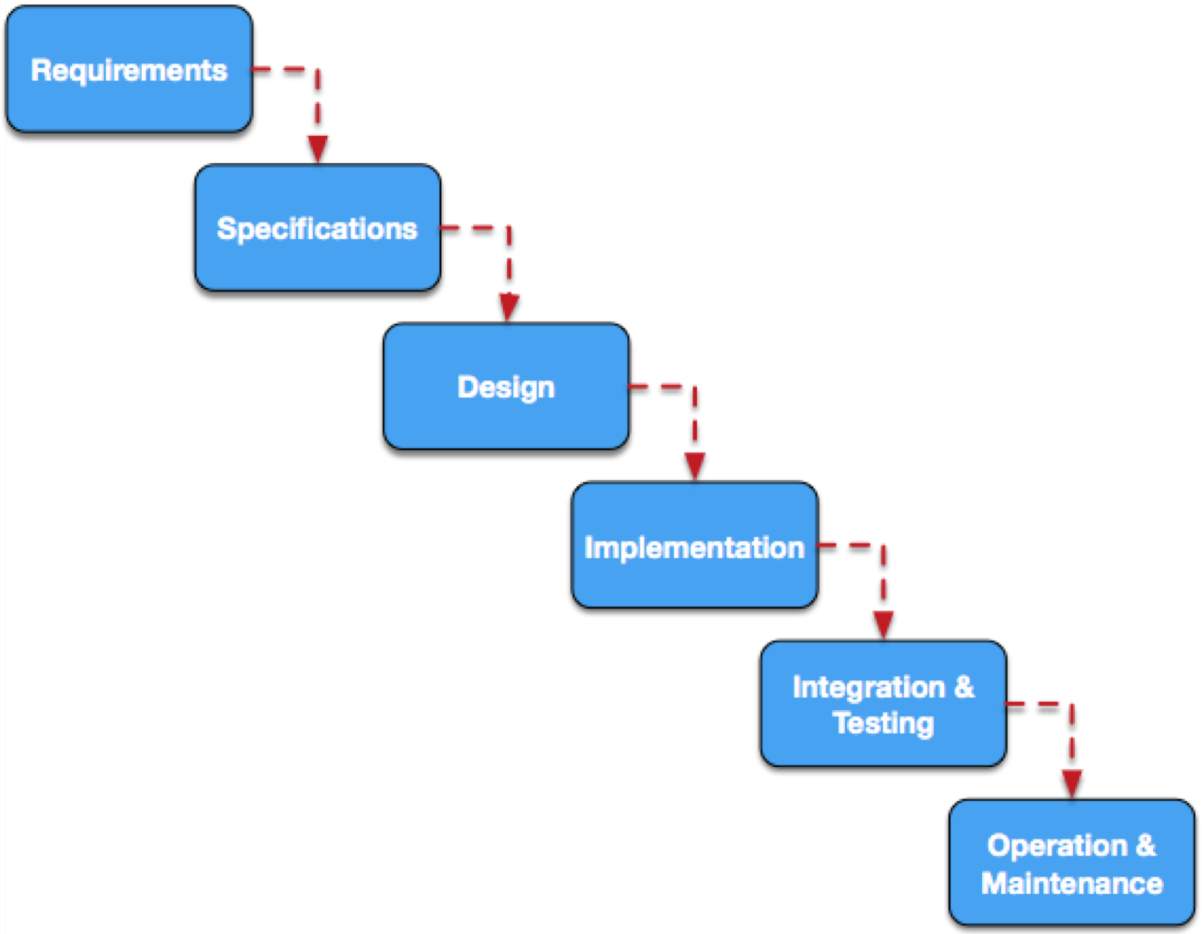
UNIVERSITY OF LEICESTER

Agile Software Development

CO3095, CO7095, CO7508

José Miguel Rojas – *material provided by Neil Walkinshaw*

Waterfall



The Rise and Fall of WATERFALL

Agile Software Development

Emerged in late 90s, early 2000's.

Borrowed ideas from IID, and other process innovations.

People and product centric (as opposed to process and requirements).

Principles enshrined in **"Agile Manifesto"** (2001) by several authors including Martin Fowler and Kent Beck.

Led to a multitude of "Agile methods" – Extreme Programming (XP), SCRUM, etc.

