Agile Development

CITS3403 Agile Web Development

From Agile in a Nutshell, by Jonathan Rassmusson

Further reading: The agile handbook

Semester 1, 2021

Agile Goals

Agile is a way to manage projects. It can be used for virtually anything, but it was founded in software development. This handbook focuses on agile for software development, but many of the principles can be expanded to other fields.

Agile breaks down larger projects into small, manageable chunks called iterations. At the end of each iteration (which generally takes place over a consistent time interval) something of value is produced. The product produced during each iteration should be able to be put into the world to gain feedback from users or stakeholders.

Unlike Waterfall project management, which is strictly sequenced: you don't start design until research is done and you don't start development until the designs are signed off on; agile has designers, developers and business people working together simultaneously.

As made popular by the "Agile Manifesto", agile values:

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

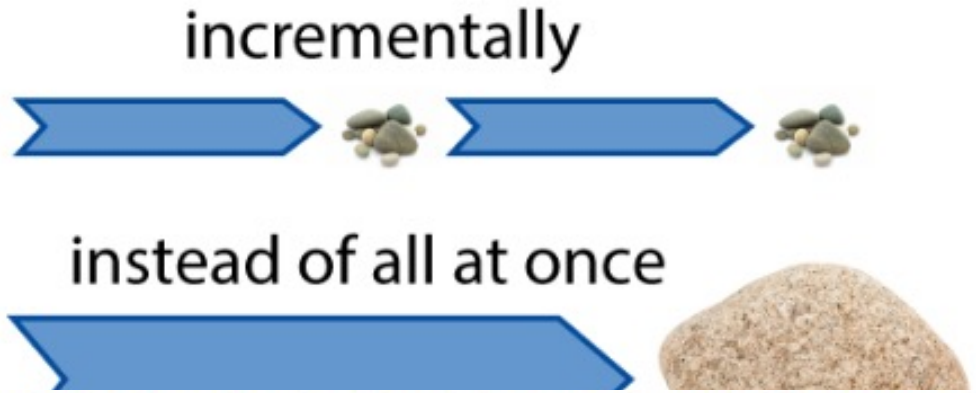
12 Key Principles

THE UNIVERSITY OF WESTERN AUSTRALIA

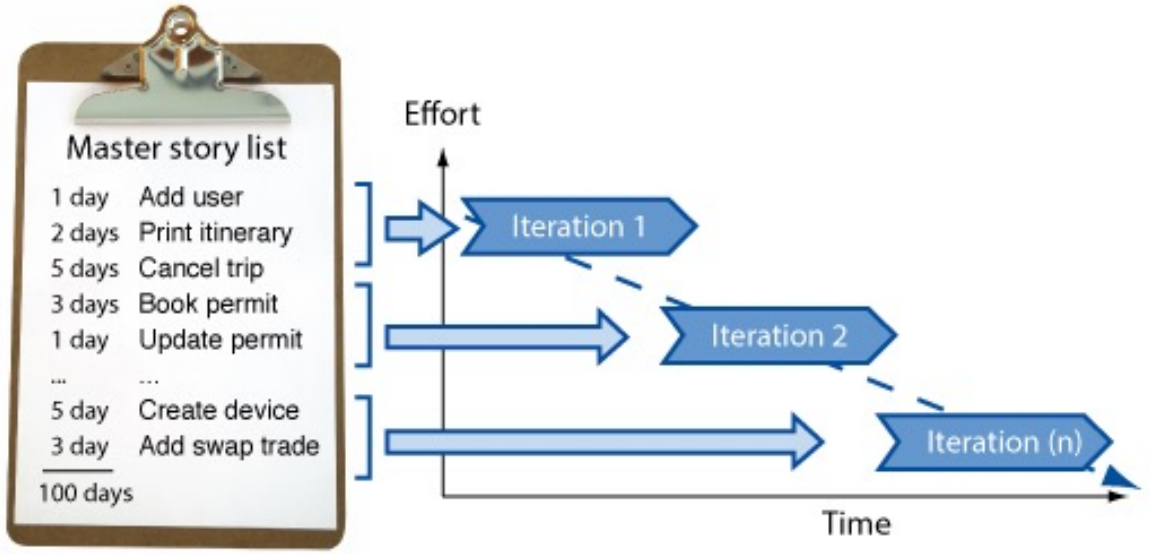
1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
3) Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
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.

What is Agile

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.

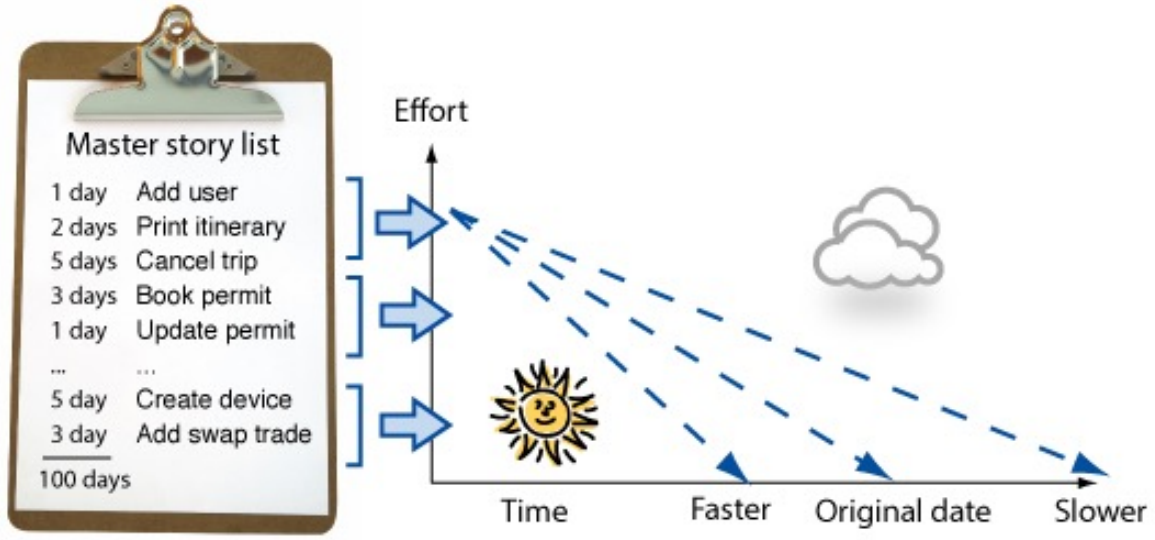


It works by breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short two week cycles called iterations.

