performing activities in parallel instead of sequentially
  - \* Example: Start assembling / building components before the design is complete. (Even though this is potentially violating a dependency.)
  - \* Use more incremental development models
  - \* Often results in increased rework

# Techniques for Compression

- \* Crashing – analyse cost and schedule to determine how to obtain the greatest compression by adding or reassigning resources
  - \* Not always realistic
  - \* Often results in increased costs, because resource allocation is not optimised
  - \* Adding resources is rarely a good solution

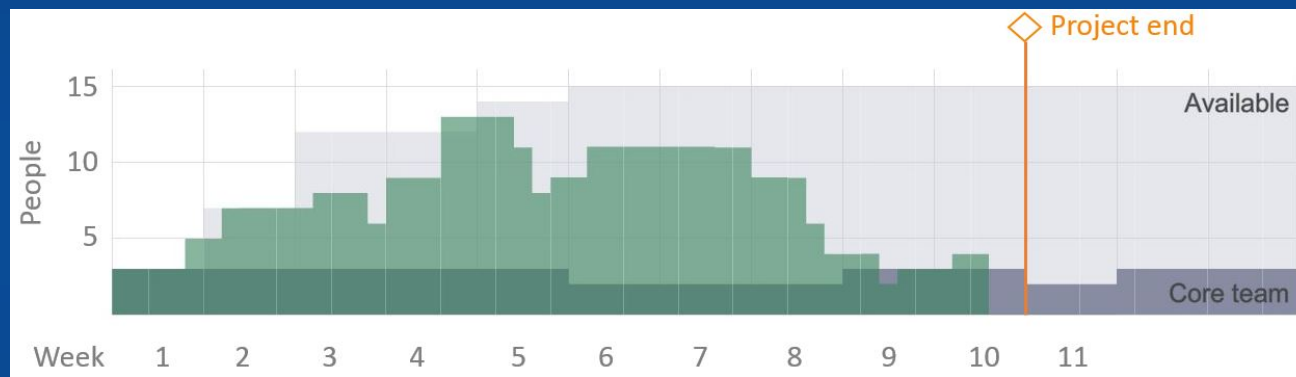
# Balancing Resources

# Balancing Resources

- \* The duration of most tasks is significantly affected by the resource assigned
  - \* Their capabilities
  - \* Their availability
- \* Don't assume that putting two people to work on a task will divide its duration by two – (see Fred Brooks – *The Mythical Man Month*).

# Resource Histogram

- \* A histogram of resources can be drawn after the Gantt Chart has been made



[Image source](#)

- \* What is the maximum number of people needed on the project at any one time?
- \* What is the size of the core team?
- \* Why is the maximum in the above graph a problem?



# Resource Histogram

- \* Any high difference in the level of resources will need to be well monitored, because integration of a new person or new teams always takes time
- \* Limiting these differences is called Resource Levelling
  - \* Achieve this first by moving *non-critical* tasks so that as few happen concurrently as possible.
    - \* Microsoft Project has a function to do this automatically
  - \* Further levelling by
    - \* Moving critical activities (i.e. extending the project),
    - \* splitting activities,
    - \* use alternative dependencies (start to start, end to end, start to end, lagged dependencies)