Here we will learn about the core approach and additional Agile-related approaches will be studied later on.

Agile Manifesto Priorities

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

Principles of Agile Software Development

Customer satisfaction

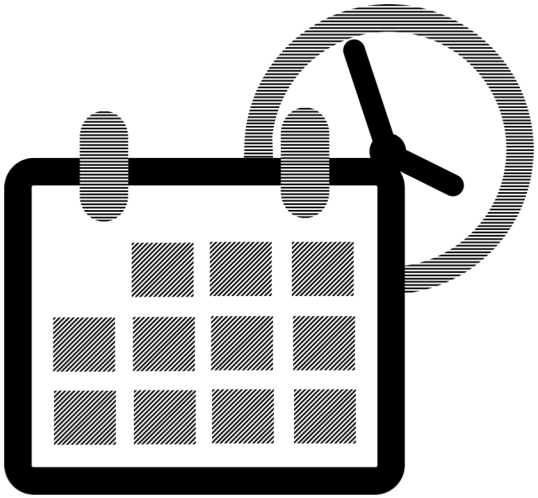
The highest priority is to satisfy the customer through early and continuous delivery of valuable software

Principles of Agile Software Development

Welcome changing requirements

And accommodate them throughout the development process
(even if late into it)

Principles of Agile Software Development



Frequent delivery

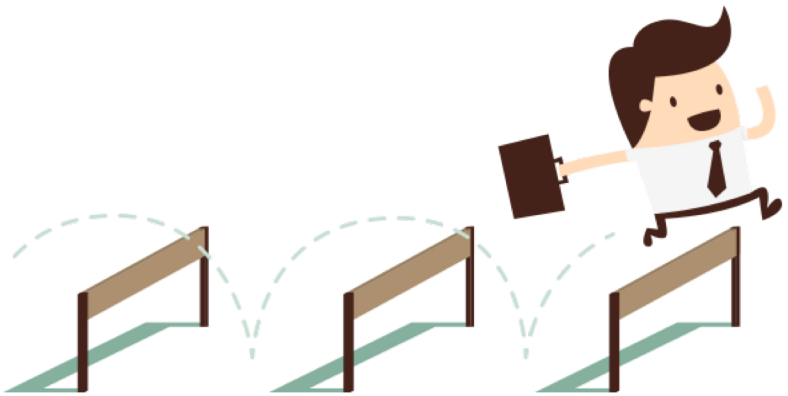
of working software – weeks rather than months

Principles of Agile Software Development

Close collaboration

Daily cooperation between business people and development team

Principles of Agile Software Development



Support, trust and motivation

Projects are built around motivated individuals who should be trusted

Principles of Agile Software Development

Working software

Primary measure of progress

Principles of Agile Software Development

Face-to-face interactions

Allows business people and developers to communicate well

Principles of Agile Software Development

Sustainable development

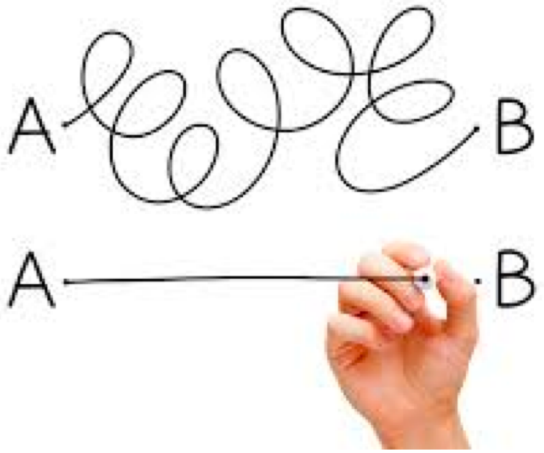
Agile processes to support a consistent development pace