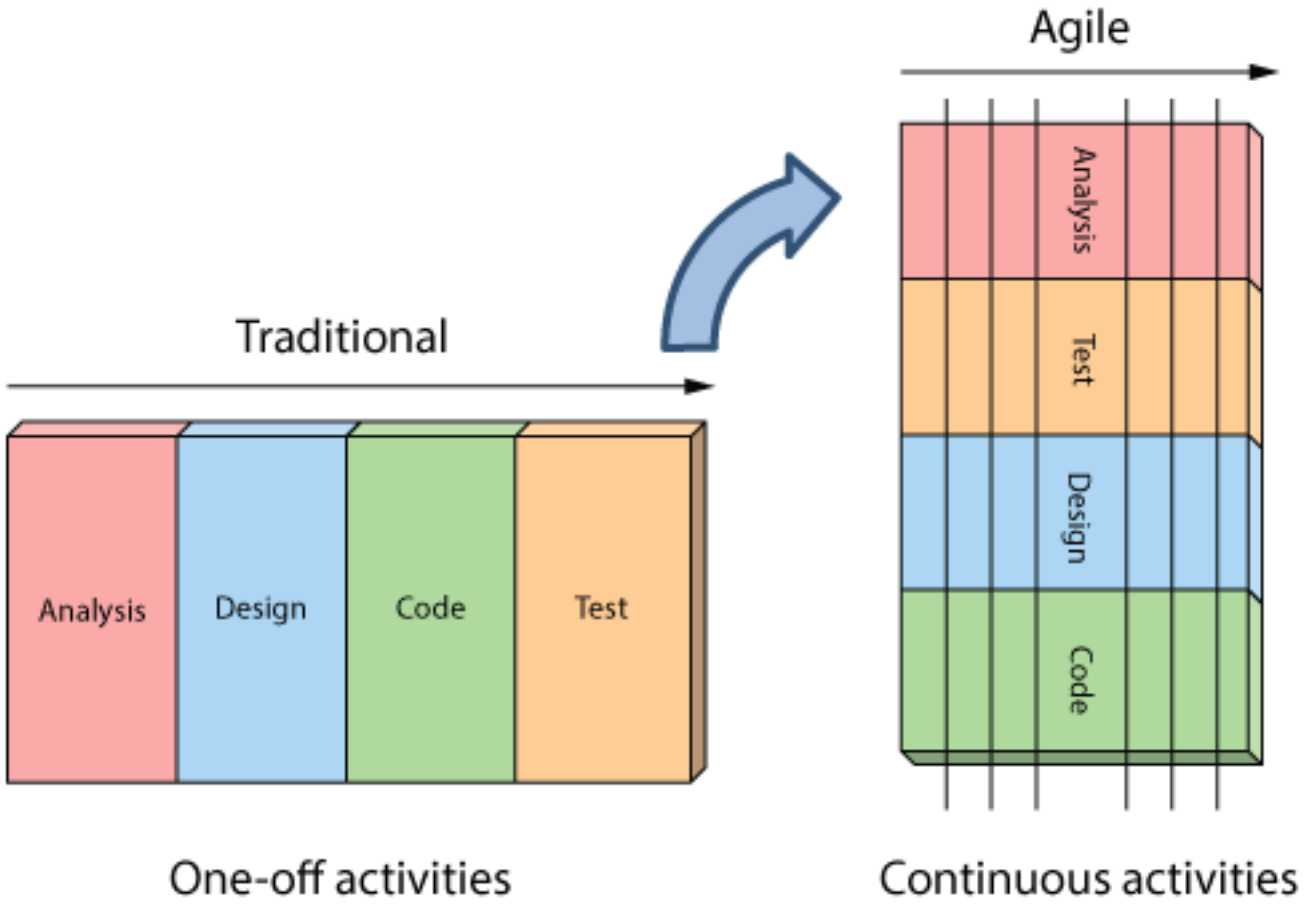


How does it work:

- You make a list
- You size things up
- You set priorities
- You start executing
- You update the plan as you go…

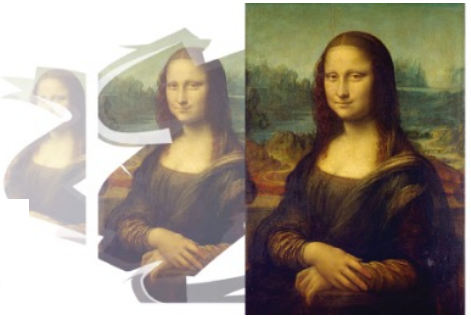


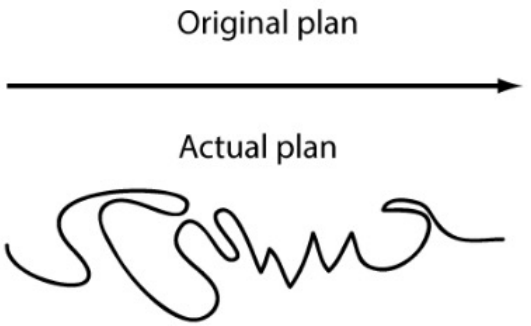
Analysis, design, coding testing are continuous

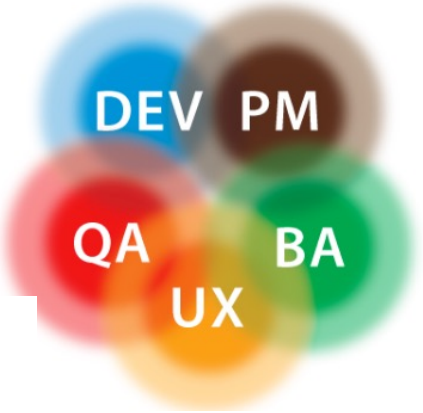


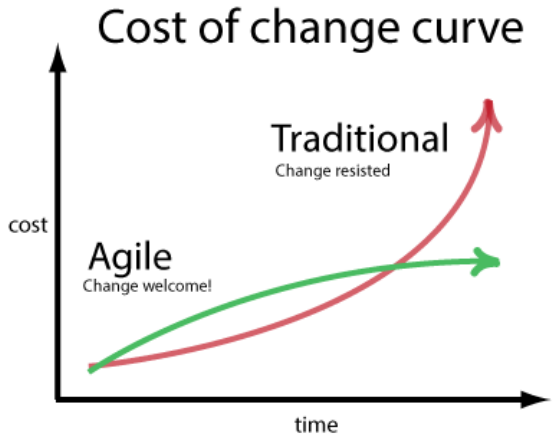
How is Agile different

- Development is iterative
- Planning is adaptive
- Roles blur
- Requirements change
- Working software