# Keeping your project on Track, with Agile

# Keeping your project on track

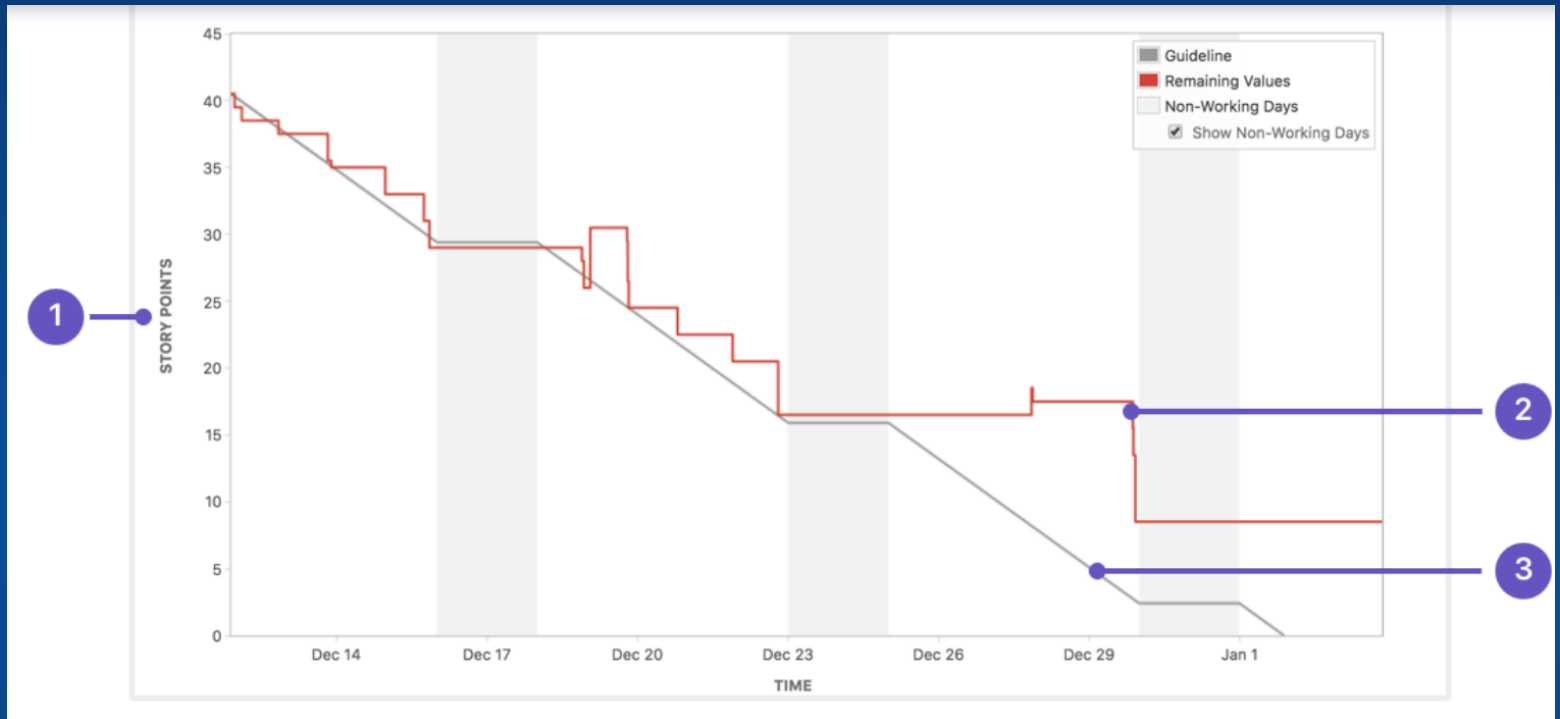
We need to monitor the project closely...

- \* To ensure it is on track to meet the deliverables, on time.
- \* To rapidly spot any deviation, and to respond and correct problems early.
