Agile Software Development

Dr Jon Bird
jon.bird@bristol.ac.uk

Thanks to Dr Simon Lock who developed many of these slides for an earlier version of this unit.

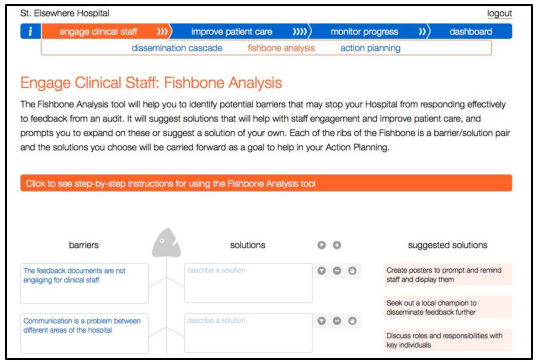
Images are royalty free from www.pexels.com

Today's Lecture

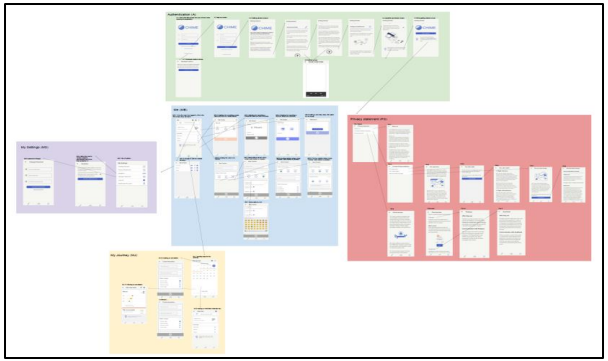
- A digital health project developed using the waterfall approach, which was the focus of last week
- Waterfall versus agile approaches to software development
- Agile software development, including: extreme programming; test-driven development; scrum; and Kanban
- A digital health project developed using an agile approach
- Recommended reading
- A reminder before this afternoon's workshop

Digital health projects

- My research involves developing digital technologies to address health issues
- What software development process do I employ in my research?