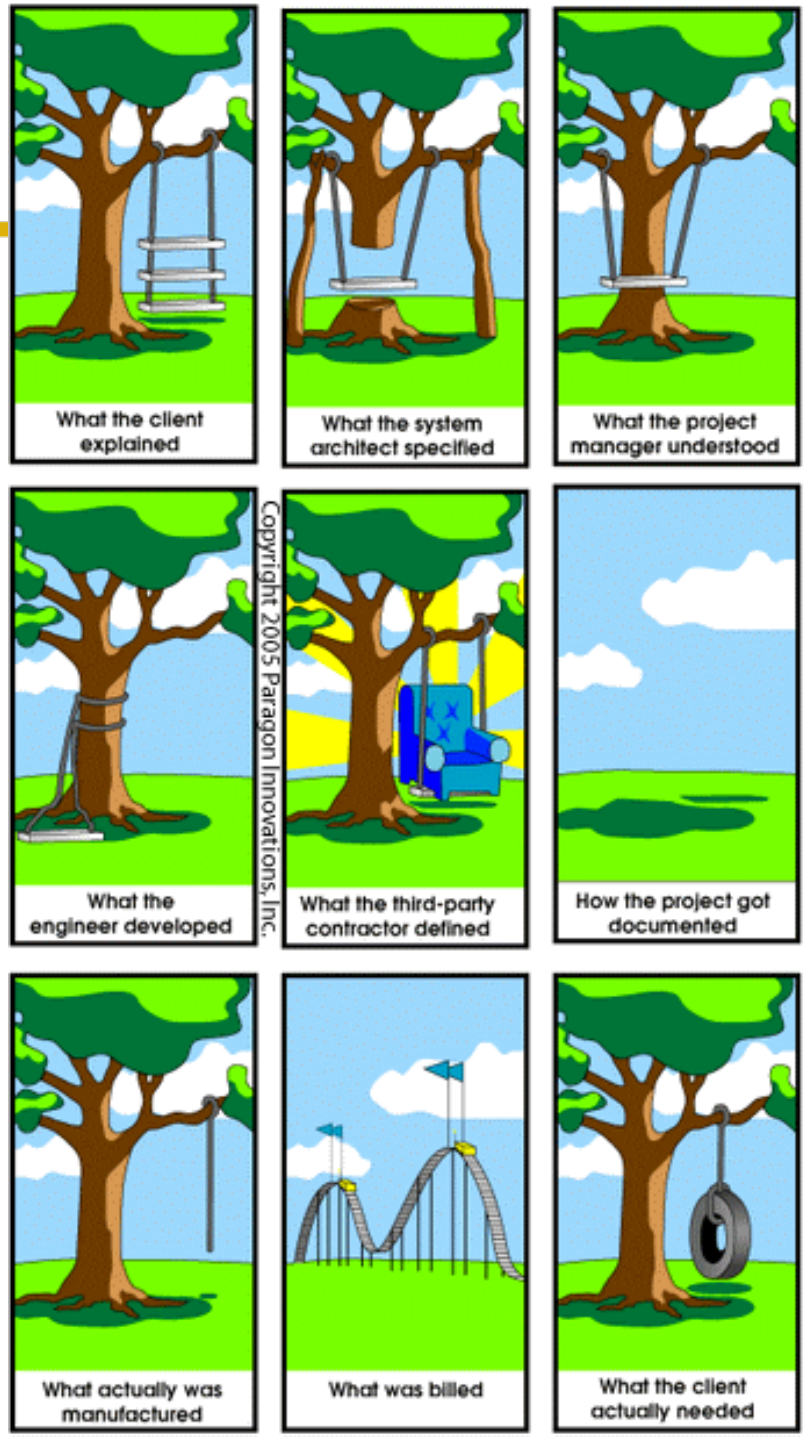






Agile myths

- Agile is a silver bullet
- Agile is anti-documentation
- Agile is anti-planning
- Agile is undisciplined
- Agile requires a lot of rework
- Agile is anti architecture
- Agile doesn’t scale



Fundamental approaches: User stories

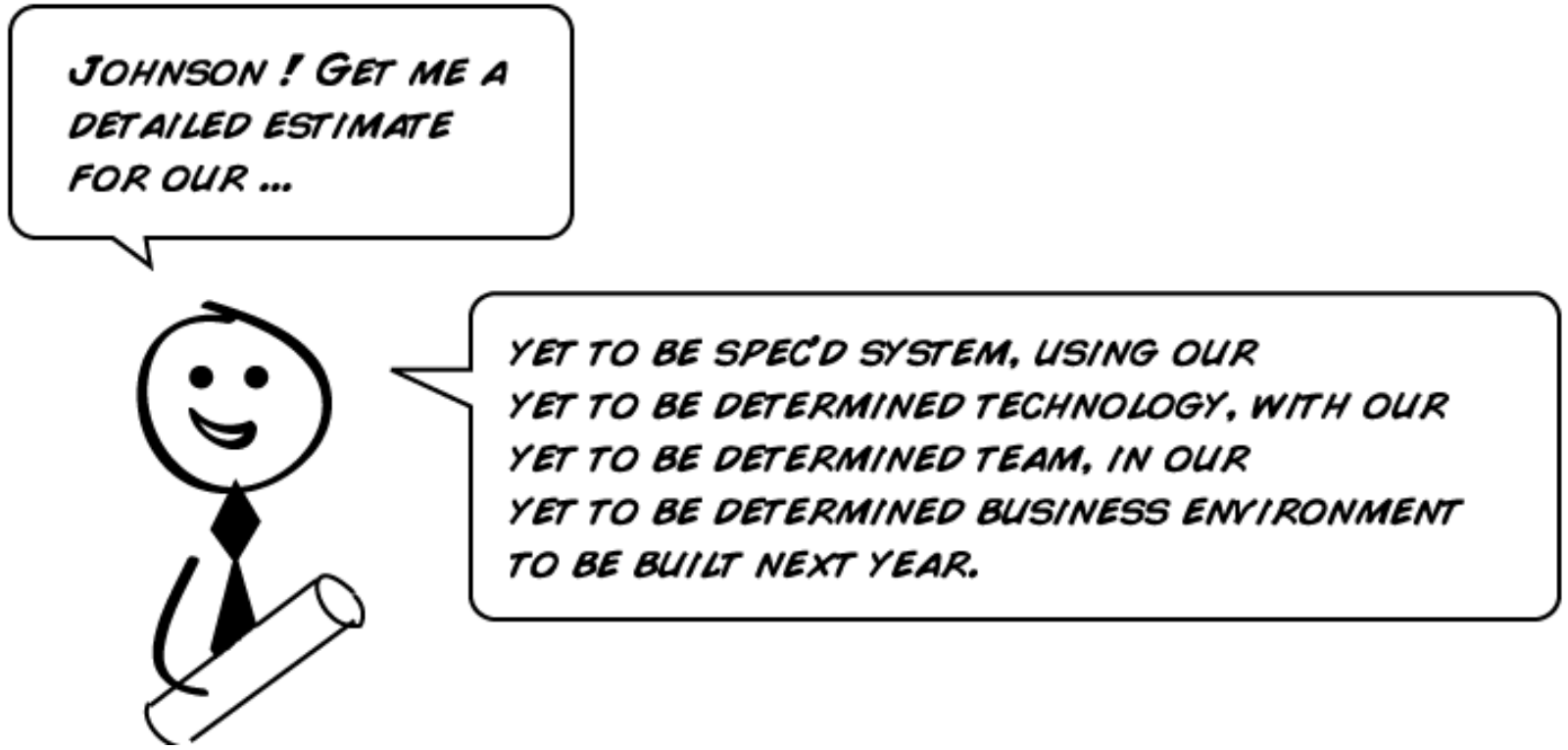
THE UNIVERSITY OF WESTERN AUSTRALIA

- User stories describe features
- They are told from the end user point of view.
- These features can be deliver in short units of work.
- They are often written on cards to facilitate communication

#	Backlog Item (User Story)	Story Point
1.	As a Teller, I want to be able to find clients by last name, so that I can find their profile faster	4
2.	As a System Admin, I want to be able to configure user settings so that I can control access.	2
3.	As a System Admin, I want to be able to add new users when required, so that...	2
4.	As a data entry clerk, I want the system to automatically check my spelling so that...	1