- \* It's a key task for a project manager

# Keeping your project on track

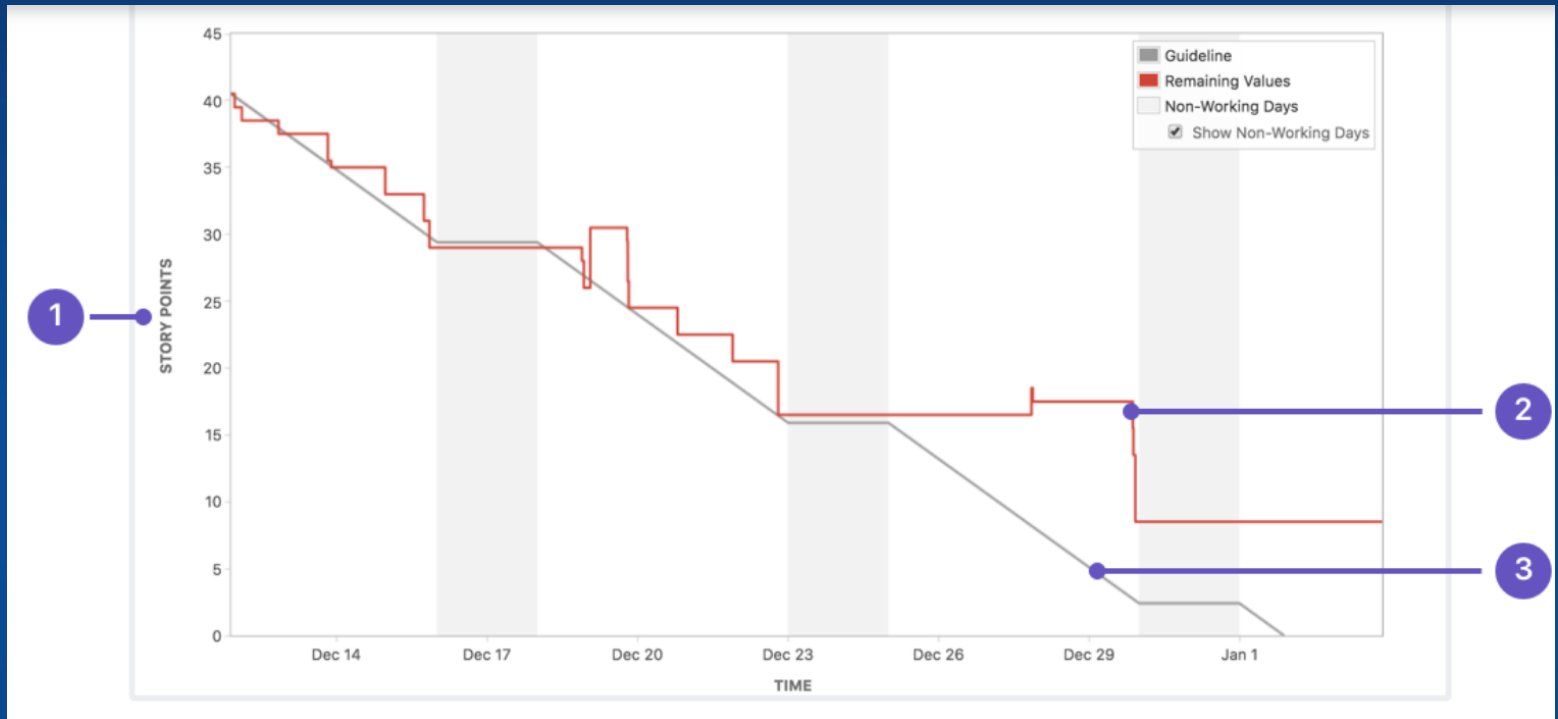
1. Burndown Chart
2. Keeping your sprint on target with stand-up meetings
3. Longer term planning in Agile
  - Roadmaps, Epics + Versions
4. Reporting in Jira
5. Continuous Integration Tools

