

Module 3: Agile Software Development

Software Engineering

School of Computing, Faculty of Engineering
Universiti Teknologi Malaysia

Outline

www.utm.my

- Agile methods
- Agile development techniques
- Agile project management
- Scaling agile methods

Note: The overall contents of the slide are based on the main reference that is Sommerville (2016) with other references specified directly in respective slides

Objectives

www.utm.my

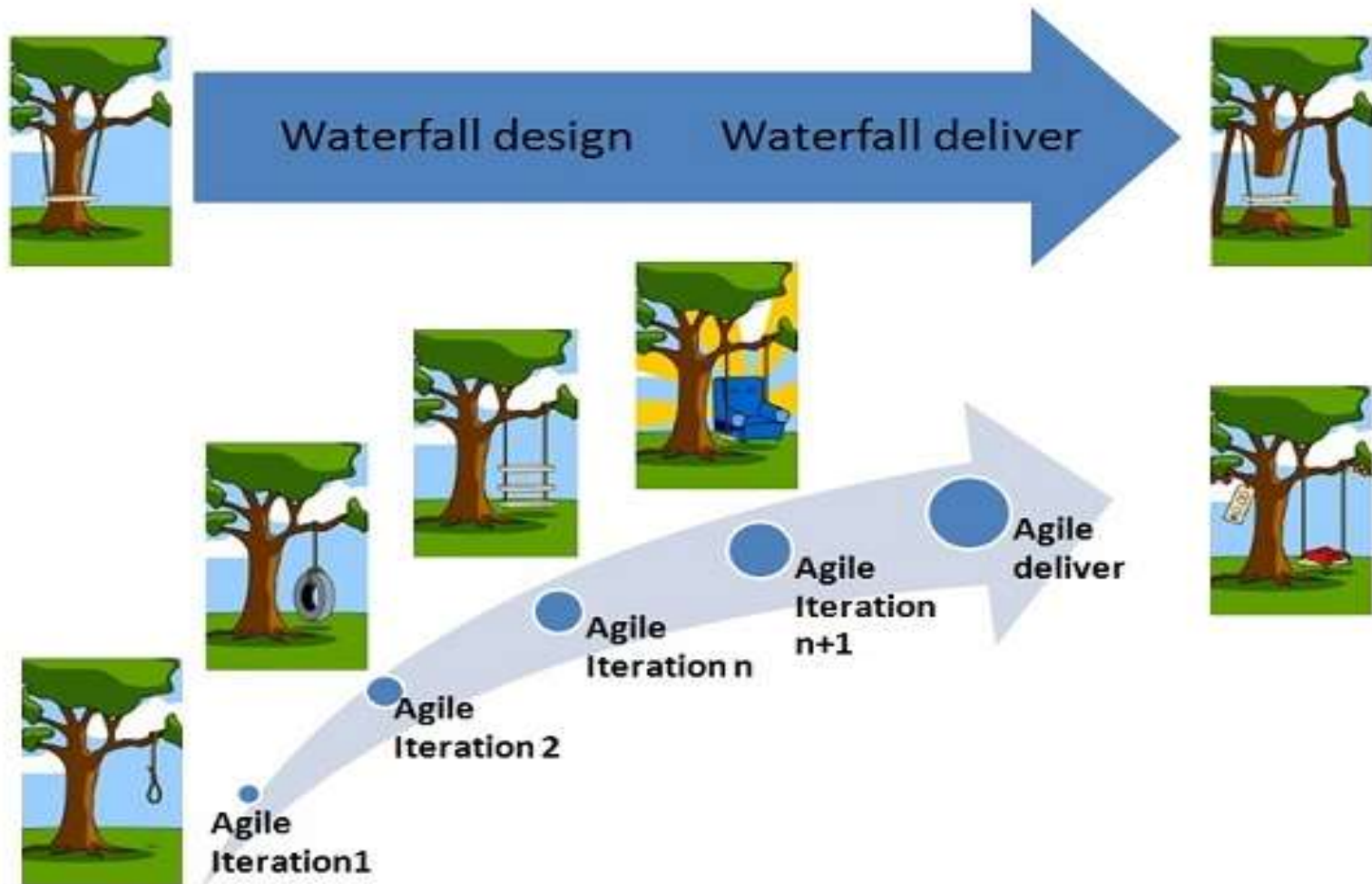
The objectives of this module are:

- To understand the rationale for agile software development methods, its manifesto, and the differences between agile and plan-driven development
- To recognise the importance of agile development techniques and practices
- To understand the Scrum approach in agile project management
- To be aware of the issues of scaling agile development methods

AGILE METHODS

Introduction

www.utm.my



Rapid Software Development

www.utm.my

- **Rapid development and delivery** is now often the most important requirement for software systems:
 - Businesses operate in a fast-changing requirement and it is practically impossible to produce a set of stable software requirements
 - Software has to evolve quickly to reflect changing business needs
- **Plan-driven development** is essential for some types of system but does not meet these business needs
- **Agile development** methods emerged in the late 1990s whose aim was to radically reduce the delivery time for working software systems

Agile Development

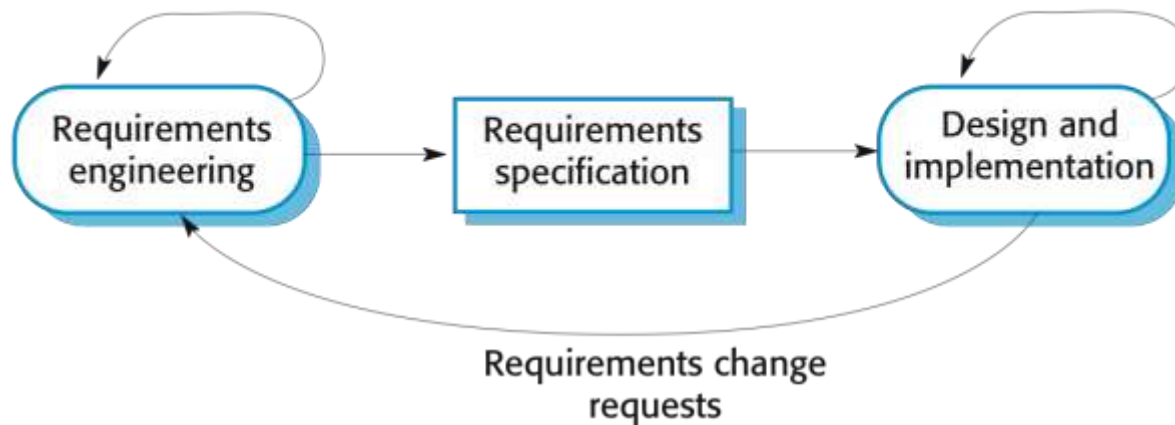
www.utm.my

- Program specification, design and implementation are **inter-leaved**
- The system is developed as a **series of versions** or increments with stakeholders involved in version specification and evaluation
- **Frequent delivery** of new versions for evaluation
- **Extensive tool** support (e.g. automated testing tools) used to support development
- **Minimal documentation** – focus on working code

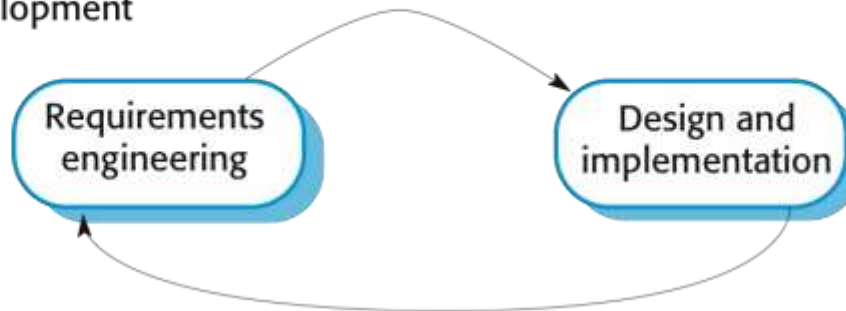
Plan-Driven and Agile Development...

www.utm.my

Plan-based development



Agile development



Plan-Driven and Agile Development

www.utm.my

- **Plan-driven development**

- A plan-driven approach to software engineering is based around **separate development stages** with the outputs to be produced at each of these stages planned in advance
- Not necessarily waterfall model – plan-driven, incremental development is possible
- **Iteration** occurs within activities