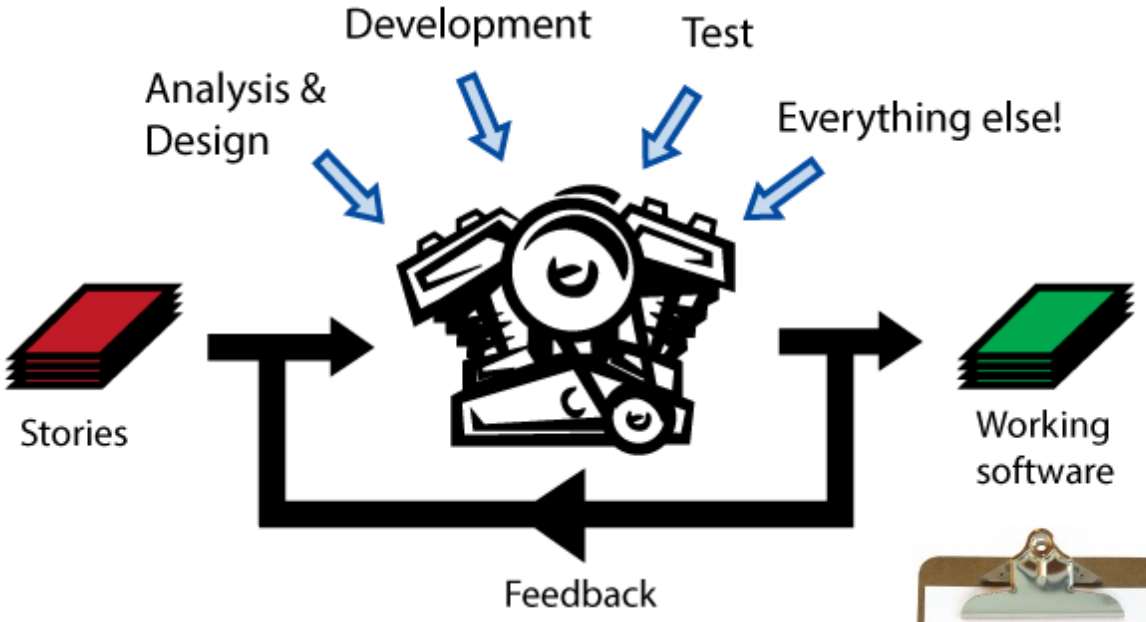
Fundamental approaches: estimation

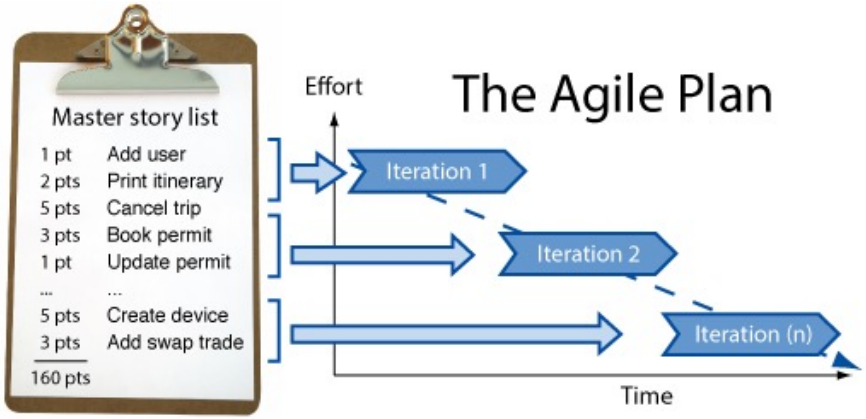
- Estimation is difficult but essential.
- You should always practice estimating the amount of time development will take.



Fundamental Approaches: Iterations

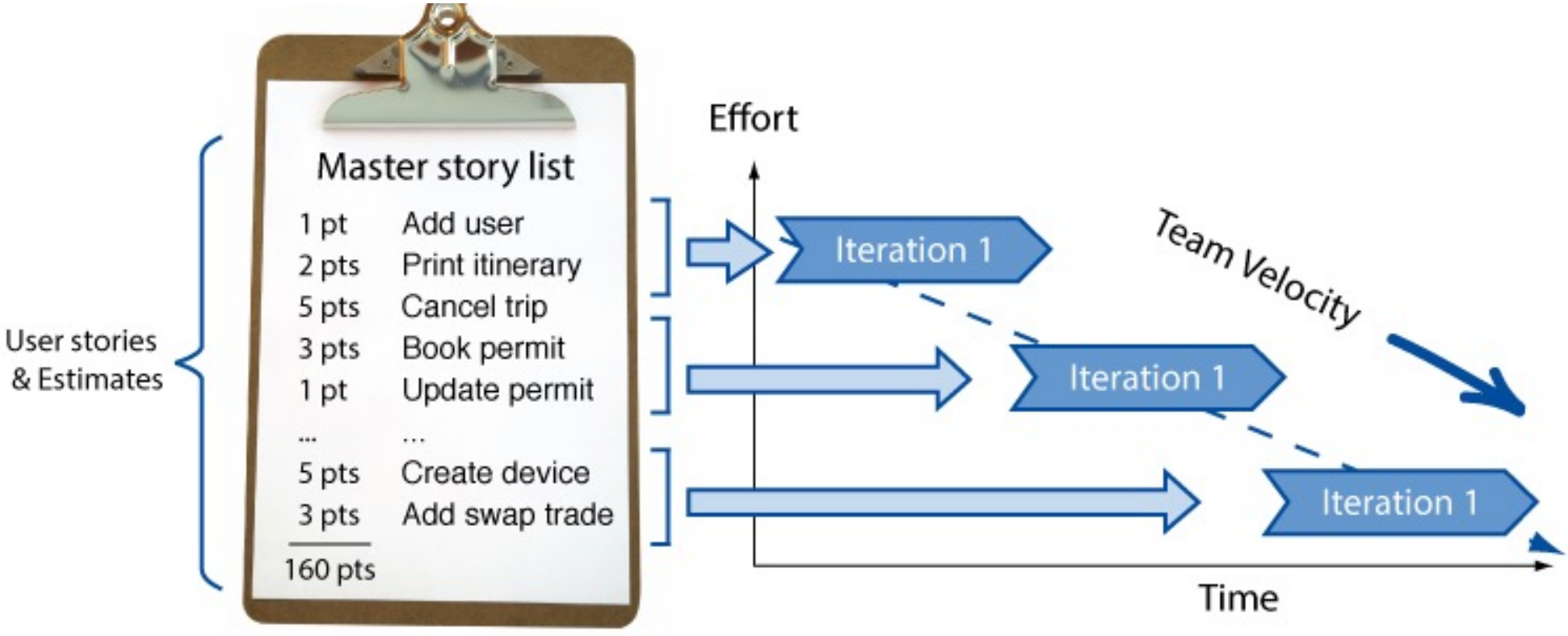
- Iterations are the core of software develpoment.





Fundamental approaches: planning

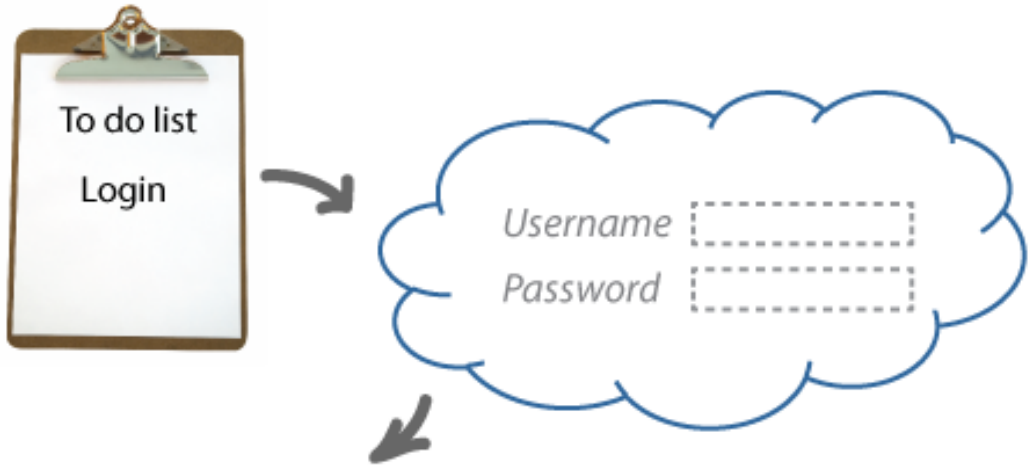
- Combines the user stories and estimations to build a feasible plan for delivery.



Unit Testing

THE UNIVERSITY OF WESTERN AUSTRALIA

Unit tests are snippets of test code developers write to prove to themselves that what they are developing actually works. Think of them as codified requirements.