# Burndown Chart



- Shows how a sprint is progressing, as story points get completed.
- A burndown chart shows the amount of work that has been completed in an epic or sprint, and the total work remaining.

# Burndown Chart

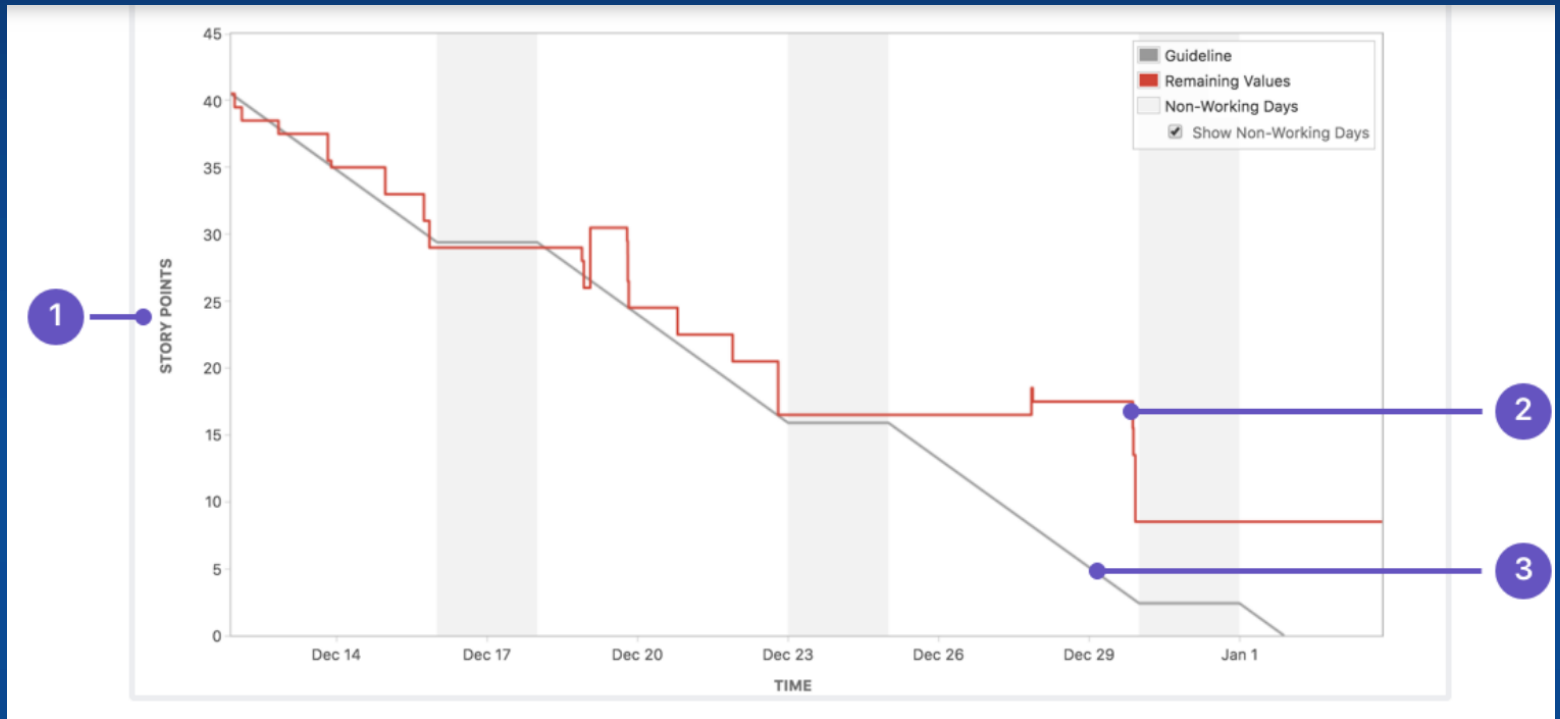


1. Estimation statistic (Story Points)
2. Remaining Values: Total story points left to go during sprint
3. Guideline: The grey line shows an approximation of where your team should be, assuming linear progress.

If red line is below grey line, then team is on track to completing all their work by the end of sprint.

*Solid grey vertical bars represent weekends – no work expected then.*

# Burndown Chart



- Burndown charts are used to predict your team's likelihood of completing their work in the time available.
- Red line should shows total story points remaining in the current sprint.
  - It should only decrease as work gets moved to “done”
  - If it increases then it means more work was put into sprint mid-way (bad)
    - That’s a sprint scope-change, which is meant to be avoided.
    - Or an issue that’s moved to done has to come back out again, e.g. due to it not being done properly the first time (e.g. poor testing / bugs); also bad.

Keeping your sprint on target  
- stand up meetings



# Keeping your sprint on target

- \* In the industry, every morning e.g. at 9am, a scrum team will hold the daily stand-up meeting.
- \* In each stand-up meeting, members take turn to speak, one at a time. Each person specifies 3 things:
  1. What did I work on yesterday?
  2. What am I working on today?
  3. What issues are blocking me?

# Daily Stand-up Meetings

1. What did I work on yesterday?
2. What am I working on today?
3. What issues are blocking me?

- \* These questions highlight progress and help the team flag up blockers.
- \* Also, it strengthens the team when everyone shares the progress they're contributing to the team.

# Daily Stand-up Meetings

1. What did I work on yesterday?
2. What am I working on today?
3. What issues are blocking me?

- \* To keep attention focused on the speaker, an object may be passed round held by the current speaker

  - \* e.g. a squeeze toy

- \* The stand-up should be brief, e.g. limited to 10 mins; and at a consistent time of day

- \* Stand-up success can be reviewed in the sprint retrospective

# Daily Stand-up Meetings

- \* In our one-hour labs, there isn't time to do a scrum meeting, a stand-up meeting, and a sprint retrospective.
  - \* However it is a key part of agile in professional projects
  - \* You should experience it as closely as possible
- \* Hence you should arrange a physical or virtual standup meeting at least once per week, e.g. every Friday / Monday
  - \* This will prevent students coming back next week saying "I couldn't do the work assigned because I got stuck" to the surprise of everyone
  - \* You could hold a virtual meeting by creating an issue each sprint called "Stand-up meeting" and commenting on it
    - \* Each student needs to answer the 3 questions in that comment thread, in turn. By the end of the working day all students should have contributed to it