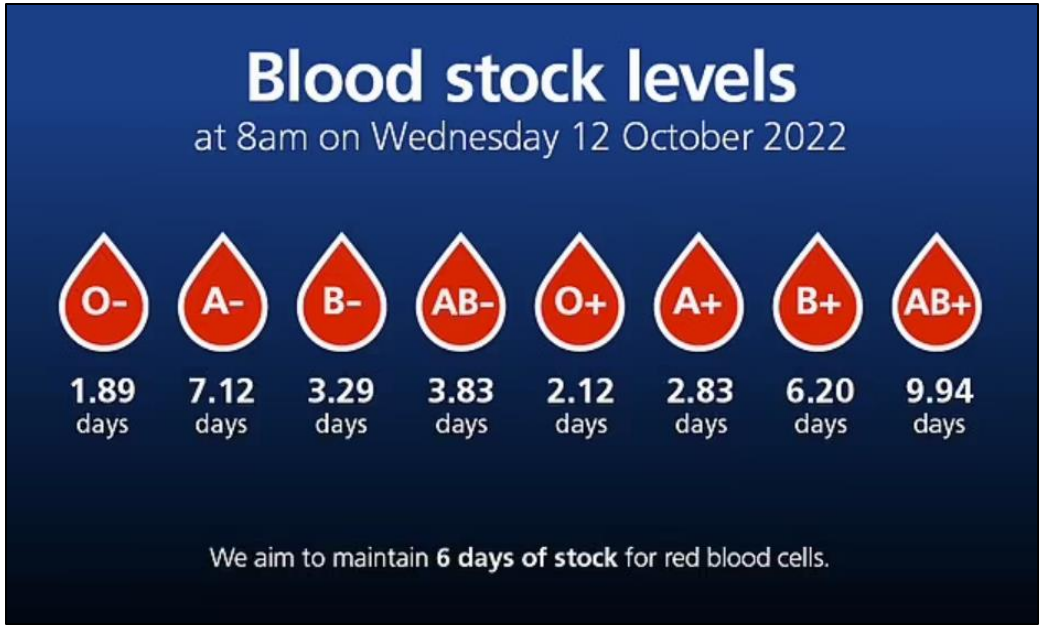
Affinitie



CHIME

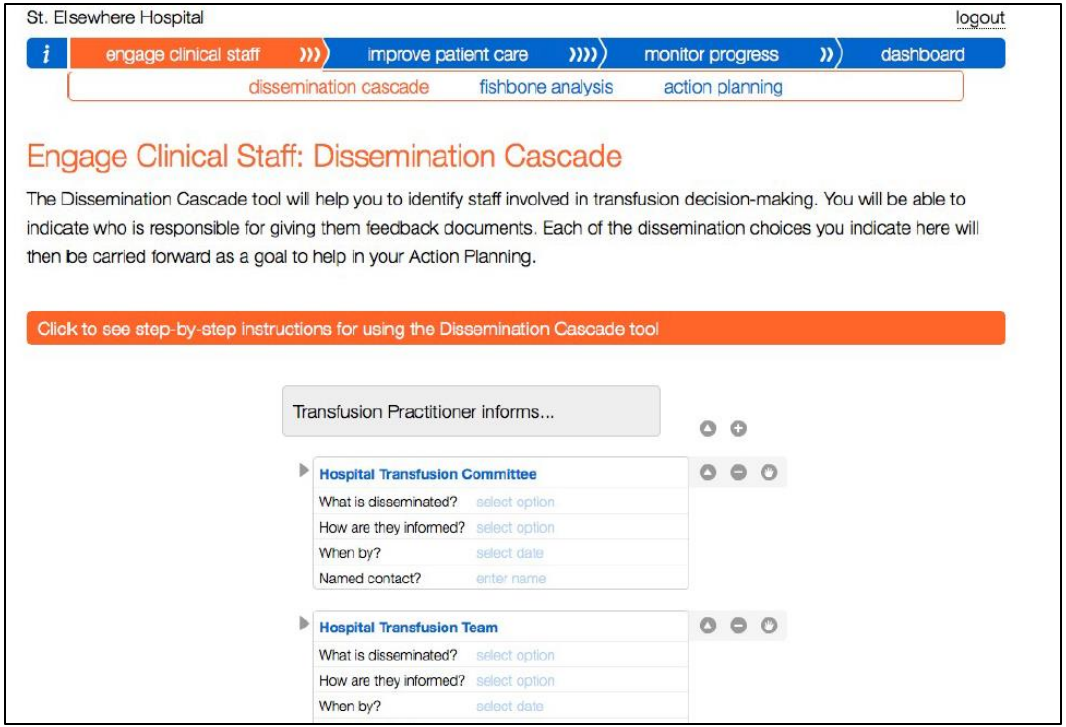
Affinitie - motivation

- There are 3 million blood transfusions annually in the UK
- Around 20% are unnecessary
- Blood components are a scarce resource
- There are health risks associated with blood transfusions
- How can we reduce unnecessary blood transfusions?