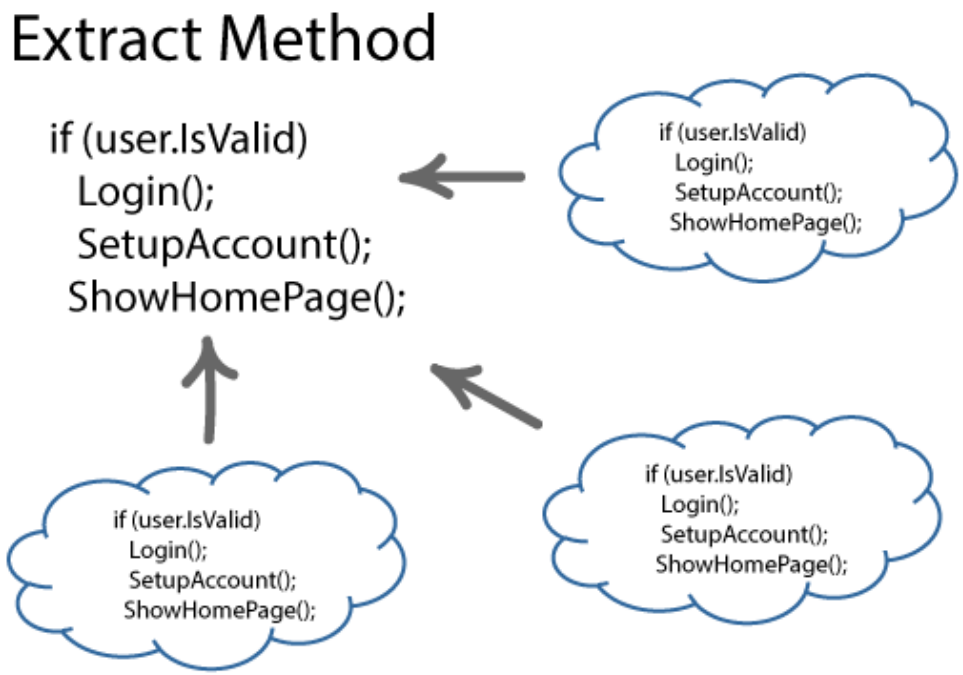


They are powerful because when combined with a continuous integration process they enable us to make changes to our software with confidence.

Refactoring

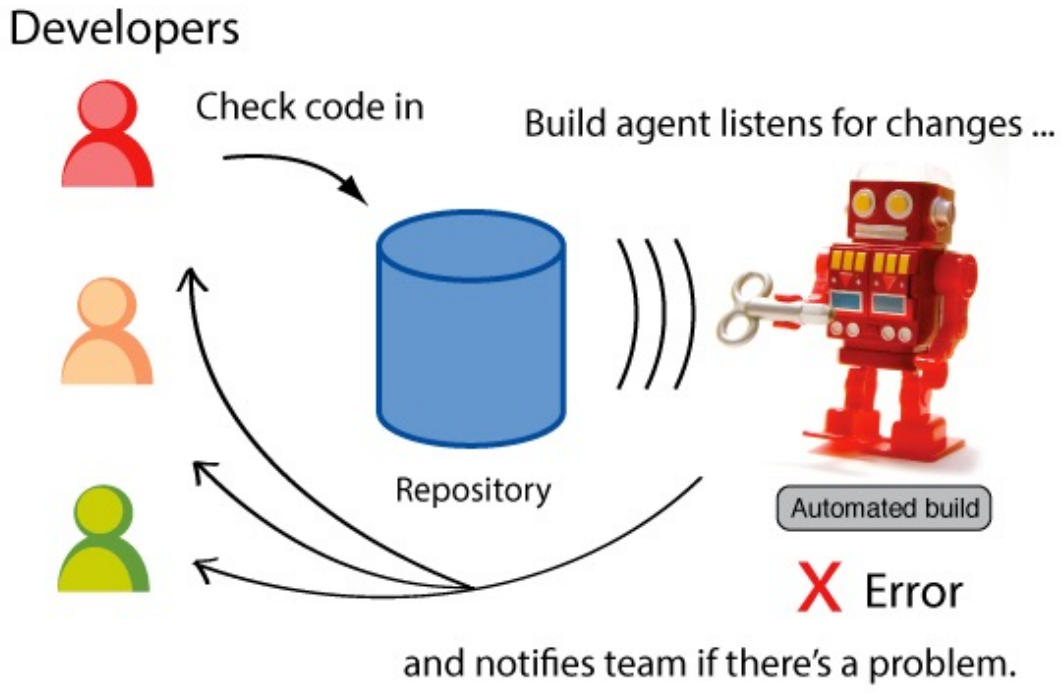
- To maintain a design and functionality, we must be prepared to refactor code.

- Organise code into manageable modules.
- Don’t repeat yourself (DRY)



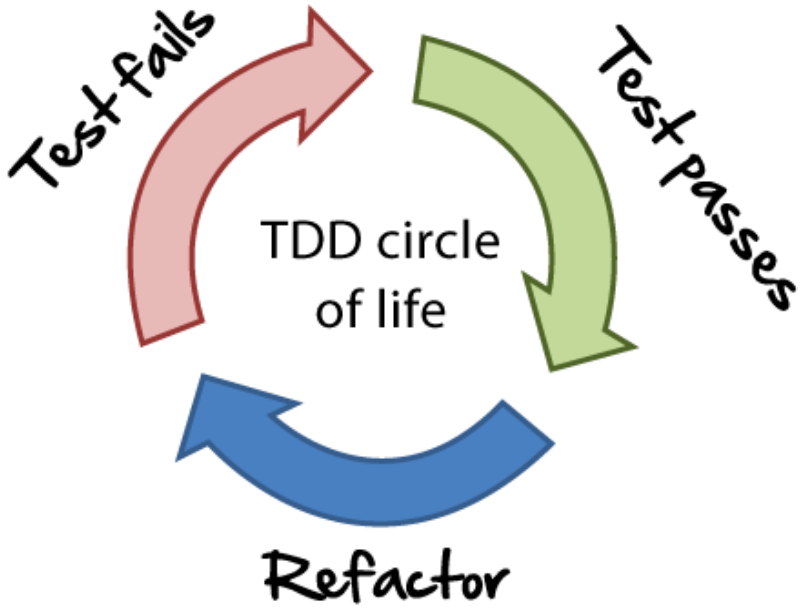
Continuous Integration

- Continuous integration keeps the code in a repository that is automatically maintained and everyone works on at the same time.