# Daily Stand-up Meetings

1. What did I work on yesterday?
2. What am I working on today?
3. What issues are blocking me?

\* Daily stand-up meetings are part of Agile Scrum

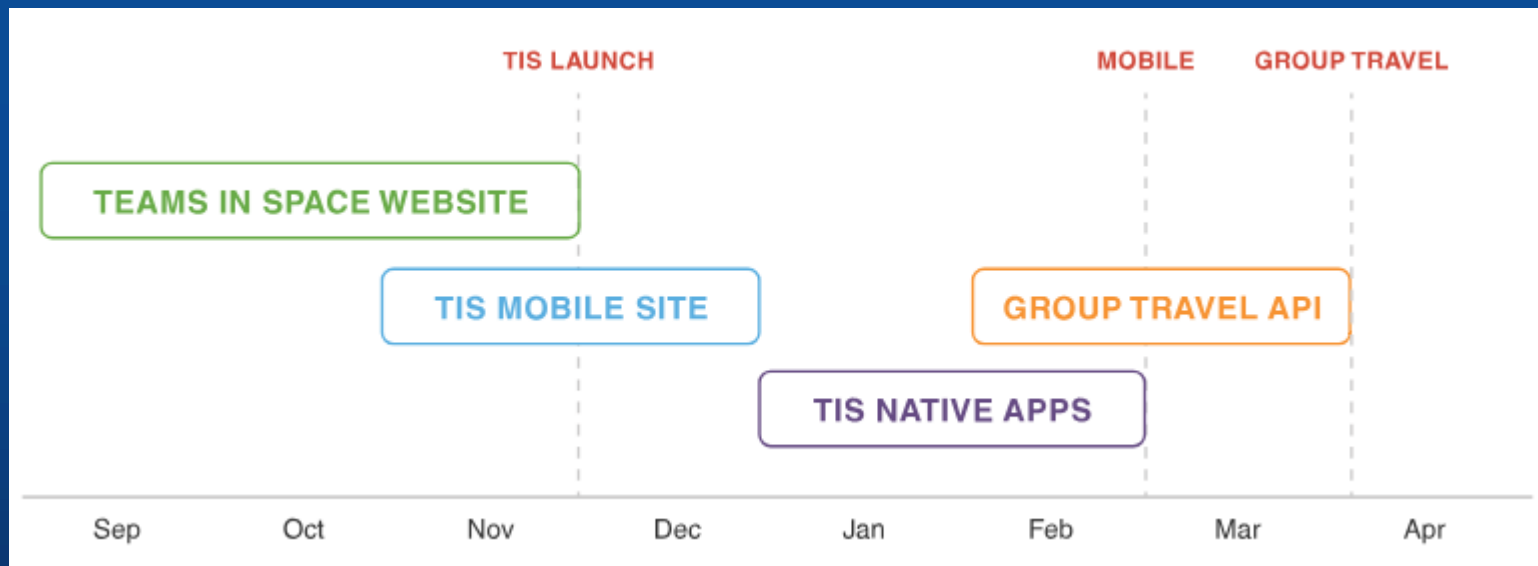
\* For more information see

<https://www.atlassian.com/agile/scrum/standups>

# Longer Term Planning in Agile

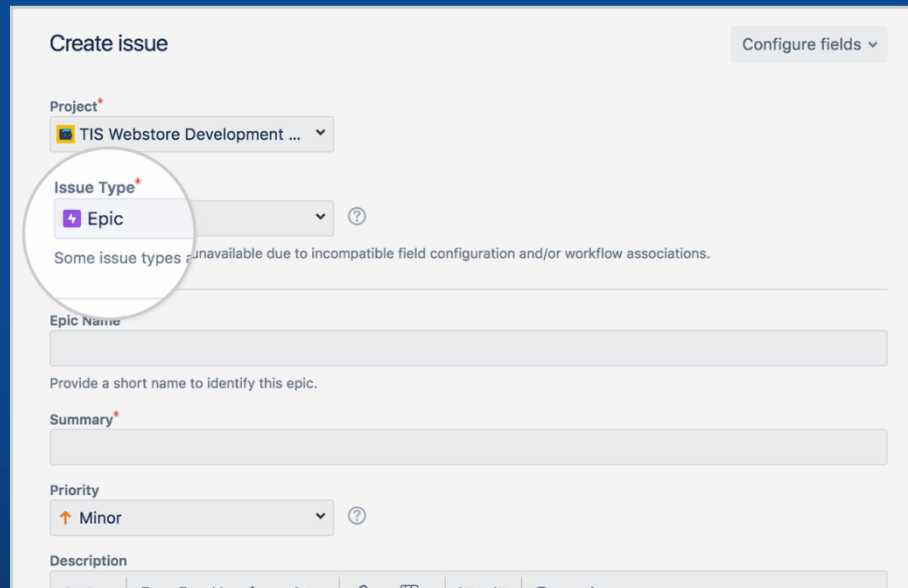
# Long term planning in Agile

- \* Sprints tend to focus on short term product improvements
  - \* Can be problematic if you don't aim for a bigger picture
- \* To plan long term releases and features, a product roadmap can be used. <https://www.atlassian.com/agile/product-management/roadmaps>



# Epics

- \* Stories and Tasks are only focussed on small product features
- \* Stories and Tasks can be grouped together into “Epics”.
- \* First create the Epic as an issue in its own right