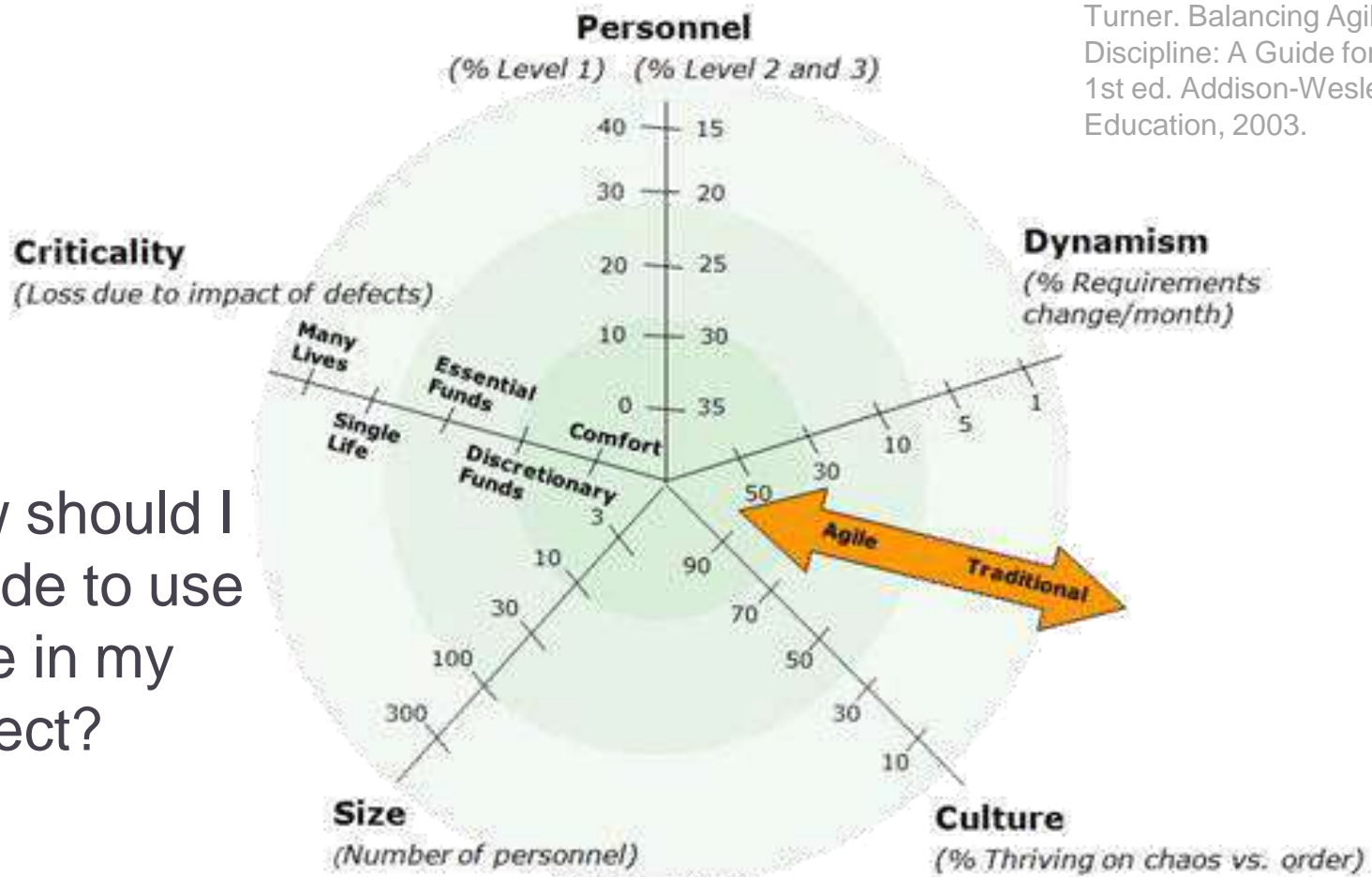
- **Agile development**

- Deliver ideally 2-3 weeks
- Specification, design, implementation and testing are **inter-leaved** and the outputs from the development process are decided through a process of negotiation during the software development process

Balance of Plan-Driven or Agile Development...