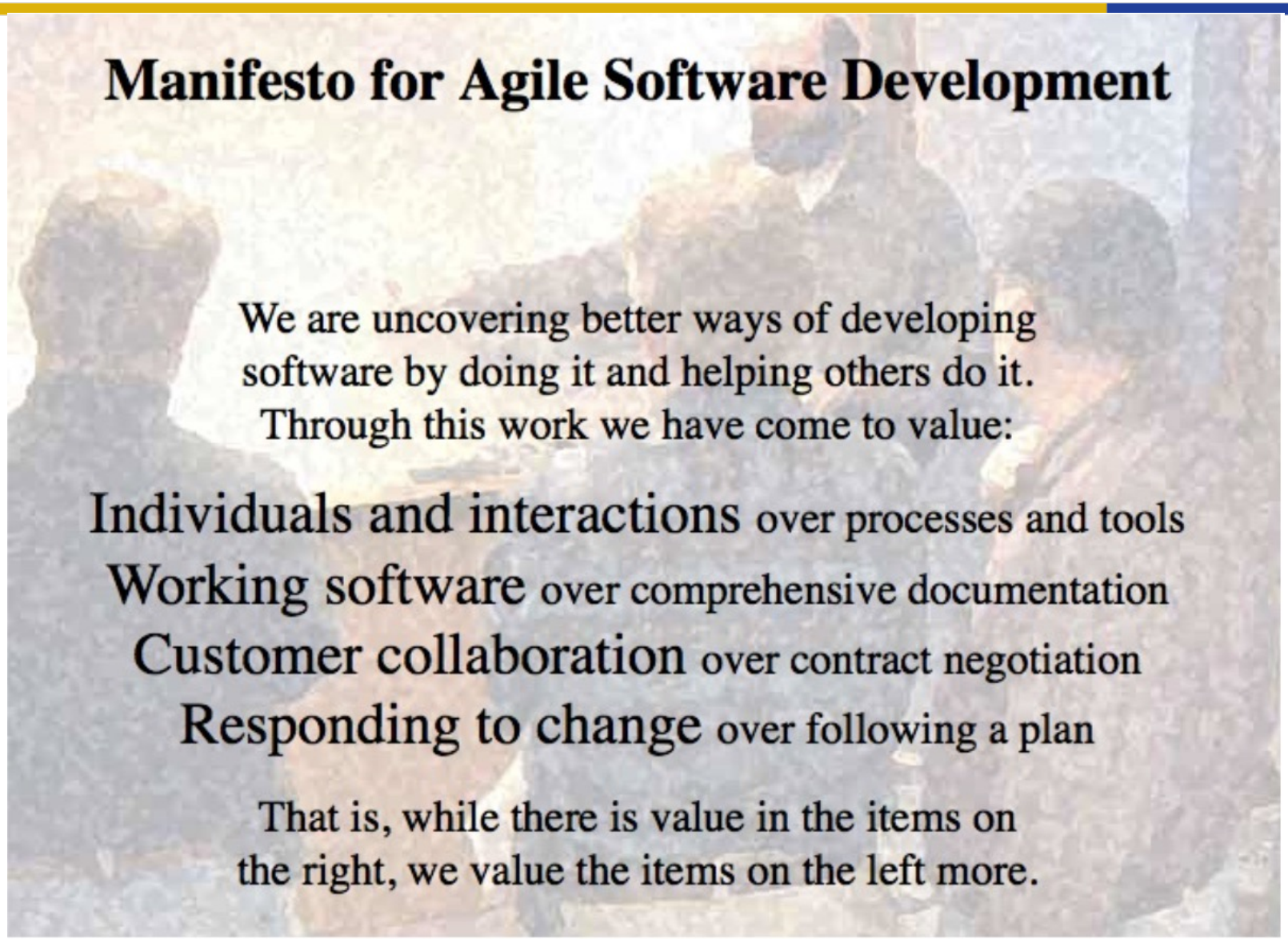


Test Driven Development

- Write tests at the start and then write code to pass the tests.
- The tests become the de facto documentation for the system.



Types of Agile:

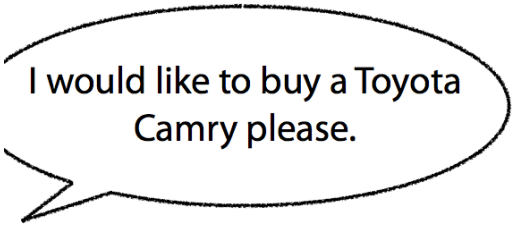


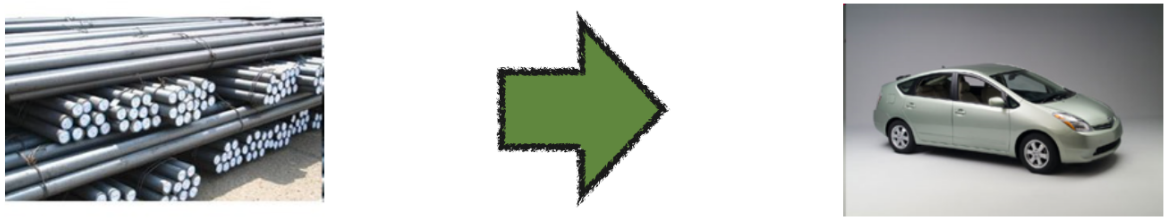
Flavours of Agile

- Good high organisation
- But not IT specific

Lean

Toyota's ultra-lean manufacturing process.



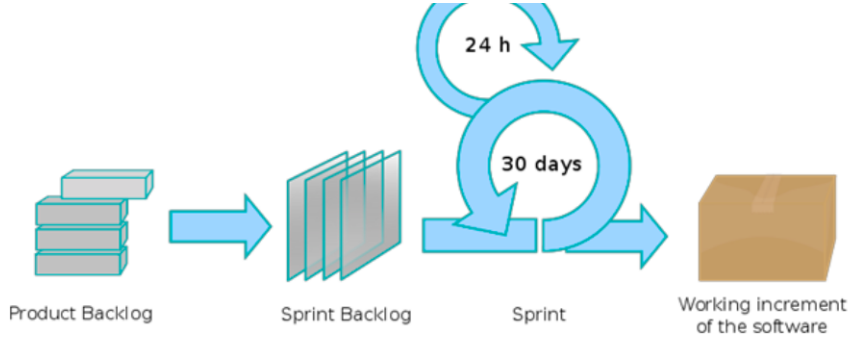


Eliminating waste

Flavours of Agile

- Easy to understand and start
- Very popular
- Not much engineering

Scrum



- A project management wrapper for incremental delivery of projects, independent of technology or business vertical.
- Can be used in non-IT projects.

Flavours of Agile

THE UNIVERSITY OF WESTERN AUSTRALIA

- Detailed engineering practices
- IT focused
- Popular with developers

Extreme Programming

- Popularized software engineering practices necessary for agile development
- Emphasizes
  - –upfront testing
  - –automation
  - –evolutionary design
  - –continuous integration

Message to Agile Web Dev Class, 2022:

THE UNIVERSITY OF WESTERN AUSTRALIA

It's really important to establish a common expectation among the group. If everyone wants to do it the night before it's due, that's not ideal, but its ok as long as everyone's on board. If you can't establish common expectations then it's possible to break the group up, but that's really a last resort.

Some ways to handle this:

1. *Manage expectations*. It's probably a better strategy to set modest goals that the entire group can contribute to, rather than expect people to suddenly become elite coders.
2. *Keep a log of meetings/chats and responsibilities*. If things fall apart, and you have been trying, this will salvage your marks at least.
3. *Break the project up into small units*, to work on individually, and make sure that you deliver your allocation, and then try to help others with theirs.
4. *Share your knowledge*. It's not your responsibility to teach others, but it is part of the agile methodology. Your team are an asset and a resource that need to be optimally leveraged. Find achievable tasks for everyone, and a short instructional session early can be a big benefit later on.

There will be an opportunity during the demonstrations to discuss who did what, and what the challenges were, and marks among the group may be split accordingly. Some people are happy to admit they were passengers and take a hit in the mark. Agile development is about dealing with adversity and adjusting, not avoid adversity all together.