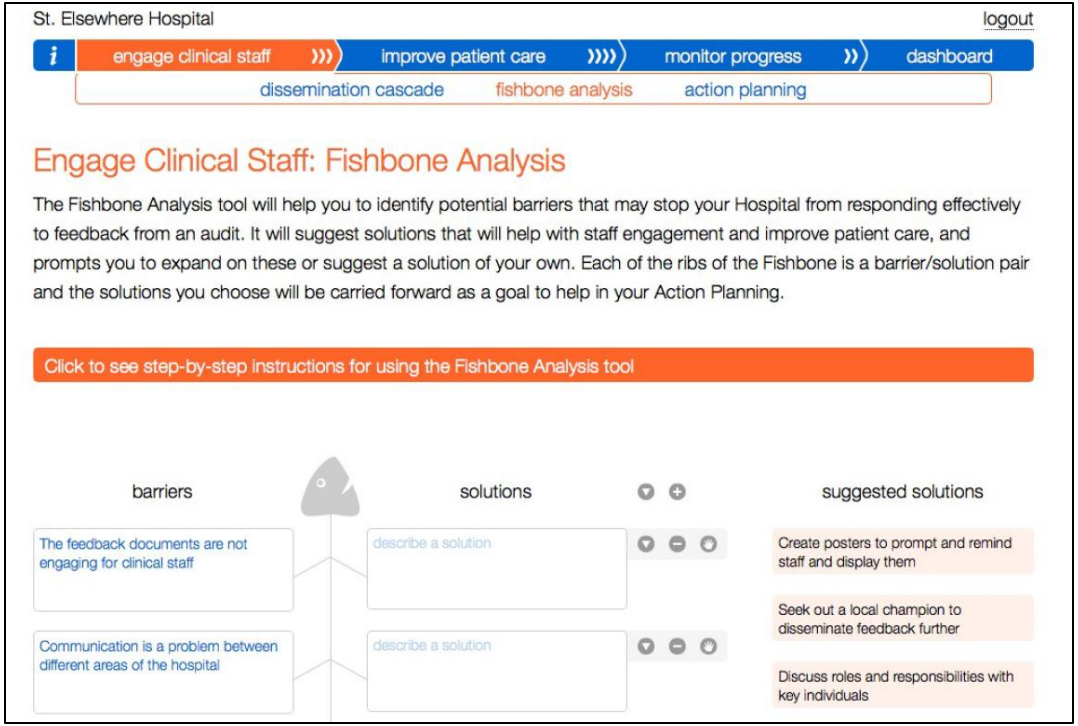
Affinitie – software

- I was asked to join an existing research project to develop a web toolkit for transfusion practitioners
- The research team consisted of health psychologists, statisticians, doctors and NHS Blood and Transplant
- The aim was to help transfusion practitioners disseminate blood transfusion audit results to everyone in the hospital

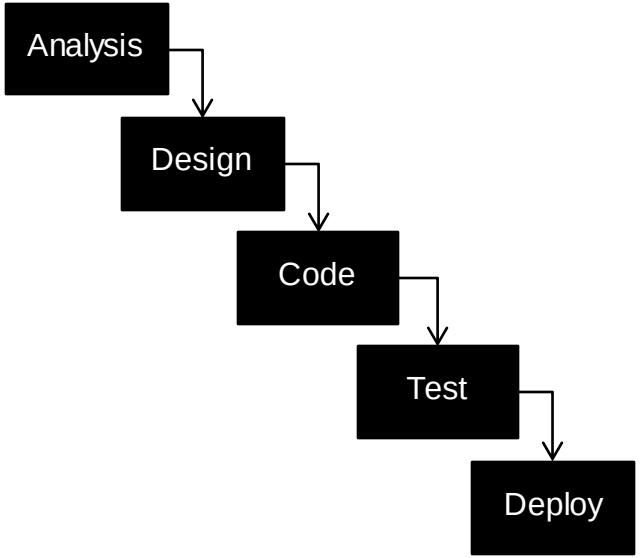


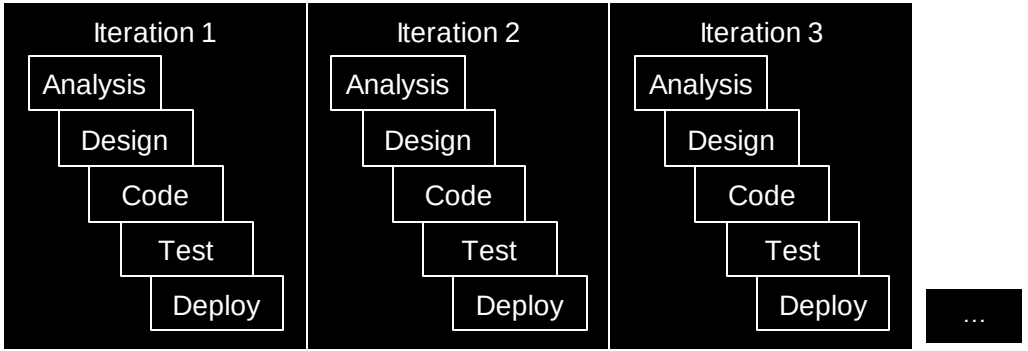
Affinitie – software development

- We were given **a clear set of requirements** by the research team
- They had already developed a set of paper-based tools for transfusion practitioners to help them disseminate blood transfusion audit results
- Software development approach: **waterfall**



Waterfall versus agile life cycles





What is Agile Software Development?