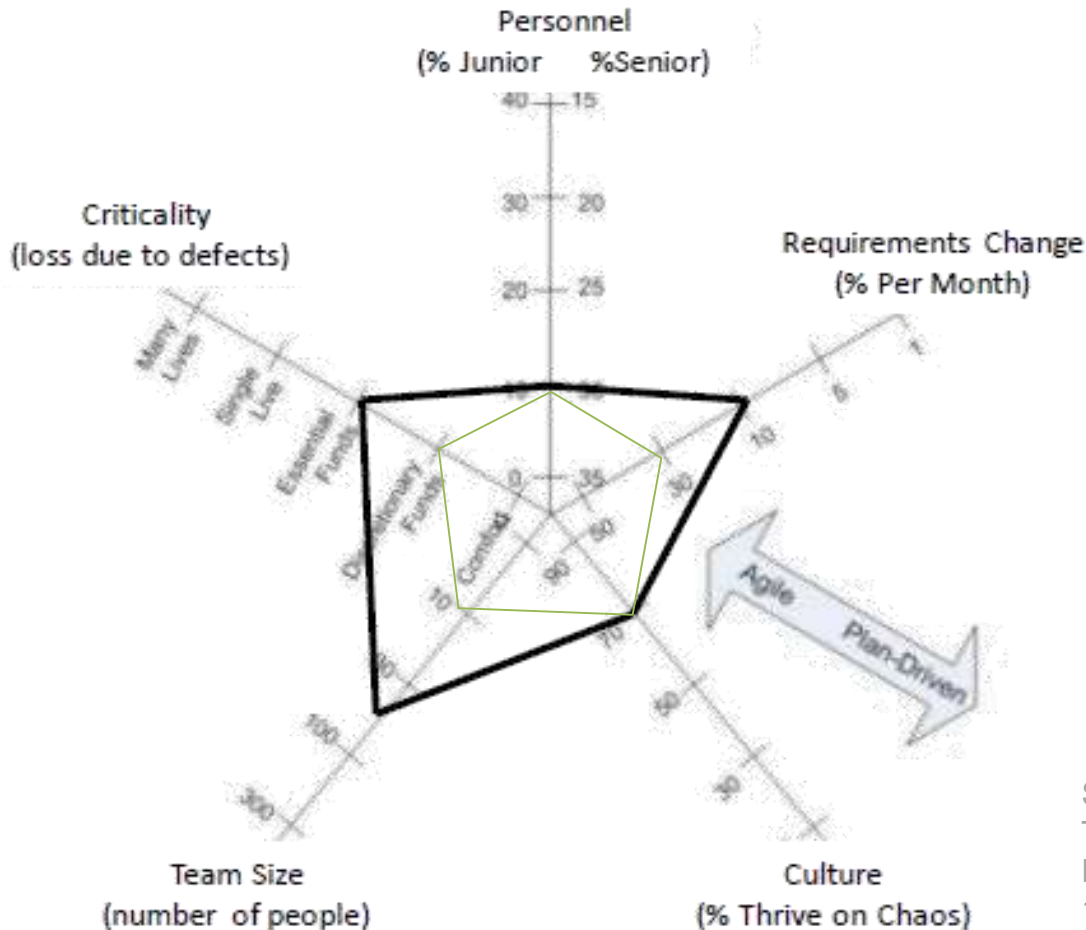
www.utm.my

Source: Boehm, Barry, and Richard Turner. Balancing Agility and Discipline: A Guide for the Perplexed. 1st ed. Addison-Wesley/Pearson Education, 2003.



✧ How should I decide to use agile in my project?

Balance of Plan-Driven or Agile Development



Source: . Boehm, Barry, and Richard Turner. Balancing Agility and Discipline: A Guide for the Perplexed. 1st ed. Addison-Wesley/Pearson Education, 2003.

Plan-Driven and Agile...

www.utm.my

ALIGNING PROJECT TRAITS *with* DEVELOPMENT METHODOLOGIES

PROJECT TRAIT/FACTOR	AGILE	PLAN - DRIVEN (WATERFALL)	COMMENTS
CUSTOMER AVAILABILITY	Prefers customer available throughout project.	Requires customer involvement only at milestones.	Customer involvement reduces risk in either model.
SCOPE/ FEATURES	Welcomes changes, but changes come at the expensive of Cost, Schedule, or other Features. Works well when scope is not known in advance.	Works well when scope is known in advance, or when contract terms limit changes.	Change is a reality so we should prefer adaptability where possible. Contract terms sometimes restrict it.
FEATURE PRIORITIZATION	Prioritization by value ensures the most valuable features are implemented first, thus reducing risk of having an unusable product once funding runs out. Funding efficiency is maximized. Decreases risk of complete failure by allowing "partial" success.	"Do everything we agreed on" approach ensures the customer gets everything they asked for; "all or nothing" approach increases risk of failure.	Contract terms may not permit partial success and may require "do everything".