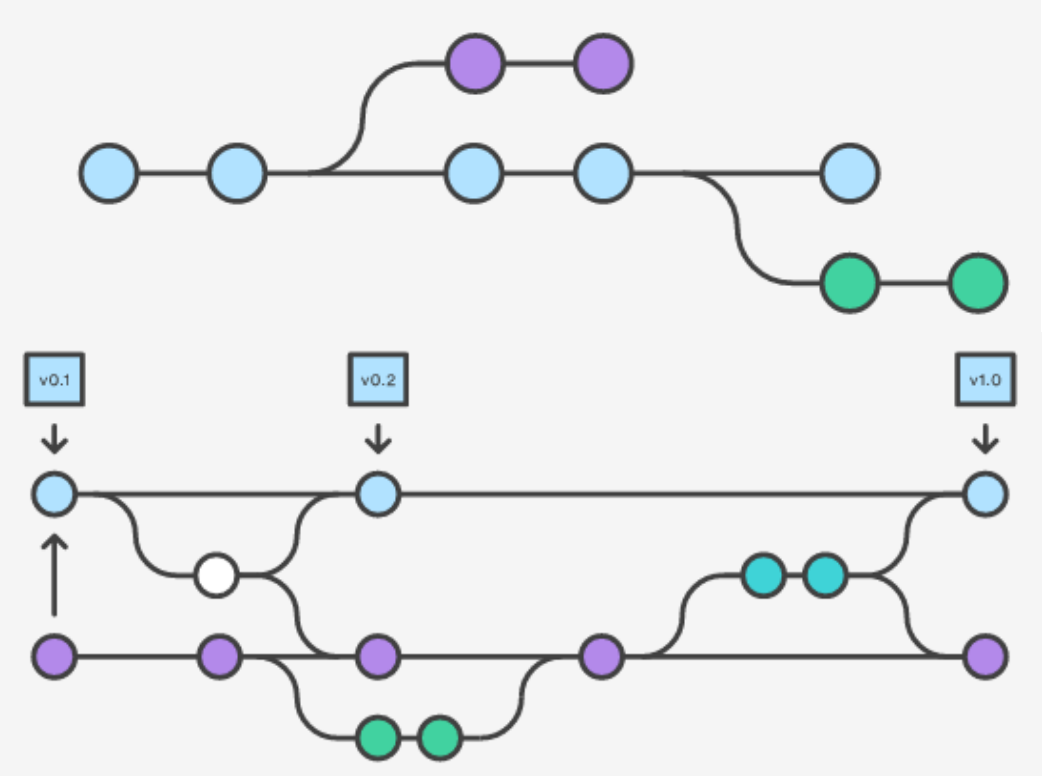
GIT

THE UNIVERSITY OF WESTERN AUSTRALIA

- Git is a distributed version control system
- Developed in 2005 by Linus Torvalds
- Now the most widely used version control system in the world.
- Git is able to manage different branches of a development, allowing teams to work on the latest branch, roll back changes, or develop independent features.

Git Theory

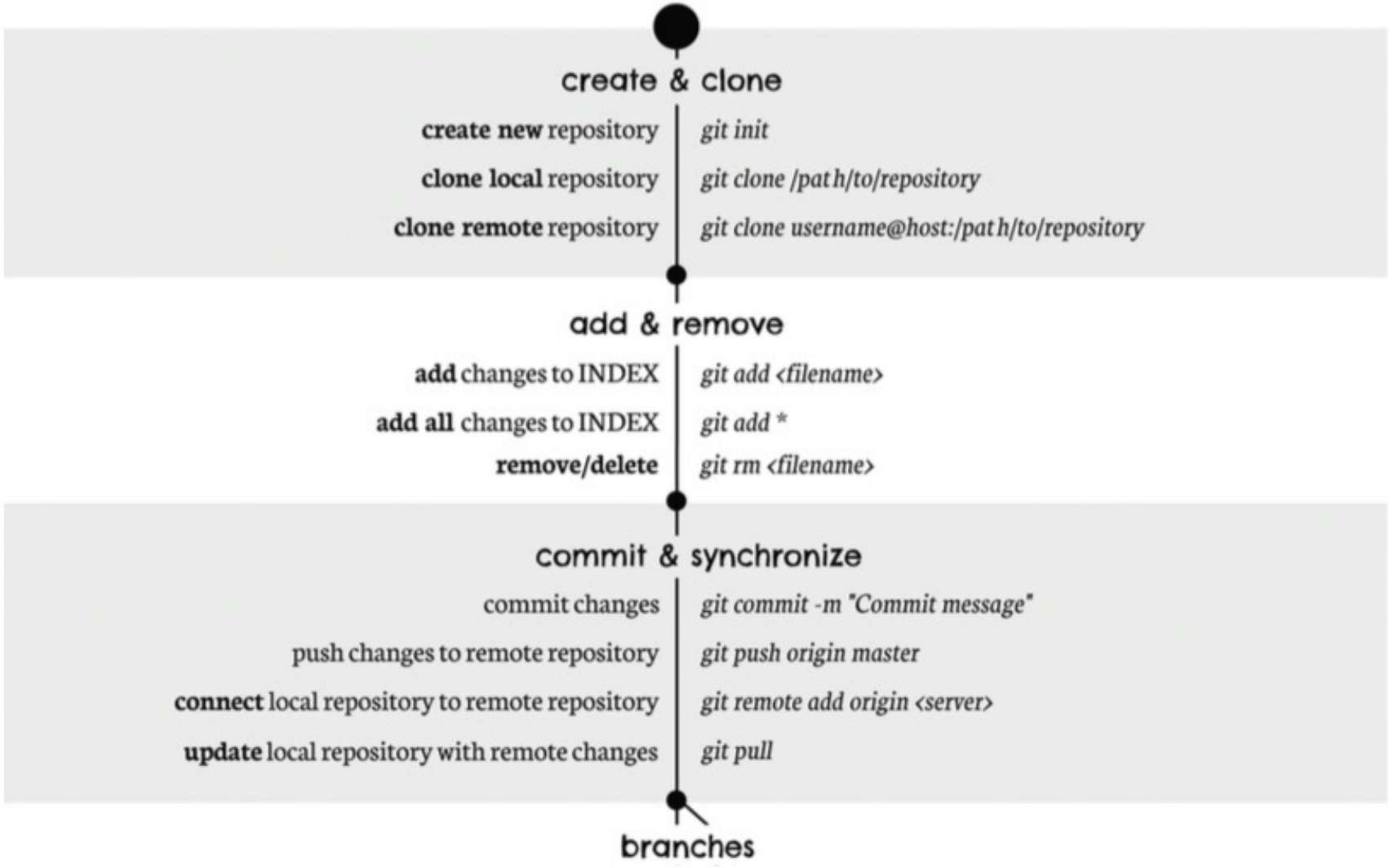
- Git has a decentralised structure. Everyone on the project has a copy of the history of the project.
- The history of the project is structured as a graph. Each commit can be undone and replayed
- Git tracks changes in the current working directory.
- Changes are *added*, then *committed*, then *pushed* to a branch.
- The new code is then *pulled* to other spaces.