Principles of Agile Software Development

Attention to technical excellence and details

Agile processes to support a consistent development pace

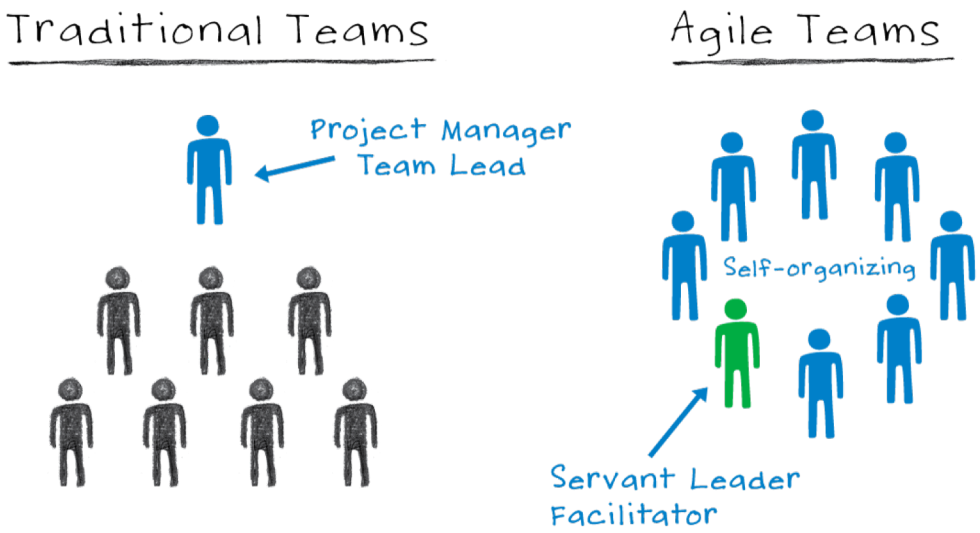
Principles of Agile Software Development



Simplicity

Develop just enough to get the job done for right now

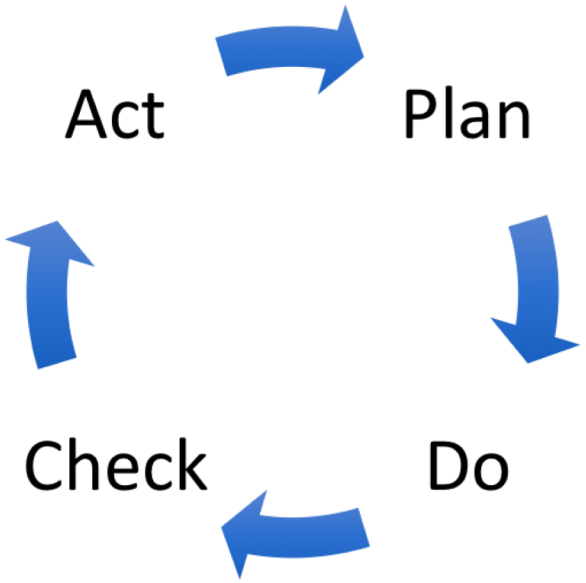
Principles of Agile Software Development



Self-organising teams

To encourage encourage great architectures, requirements, and designs

Principles of Agile Software Development



Regular reflection

On how to become more effective

Principles of Agile Software Development

1. **Customer Satisfaction** by early and continuous delivery of software.
2. **Welcome changing requirements** even if late in development.
3. **Working software is delivered frequently** (weeks rather than months).
4. **Close, daily cooperation** between business people and developers.
5. **Projects are built around motivated individuals** who should be trusted.
6. **Face-to-face conversation** is the best form of
7. **Working software** is the principle measure of
8. **Sustainable development** able to maintain a
9. **Continuous attention to technical excellenc**
10. **Simplicity** - the art of maximising the amount
11. **Self-organising teams** produce the best syste
12. **The team must regularly reflect** on how to be

Heijunka - have the correct number of parts required to build a vehicle
Nemawashi - decisions should be arrived at as a team, not dictated by individuals
Genba - understand work load on individuals, ensure processes are transparent

Andon - enable workers to immediately highlight threats via alert mechanisms
Just-In-Time - Products should only be developed if and when they are required
Jidoka - Capture faults as close to their source as possible
Genchi Genbutsu - the best way to solve a problem is to see it for yourself
Kanban - a board to communicate the state of the manufacturing system

Kaizen - Staff encouraged to look for areas of improvement.
Muda, Muri, Mura - eliminate waste.

Poke Yoke - Prevent errors from occurring by embedding prevention mechanisms.
Hansen - Learn from mistakes to prevent them from recurring

Similarities with the Toyota Production System?