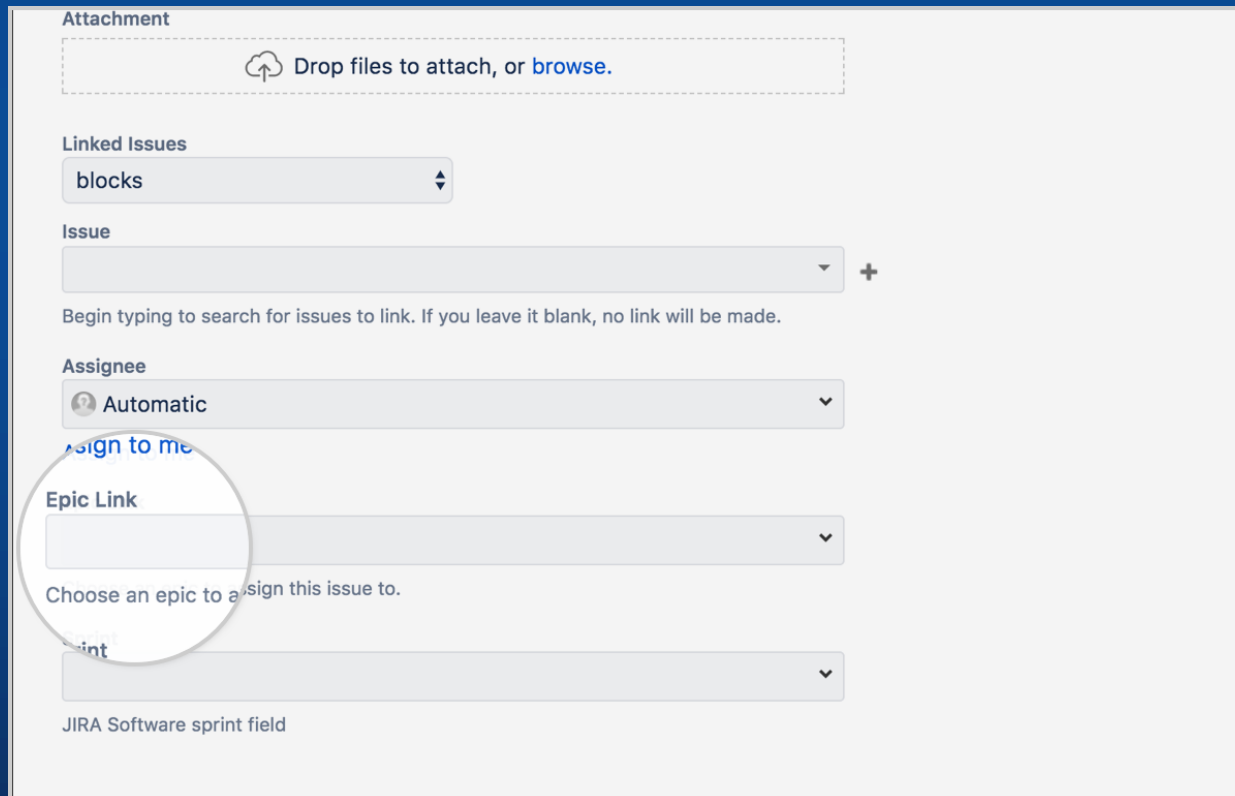


The screenshot shows the 'Create issue' form in Jira. The 'Project' field is set to 'TIS Webstore Development ...'. The 'Issue Type' dropdown is highlighted with a red circle and set to 'Epic'. Below the dropdown, a message states: 'Some issue types are unavailable due to incompatible field configuration and/or workflow associations.' The 'Epic Name' field is empty, with a prompt 'Provide a short name to identify this epic.' below it. The 'Summary' field is also empty. The 'Priority' dropdown is set to 'Minor'. The 'Description' field is at the bottom, with a rich text editor toolbar visible.



# Epics

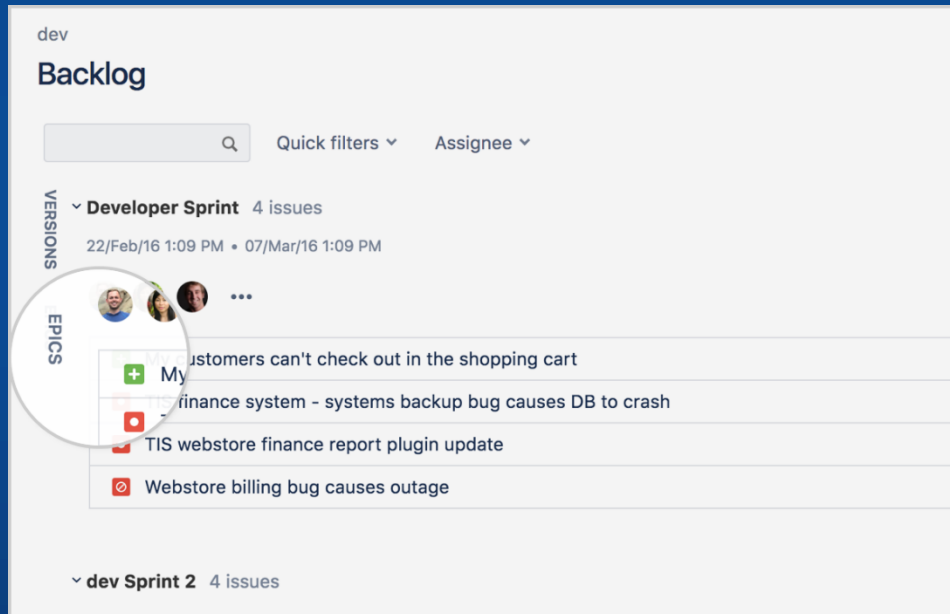
- \* Then assign different stories to that epic
- \* Edit the story's attributes, and assign that story to your new epic:



The screenshot displays the 'Attachment' section of a JIRA issue, which includes a dashed box for file uploads and a 'browse' link. Below this is the 'Linked Issues' section with a dropdown menu currently set to 'blocks'. The 'Issue' section features a search input field with a '+' icon and a placeholder text: 'Begin typing to search for issues to link. If you leave it blank, no link will be made.' The 'Assignee' section shows a dropdown menu with 'Automatic' selected. The 'Epic Link' section is highlighted with a red circle; it contains a dropdown menu and the text 'Choose an epic to assign this issue to.' Below this is a 'Sprint' section with a dropdown menu and the text 'JIRA Software sprint field'.

# Epics

- \* To view epic, go to epics panel.
- \* Or view on backlog:



- \* Use Epics to organise your stories into related groups.
- \* Further information:  
<https://www.atlassian.com/agile/tutorials/epics>

# Epic Burndown Chart

- \* Shows how many story points remaining in the epic, as the sprints progress
- \* Lets you predict if your epic will be completed in time



# Epic Burndown Chart

- \* Light blue=number of story points originally in epic at start of each sprint.
  - \* Light green=number of story points done in that sprint
  - \* Dark blue = number of story points added to epic during sprint
  - \* Dark/light grey: predictions, based on team's velocity
- Q: When do we think this epic is going to be completed?



# Versions

- \* Stories Tasks and Epics can be grouped further, into “Versions”
- \* Version is complete when all of its issues are complete

The screenshot displays the 'Releases' section in Jira. At the top, there are 'QUICK FILTERS' for 'Released', 'Unreleased', and 'Archived'. Below this is a table with columns: Version, Status, Progress, Start date, Release date, and Description. The table lists four versions: Version 4.0 (UNRELEASED, No issues), Version 3.0 (UNRELEASED, progress bar), Version 2.0 (UNRELEASED, progress bar), and Version 1.0 (RELEASED, progress bar). A context menu is open for Version 2.0, showing options: 'Version: Version 2.0', '9 issues in total', 'Done', 'In Progress', 'To Do', and 'All issues'. The menu is annotated with numbers 2, 3, and 4. The 'Done' option is highlighted. The 'To Do' option is also highlighted. The 'All issues' option is highlighted. The 'In Progress' option is highlighted. The 'Done' option is highlighted. The 'Version: Version 2.0' option is highlighted. The '9 issues in total' option is highlighted. The 'Done' option is highlighted. The 'In Progress' option is highlighted. The 'To Do' option is highlighted. The 'All issues' option is highlighted.

Version	Status	Progress	Start date	Release date	Description
Version 4.0	UNRELEASED	No issues			
Version 3.0	UNRELEASED				
Version 2.0	UNRELEASED				
Version 1.0	RELEASED				

# Versions

- \* Can view Release Burndown chart
- \* Same idea as epic burndown chart



- \* For more information on creating versions in Jira:  
<https://www.atlassian.com/agile/tutorials/versions>

# Other reports in Jira

- \* Burndown chart
- \* Velocity chart (Previous lecture)
  - \* Gives average rate of story points delivered per week
- \* Version / Epic burndown chart