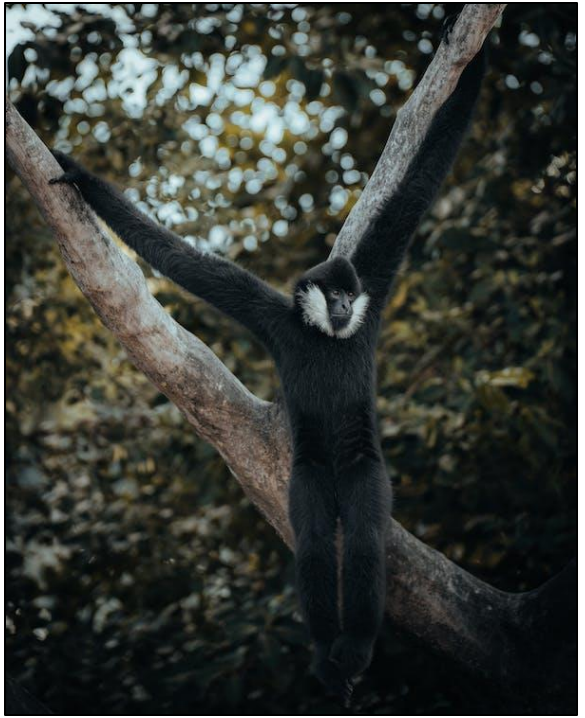
- Agile is a way of thinking about software development
- In winter 2001, 17 software developers met at a ski resort in Utah and drafted a manifesto outlining an alternative way to develop software to the documentation-driven software development processes of the time
- The manifesto is succinct and puts forward four key values for software development

The Manifesto for Agile Software Development

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

That is, while there is value in the items on the right, we value the items on the left more



How should you read the manifesto?

- The values are purposefully provocative in order to get people to think about software development
- The Agile Manifesto is **not** proposing that we disregard aspects of software development like processes, tools and documentation
- Rather, the Agile Manifesto wants people to think about alternative ways of doing aspects of software development

Agile was created by coders for coders

Coders like
Writing quality code
Ticking things off their to do list
Impressing clients by showing them working software

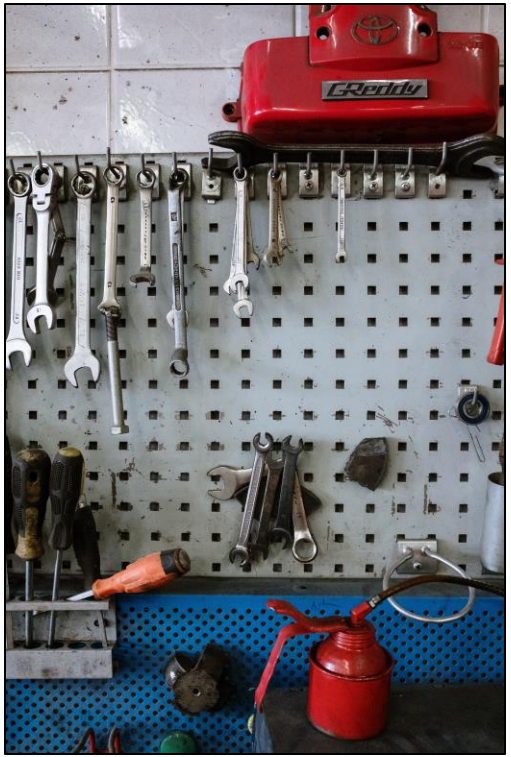
Coders dislike
Writing extensive documentation
Committing to a final design in advance
Being micromanaged
Working to big, immovable deadlines

Twelve agile principles

Satisfy clients' needs	Satisfy coders' needs
The highest priority is satisfying the client (by delivering working software early and continuously)	Work at a steady, sustainable pace (no heroic efforts)
Embrace change (even late in the development cycle)	Rely on self-organising teams
Collaborate every day with the client	Teams reflect regularly on their performance
Use face to face communication	Progress is measured by the amount of working code produced
Deliver working software frequently	Continuous attention to technical excellence
	Minimise the amount of unnecessary work
	Build teams around motivated individuals

Agile Methods

- There are various approaches that adhere to Agile values and principles
- Different companies choose different approaches
- Popular methods include:
  - Extreme Programming (XP) (the two co-creators were signatories of the manifesto)
  - Test-driven development (creator was a signatory)
  - Kanban
  - Scrum (the two co-creators were signatories)
- We'll introduce some key practices from each of these methods that we think you will be useful in this unit and your summer projects
- There are links in the reading to some of these methods