UNIVERSITY OF LEICESTER

Toyota Production System

Plan	**Heijunka** - have the correct number of parts required to build a vehicle **Nemawashi** - decisions should be arrived at as a team, not dictated by individuals **Genba** - understand work load on individuals, ensure processes are transparent
Do	**Andon** - enable workers to immediately highlight threats via alert mechanisms **Just-In-Time** - Products should only be developed if and when they are required **Jidoka** - Capture faults as close to their source as possible **Genchi Genbutsu** - the best way to solve a problem is to see it for yourself **Kanban** - a board to communicate the state of the manufacturing system
Check	**Kaizen** - Staff encouraged to look for areas of improvement. **Muda, Muri, Mura** - eliminate waste.
Act	**Poke Yoke** - Prevent errors from occurring by embedding prevention mechanisms. **Hansen** - Learn from mistakes to prevent them from recurring

Maximising opportunity for feedback.

How Agile Compares with Traditional Approaches

	Traditional	Agile

How Agile Compares with Traditional Approaches

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality adaptive software can be developed by small teams using continuous design improvement and testing, based on rapid feedback and change

How Agile Compares with Traditional Approaches

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality adaptive software can be developed by small teams using continuous design improvement and testing, based on rapid feedback and change
Control	Process-centric	People-centric

How Agile Compares with Traditional Approaches

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality adaptive software can be developed by small teams using continuous design improvement and testing, based on rapid feedback and change
Control	Process-centric	People-centric
Management Style	Command-and-control	Leadership-and-collaboration.

How Agile Compares with Traditional Approaches

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality adaptive software can be developed by small teams using continuous design improvement and testing, based on rapid feedback and change
Control	Process-centric	People-centric
Management Style	Command-and-control	Leadership-and-collaboration.
Knowledge Management	Explicit.	Tacit

How Agile Compares with Traditional Approaches

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality adaptive software can be developed by small teams using continuous design improvement and testing, based on rapid feedback and change
Control	Process-centric	People-centric
Management Style	Command-and-control	Leadership-and-collaboration.
Knowledge Management	Explicit.	Tacit
Role Assignment	Individual – favours specialisation	Self-organising teams, encourages role interchangeability

How Agile Compares with Traditional Approaches

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality adaptive software can be developed by small teams using continuous design improvement and testing, based on rapid feedback and change
Control	Process-centric	People-centric
Management Style	Command-and-control	Leadership-and-collaboration.
Knowledge Management	Explicit.	Tacit
Role Assignment	Individual – favours specialisation	Self-organising teams, encourages role interchangeability
Communication	Formal	Informal

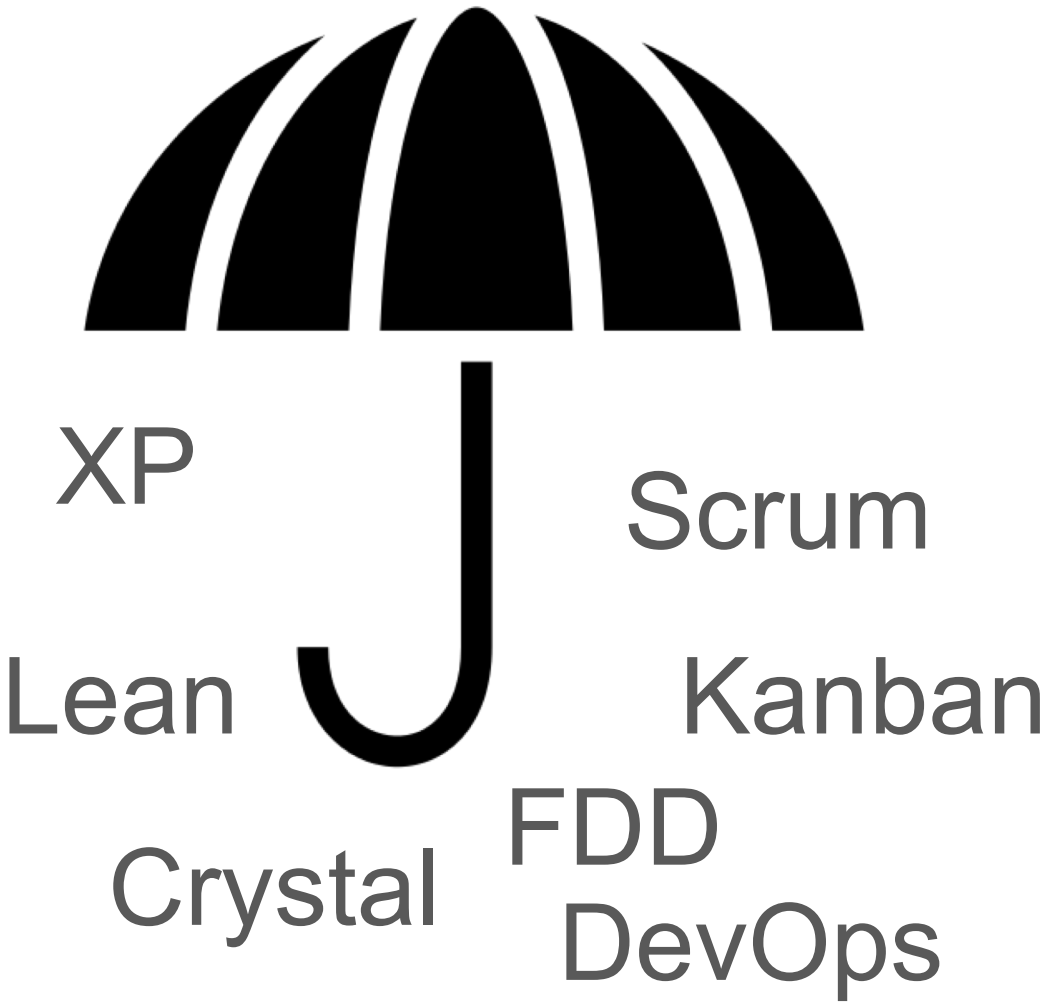
How Agile Compares with Traditional Approaches