Git CheatSheet

THE UNIVERSITY OF WESTERN AUSTRALIA

learn more about git the simple way at *rogerdudler.github.com/git-guide/*
cheat sheet created by Nina Jaeschke of *ninagrafik.com*



GIT CheatSheet

THE UNIVERSITY OF WESTERN AUSTRALIA

branches

create new branch	*git checkout -b <branch>* *e.g. git checkout -b feature_x*
switch to master branch	*git checkout master*
delete branch	*git branch -d <branch>*
push branch to remote repository	*git push origin <branch>*

merge

merge changes from another branch	*git merge <branch>*
view changes between two branches	*git diff <source_branch> <target_branch>* *e.g. git diff feature_x feature_y*

tagging

create tag	*git tag <tag> <commit ID>* *e.g. git tag 1.0.0 1b2e1d63ff*
get commit IDs	*git log*

restore

replace working copy with latest from HEAD	*git checkout -- <filename>*

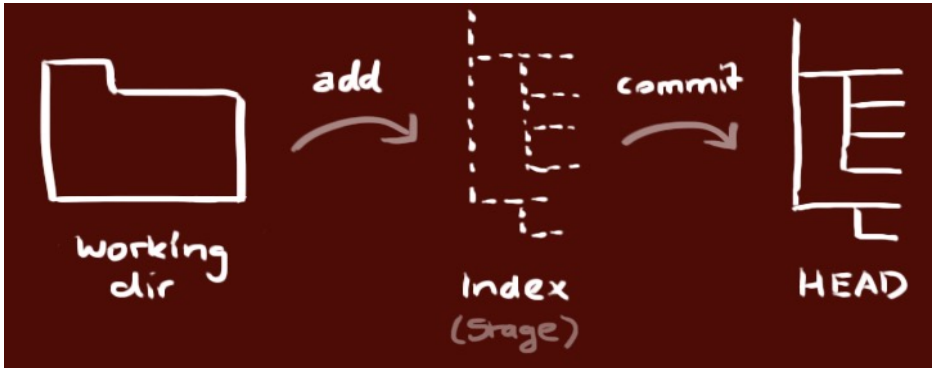
Tip

Want a simple but powerful git-client for your mac?

Try Tower: www.git-tower.com/

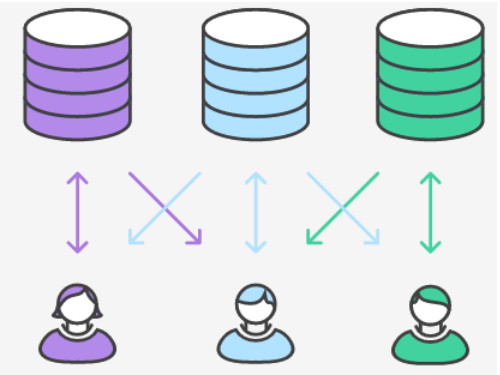
GIT operations

- Create a new repository: `git init`
  - adds files monitoring changes, preferences etc.
- Checkout a repo: `git clone usr@url:[path]`
  - copies files and history to a local copy
- The local repo has three trees:
  - the *working directory* (the actual files)
  - the *index* (a set of changes that is ready to commit)
  - the *head* (the last commit you made)



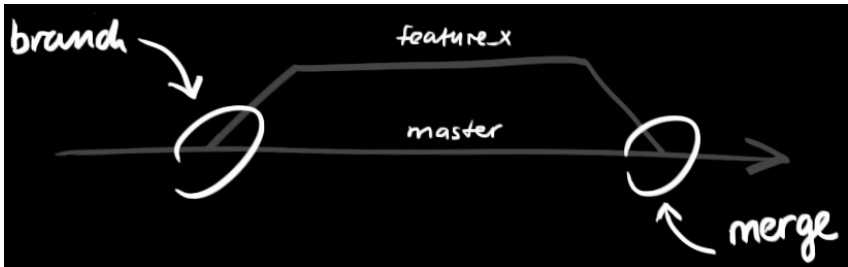
GIT add, commit, push, pull

- Add new files: `git add <filename>`
  - adds file to index
- Commit changes to Head: `git commit –m "msg"`
  - commits changes in index to the head with the msg.
- Push changes to repository:
  - `git remote add origin <server>`
  - `git push origin master`
  - adds remote server, and pushes latest commit to it.
- Get latest commit from repo: `git pull`



GIT branching

- Create a new branch: `git checkout -b b1`
  - creates a new branch that can change independently.
- Switch back to master: `git checkout master`
  - changes to the master branch.
- Delete Branch: `git branch -d b1`
- Push new branch: `git push origin <branch>`
- Merge into your branch: `git merge <branch>`
  - merging is automatic, but there may be conflicts.
- To see the differences between two branches:
  `git diff <b1> <b2>`



GIT utilities

- Get a log of commits: `git log`
- Tag a version: `git tag 1.0.0 <commit-id>`
  - gives a version number to a commit tag.
- Rollback changes: `git checkout -- <file>`
  - returns local file to last commit.
- Delete Branch: `git branch -d b1`
- Undo all changes and commits: `git fetch origin`
  - then `git reset -hard origin/master`

- Lots of GUIs, environments exist: GitHub, BitBucket, GitKraken

GitHub

THE UNIVERSITY OF WESTERN AUSTRALIA

- GitHub is a service that hosts Git repositories.
- You can develop colaboratively, and use GitHub as a remote Repo.
- **Note:** Free GitHub can be public or private. IF they are public, anyone can see your code.
- Students are able to get free education accounts, which allow private repositories.
  - https://education.github.com/pack

- Bitbucket is a similar service, from Atlassian

Coordinating Dev Environments

THE UNIVERSITY OF WESTERN AUSTRALIA

- Git is a great way to link development and deployment environments:
  - Work on your local machine (a laptop) with all the latest features and branches
  - Push commits to GitHub or some central repo
  - Pull changes from GitHub to a server environment for test or deployment
  - You can incorporate testing, documenting and reporting into these workflows.

Git and the project

- Git should be used to manage your project and you git repository will need to be submitted.
- You will be assessed on your commit history, and the readme should contain a report of your work in *markdown* (a lightweight subset of HTML).

Markdown is a way to style text on the web. You control the display of the document; formatting words as bold or italic, adding images, and creating lists are just a few of the things we can do with Markdown. Mostly, Markdown is just regular text with a few non-alphabetic characters thrown in, like # or *.

HEADERS

```
# This is an <h1> tag
## This is an <h2> tag
###### This is an <h6> tag
```

EMPHASIS

```
*This text will be italic*
_This will also be italic_

**This text will be bold**
__This will also be bold__

*You **can** combine them*
```

LISTS

Unordered

```
* Item 1
* Item 2
  * Item 2a
  * Item 2b
```

Ordered

```
1. Item 1
2. Item 2
3. Item 3
   * Item 3a
   * Item 3b
```

IMAGES

```
![GitHub Logo](/images/logo.png)

Format: ![Alt Text](url)
```

LINKS

```
http://github.com - automatic!

[GitHub](http://github.com)
```

BLOCKQUOTES

```
As Grace Hopper said:

> I've always been more interested
> in the future than in the past.
```

As Grace Hopper said:

> I've always been more interested in the future than in the past.

BACKSLASH ESCAPES

Markdown allows you to use backslash escapes to generate literal characters which would otherwise have special meaning in Markdown's formatting syntax.

```
\*literal asterisks\*
```

literal asterisks

Markdown provides backslash escapes for the following characters:

\ backslash	() parentheses
` backtick	# hash mark
* asterisk	+ plus sign
_ underscore	- minus sign (hyphen)
{} curly braces	. dot
[] square brackets	! exclamation mark

Hanabi AI for the unit CITS3001 at UWA

Manage topics

41 commits | 2 branches | 0 releases | 4 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or down

drtnf Merge branch 'master' of https://github.com/drtnf/hanabAI ... Latest commit 3bc4113 on Oct 19

src	Merge branch 'master' of https://github.com/drtnf/hanabAI	5 month
.gitignore	ignore documentation directory	6 month
LICENSE	initial commit	7 month
README.md	fix markdown issues	7 month

README.md

HanabAI

AI platform for the card game Hanabi.

Description

This projects aims to provide a set of Java classes and interface to facilliate agents playing the card game: Hanabi. This project is designed for use in the unit CITS3001 *Algorithms, Agents and Artificial Intelligence* at The University of Western Australia. Students may use and modify these files to reserach different artificial intelligence agents in the game Hanabi. There will also be an AI bot tournament with the additional rules specified below.

Rules

Rules - Hanabi