Hands up if any of these apply to you

- Your code structure is complex and “sophisticated”
- You work primarily on your own
- In group projects you are responsible for just your own code
- You write code in your own style
- You work some weekends and so some “all-nighters”
- You code develops in “heroic bursts”

Extreme Programming Ethos

- **Simple design**: use the simplest way to implement features
- **Sustainable pace**: effort is constant and manageable
- **Coding standards**: teams follow an agreed style and format
- **Collective ownership**: everyone owns all the code
- **Whole team approach**: everyone is included in everything

Extreme Programming Practices

- **Pair programming**: two heads are better than one
- **Test driven**: ensure the code runs correctly
- **Small releases**: deliver frequently and get feedback from the client
- **Continuous integration**: ensure the system is operational
- **Refactor**: restructure the system when things get messy

Pair programming in more detail

Code is written by two programmers on one machine:

- The **helm** uses the keyboard and mouse and does the coding
- The **tactician** thinks about implications and potential problems

- Communication is essential for pair programming to work
- Pair programming facilitates project communication

- The pair doesn't "own" that code - anyone can change it
- Pairings can (and should) evolve at any time

- All code is reviewed as it is written
- The **tactician** is ideally positioned to recommend refactoring

The impact of pair programming

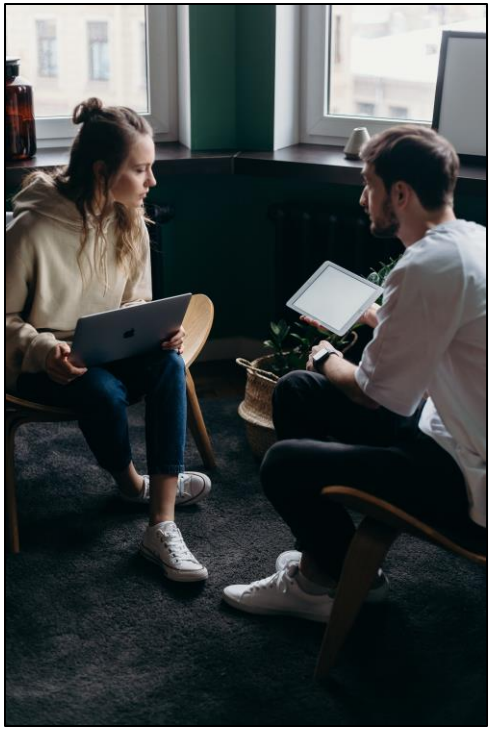
Research studies have assessed the impact of pair programming and identified a number of benefits

Single programmer 77 source lines per month versus pair programming 175 source lines per month

[Jensen, 2005]

15% increase in development-time costs but improves design quality, reduces defects, reduces staffing risk, enhances technical skills, improves team communications and is considered more enjoyable at statistically significant levels.

[Cockburn & Williams, 2000]



Test-driven development in more detail

- Tests are written before any code and they drive all development
- A programmer's job is to write code to pass the tests
- If there's no test for a feature, then it is not implemented
- Tests are the requirements of the system

The benefits of test-driven development

- Code coverage

We can be sure that all code written has at least one test because if there were no test, the code wouldn't exist

- Simplified debugging

If a test fails, then we know it must have been caused by the last change

- System documentation

Tests themselves are one form of documentation because they describe what the code should be doing