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality adaptive software can be developed by small teams using continuous design improvement and testing, based on rapid feedback and change
Control	Process-centric	People-centric
Management Style	Command-and-control	Leadership-and-collaboration.
Knowledge Management	Explicit.	Tacit
Role Assignment	Individual – favours specialisation	Self-organising teams, encourages role interchangeability
Communication	Formal	Informal
Project Cycle	Guided by tasks or activities	Guided by product features

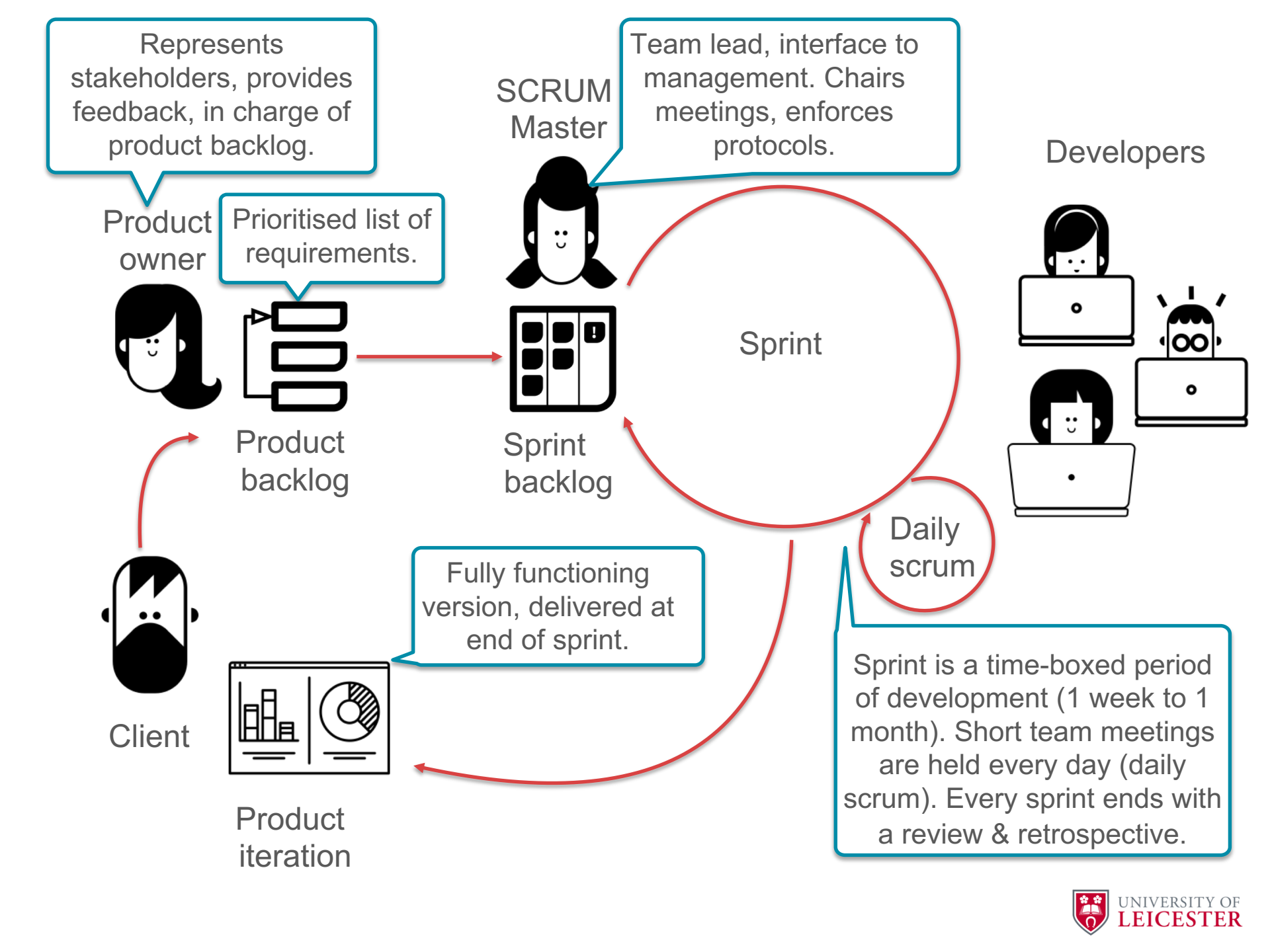
How Agile Compares with Traditional Approaches