# Other reports in Jira

## Sprint Report

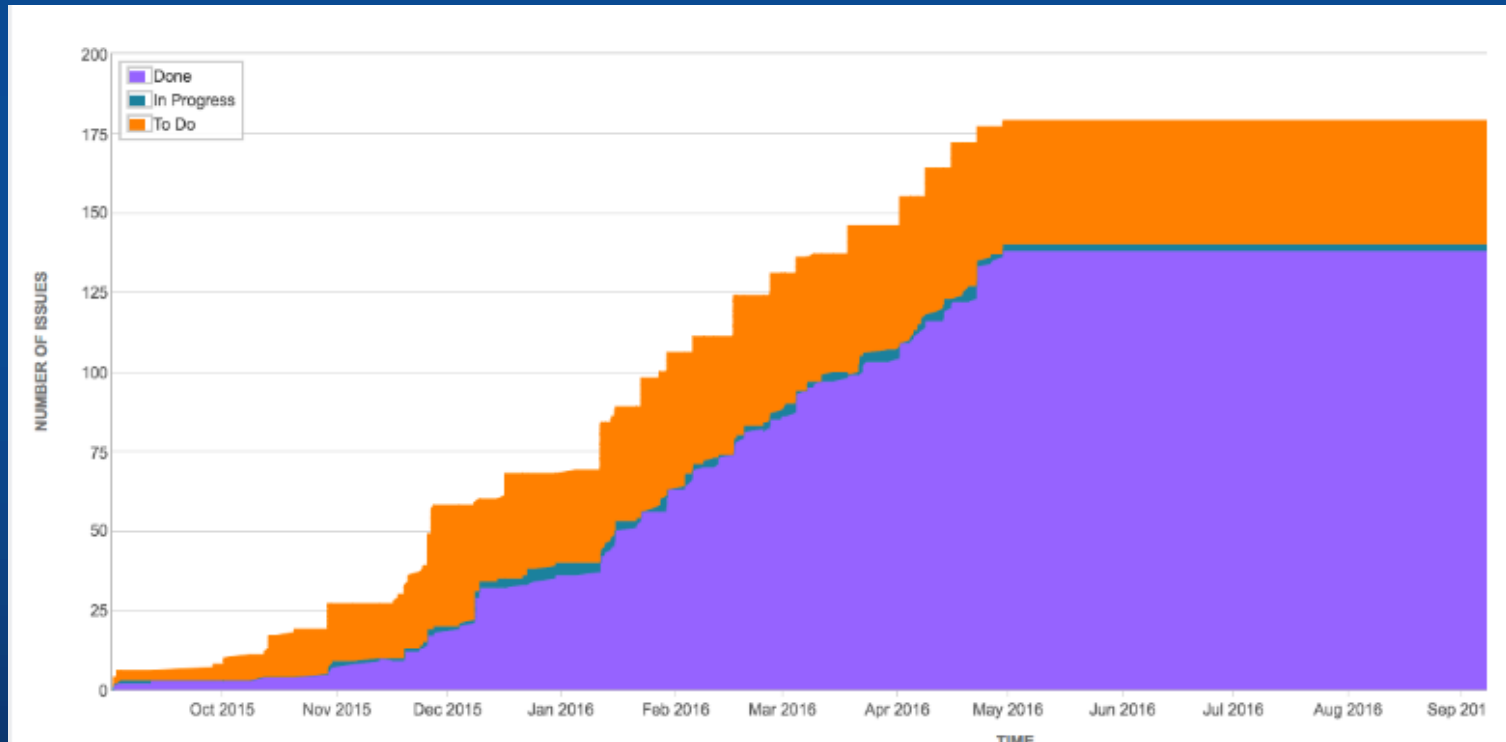
- \* Lists issues completed in a sprint
- \* Good information for discussion during sprint retrospective.
- \* [Further information](#)



# Other reports in Jira

## \* Cumulative Flow Diagram –

- \* Checks that rate of issue's being done is keeping up with rate of issue creation
- \* Useful for Kanban projects + support tickets work-rate management.



\* [Further information](#)

# Continuous Integration Tools

# Continuous Integration (CI)

- \* You can configure the gitlab server to perform CI
  - \* See <https://docs.gitlab.com/ee/ci/> for details
- \* CI can be used to continually run unit-tests on your application
  - \* And email you whenever one of them breaks
  - \* Tests are scheduled to run automatically whenever someone pushes code to the server
- \* Obviously you need to have first created a good suite of unit-tests for this to be effective

# Continuous Integration (CI)

- \* Continuous testing in CI is an effective way to stop bugs you've fixed once from ever coming back
- \* Vital when many people are contributing towards a project collaboratively
  - \* Many ways for people to break things!

# Continuous Integration (CI)

- \* CI can also be used for “continuous deployment” (CD)”
  - \* Live website automatically updates itself when you push code to gitlab
  - \* Can configure the specifics
    - \* e.g. it only goes to a team of testers’ version of the website first!
- \* Also, Continuous security
  - \* Emails you whenever one of your software tools is out of date, and therefore vulnerable

# Summary of Lecture

- \* Gantt Chart, Project Compression, and Resource Levelling
- \* Keeping your project on track in Agile
  - \* Stand-up meetings
  - \* Viewing burndown charts
  - \* Producing a project Roadmap, Epics and Versions
  - \* Continuous Integration tools

\* Lecture quizzes:

\* **Critical-path analysis I**

\* **Keeping the project on Track**