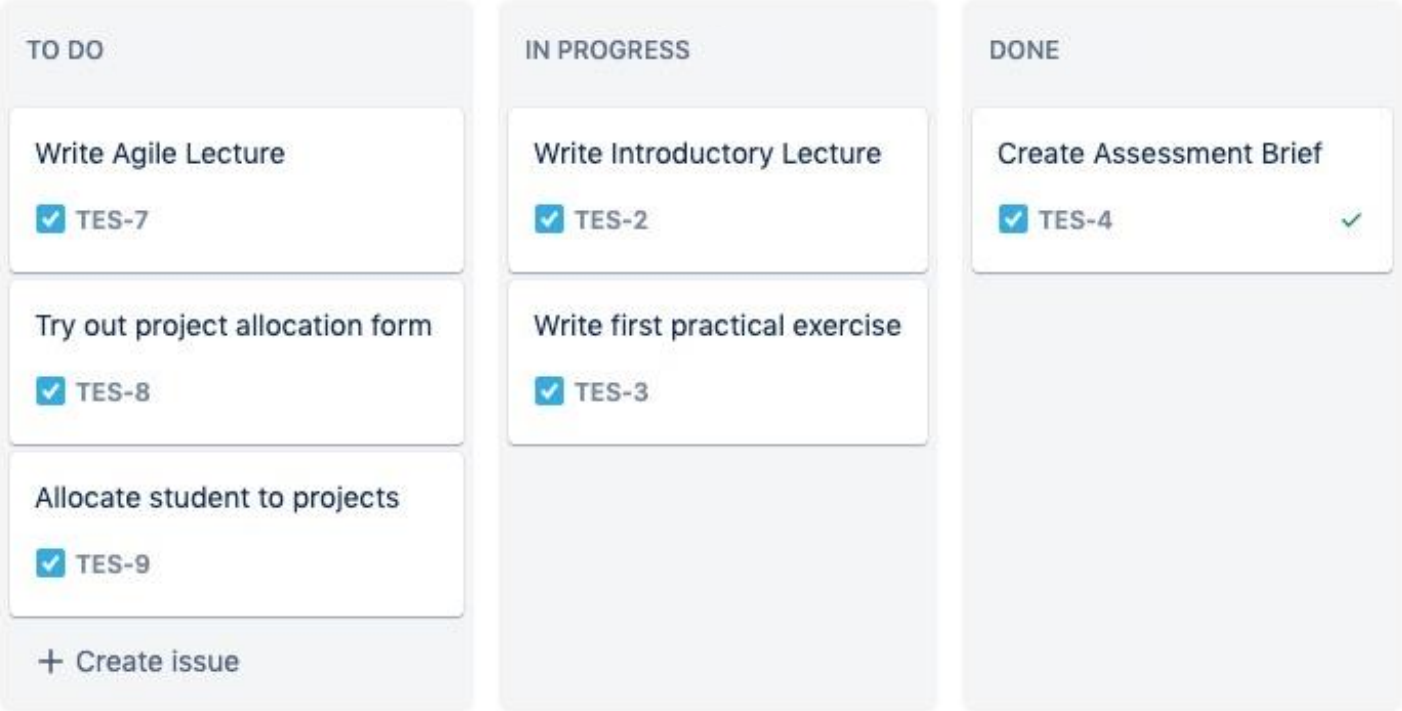
Scrum

- Scrum is a project management approach
- Some key concepts are:
- **The Scrum** – a **stand-up** daily meeting of the entire team
- **Scrum Master** - team Leader
- **Sprint -** a short, rapid development iteration
- **Product Backlog** – To do list of jobs that need doing
- **Product Owner** – the client (or their representative)

Kanban Board

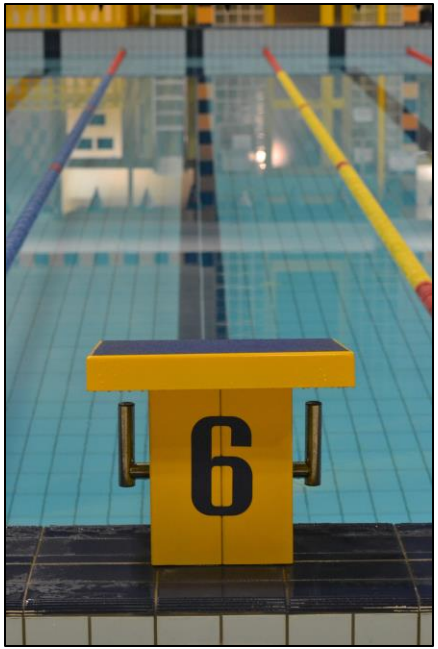
- A concept taken from the Kanban method which was first defined in 2007 but came out of a scheduling system developed by Toyota in the 1950s for just-in-time manufacturing
- In Japanese “kanban” means “visual board” or “sign”
- It’s basically a flexible “to do” list tool
- Issues progress through various states from ”To do” to “Done”
- It was originally implemented as post-it notes on a whiteboard
- Various digital tools now fulfil the same function e.g. Jira