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality adaptive software can be developed by small teams using continuous design improvement and testing, based on rapid feedback and change
Control	Process-centric	People-centric
Management Style	Command-and-control	Leadership-and-collaboration.
Knowledge Management	Explicit.	Tacit
Role Assignment	Individual – favours specialisation	Self-organising teams, encourages role interchangeability
Communication	Formal	Informal
Project Cycle	Guided by tasks or activities	Guided by product features
Development Model	Life-cycle model	Evolutionary – delivery model

Agile Methodologies



The SCRUM Method



User Stories

ID - Name:

As a <Role> I want <Function/Attribute> so that I can achieve <Goal>.

Priority _ Cost _

User Stories

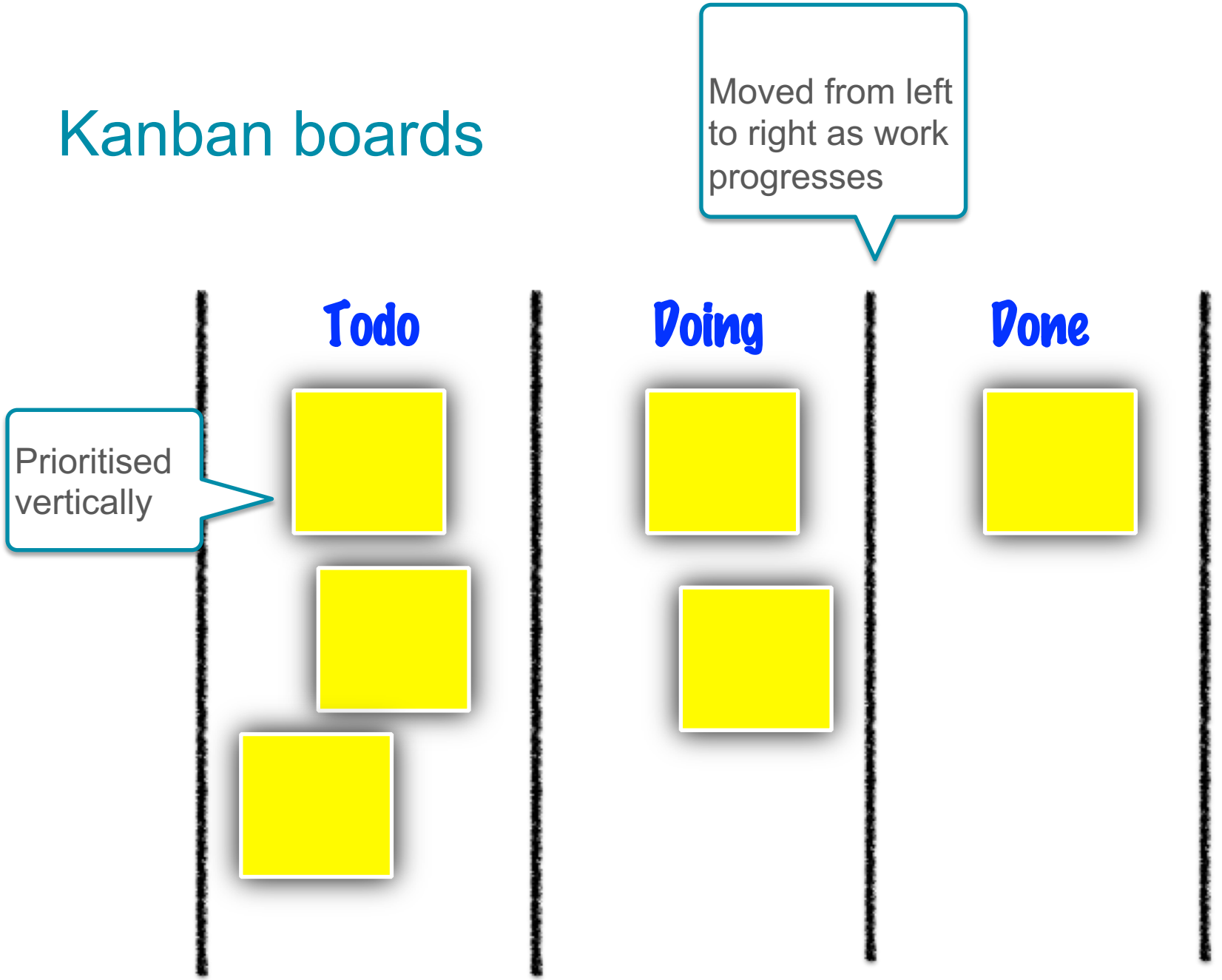
User story #1 - Automatic Account Expiry

As a system administrator I want to be able to set an account expiry date so that user accounts can be disabled automatically.

Priority: MUST **Estimate: 4**

We will come on to prioritisation and cost estimation later!

Kanban boards



Daily SCRUM

A standing meeting every day.

Every member states in turn:

What they did in the last day.

What they are planning for this day.

Are there any impediments.

Standing meetings date back to Queen Victoria, who made privy council meetings stand so that they would take less time – a tradition that last to this day.

Is Agile exclusive to Software Development?

"[Agility] comes in different forms, but basically it's the ability to quickly adapt to or even anticipate and lead change. Agility in the broadest form affects strategic thinking, operations, technology innovation and the ability to innovate in products, processes and business models."

Michael A. Cusumano, MIT Sloan School of Management

?

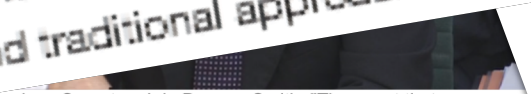
Does “Agile” always succeed?

BBC Sign in News Sport Weather

NEWS POLITICS

Home | World | UK | England | N. Ireland | Scotla

9 December 2013 Last



and Pensions Secretary Iain Duncan Smith: "The asset that we now actually own that will take universal credit forward is actually worth to the government £151 million," he said.

Work and Pensions Secretary Iain Duncan Smith has rejected claims his flagship welfare policy is a "debacle".

Mr Duncan Smith was being questioned by MPs amid fears the Universal Credit system would miss key targets and concerns over the IT system being used.

He said £40.1m had been written off on software and computing costs in implementing the new system.