Jira Kanban Board



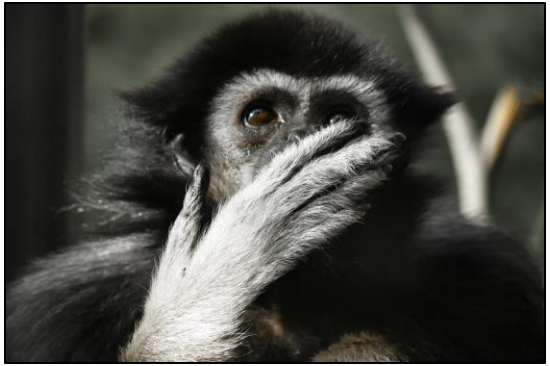
Columns (Swim lanes)

- It is common to use three columns
- But Jira allows you to customize the layout
- For example, you might have columns for:
  - Backlog
  - Being Verified
  - Awaiting integration
- Do what works for your team but make sure you have a “Done” column



Problems with Agile

- Hard to draw up legally binding contracts - a full specification is never written in advance
- Good for green-field development when you have a clean slate and are not constrained by previous work. However, it's not so effective for brownfield development which involves improving and maintaining legacy systems.
- Works well for small co-located teams, but what about large distributed development ?
- Relies on the knowledge of developers in the team but what if they aren't around (holidays, illness, turnover) ?



CHIME - motivation

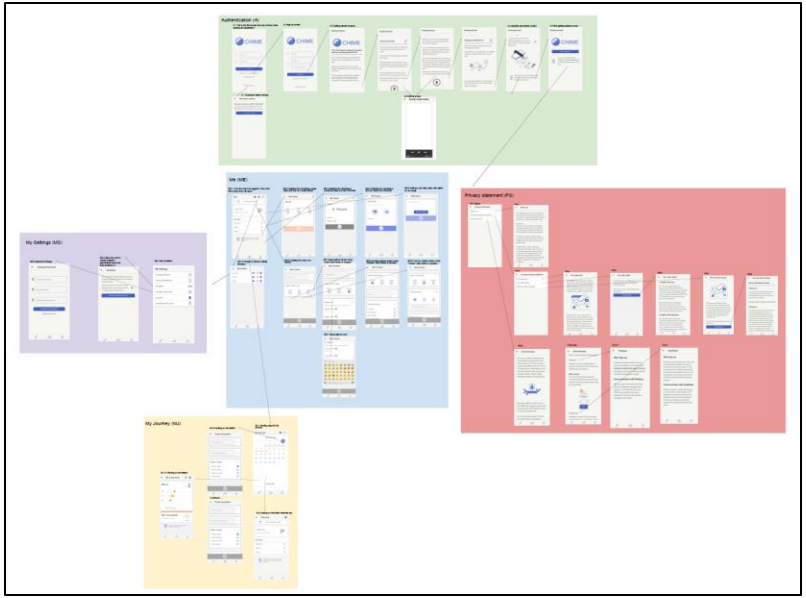
- Tracking health and lifestyle data can help people manage long term conditions
- Sharing these data with healthcare professionals can improve clinical decision making

CHIME - software

- Chime is an app designed for people living with HIV (PLHIV)
- It was developed by researchers at a number of UK universities in collaboration with the Terrence Higgins Trust, the leading HIV charity in the UK

CHIME – software development

- I project manged two programmers who developed the app code at UoB
- We initially designed and built a prototype app based on the requirements identified by other researchers on the project
- The app was evaluated by stakeholders who identified more requirements
- We then carried out a series of four two-week sprints and presented working code at the end of each sprint to other researchers
- The app was then evaluated by PLHIV
- Software development approach: **agile**



Reading

- Two research papers about evaluating pair programming mentioned in the lecture

R. W. Jensen (2005) A Pair Programming Experience, Overload, 13(65)

A. Cockburn & L. Williams (2001) The Costs and benefits of pair programming. Extreme programming examined. G. Succi and M. Marchesi, Addison Wesley: 223-243.

- A good overview of the Agile Manifesto and approach

Overview of the Agile Approach

- The Kanban method

https://kanbanize.com/kanban-resources/getting-started/what-is-kanban

- Unit testing

Unit testing in Java

Reminder before the workshop

- Please install Processing on your laptop: https://processing.org/download
- Please make sure that the next two items are in the “Done” column of your Kanban board:
  - Upload a team photo to your team’s repo
  - Research games and upload a list of games you find inspiring to your team’s repo