Labour claims only a fraction of those expected to be switched to the new

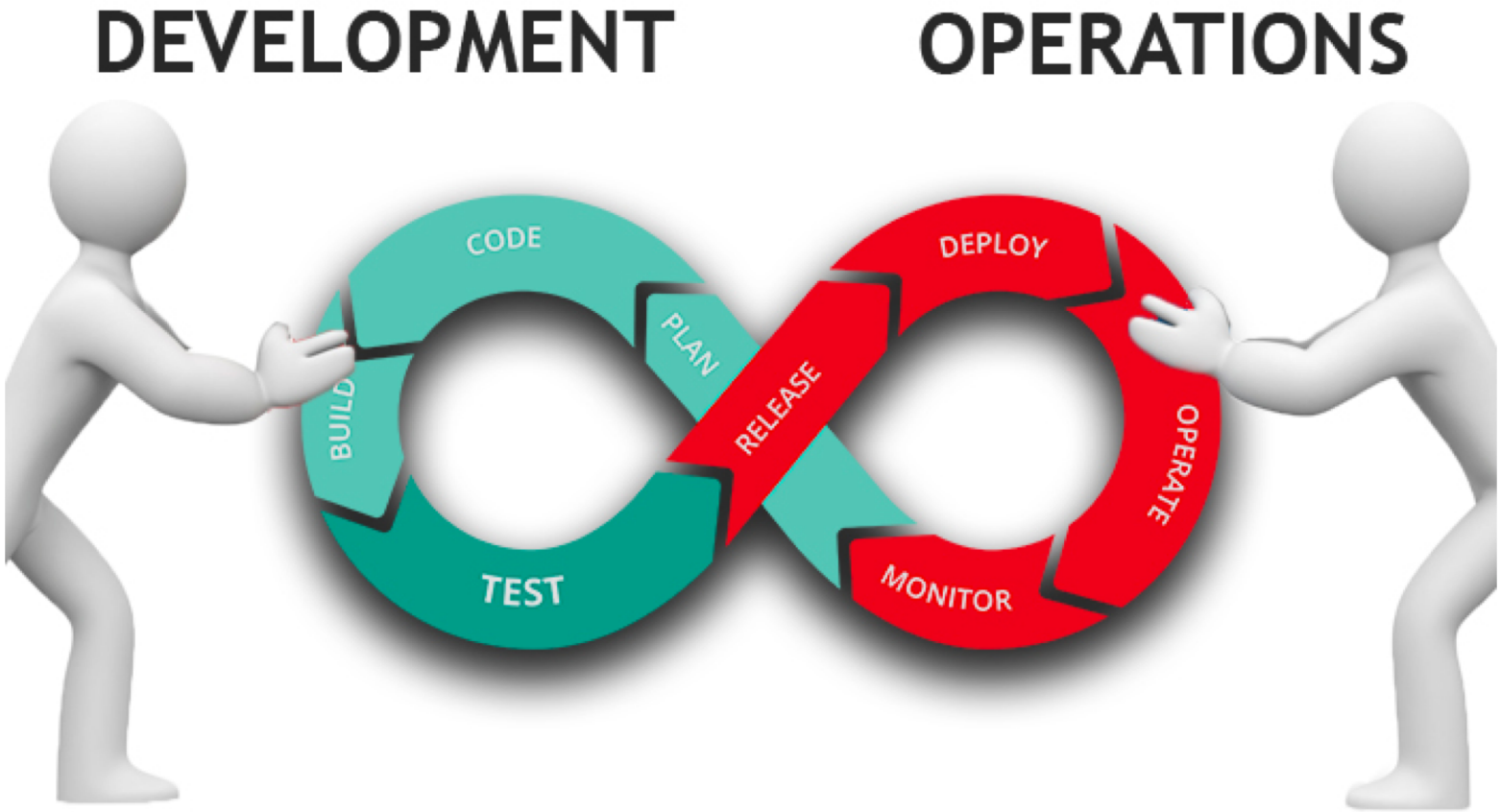
NAO National Audit Office

17 The Department tried to use an 'agile' approach to develop processes and systems at the same time as defining policy requirements. The agile approach uses iterative and collaborative project management to develop its IT and policy. This was the first time the Department had tried to use this approach on a major programme of this scale. The Department experienced problems incorporating the agile approach into existing contracts, governance and assurance structures. In January 2012, the Department introduced Agile 2.0, a hybrid approach which tried to combine elements of agile and traditional approaches to IT programme management (paragraphs 3.6 to 3.9).

SESSION 2013-14 5 SEPTEMBER 2013

UNIVERSITY OF LEICESTER

DevOps



DevOps strives to focus on the overall service or software fully delivered to the customer instead of simply “working software”.

Conclusions

Agile techniques are example of IID.

Include principles that are derived from manufacturing advances such as Toyota's TPS and TQM.

Emphasis on *people*. Focus on interactions amongst developers and with clients, as opposed to processes.