Plan-Driven and Agile

www.utm.my

TEAM	Prefers smaller, dedicated teams with a high degree of coordination and synchronization.	Team coordination/ synchronization is limited to handoff points	Teams that work together work better, but when contracts are issued to different vendors for different aspects of the project, high degrees of synchronization may not work.
FUNDING	Works extremely well with Time & Materials or other non-fixed funding, may increase stress in fixed-price scenarios.	Reduces risk in Firm Fixed Price contracts by getting agreement up-front.	Fixed price is tough when scope is not known in advance, but many government contracts require it.
SUMMARY	Agile is better, where it is feasible.	Plan-Driven may reduce risk in the face of certain constraints in a contract between a vendor and external customer such as the government.	Through educating our customers about the strengths and weaknesses of each model, we hope to steer them towards a more Agile approach. This may require changes to how our customers, particularly the government, approach software development projects.

Agile Methods

www.utm.my

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods that:
 - **Focus on the code** rather than the design
 - Are based on an **iterative approach** to software development
 - Are intended **to deliver working software quickly** and **evolve this quickly** to meet changing requirements
- The aim of agile methods is to **reduce overheads** in the software process (e.g. by limiting documentation) and **to be able to respond quickly** to changing requirements without excessive rework

Agile Manifesto

www.utm.my

Source: <http://agilemanifesto.org>

- ✧ The philosophy emphasizes flexibility and producing software quickly and capably.
- ✧ *“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.”*

The Principles of Agile Methods

www.utm.my

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is to provide and prioritize new system requirements, select the features to be included in each successive build and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Agile Methods Applicability

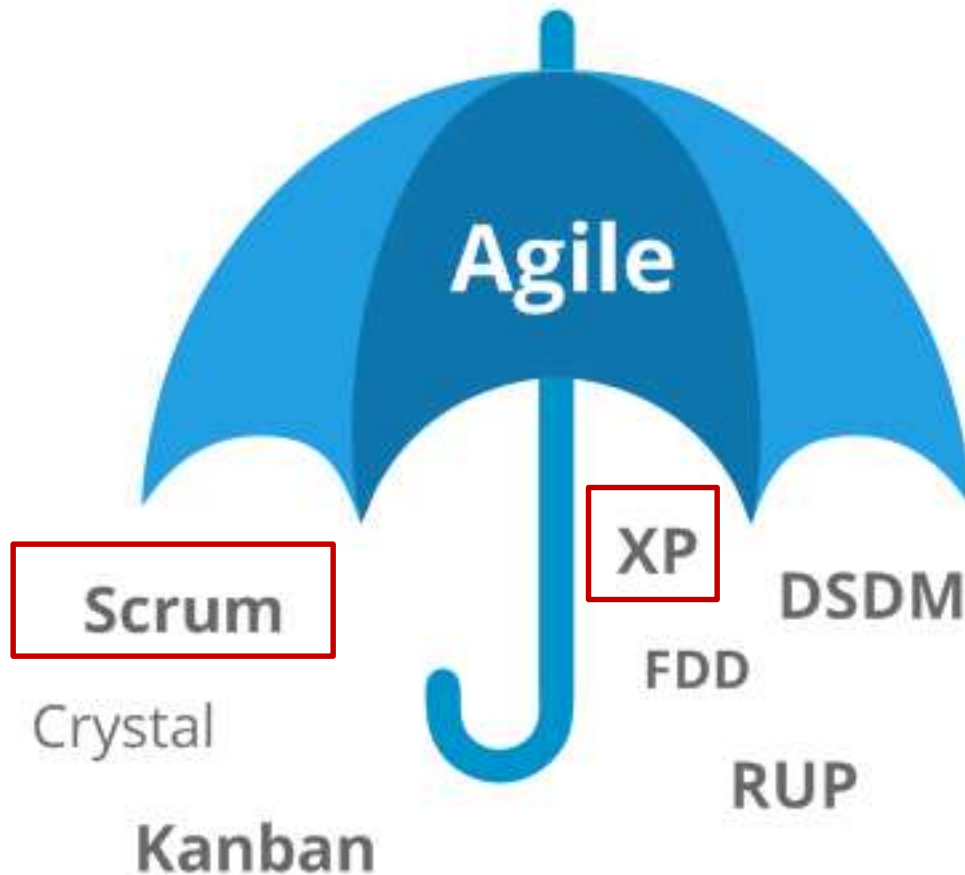
www.utm.my

- Product development where a software company is developing a **small or medium-sized product** for sale
 - Virtually all software products and apps are now developed using an agile approach
- Custom system development within an organization, where:
 - there is a **clear commitment from the customer** to become involved in the development process
 - there are **few external rules and regulations** that affect the software

AGILE DEVELOPMENT TECHNIQUES

Agile Development Techniques

www.utm.my



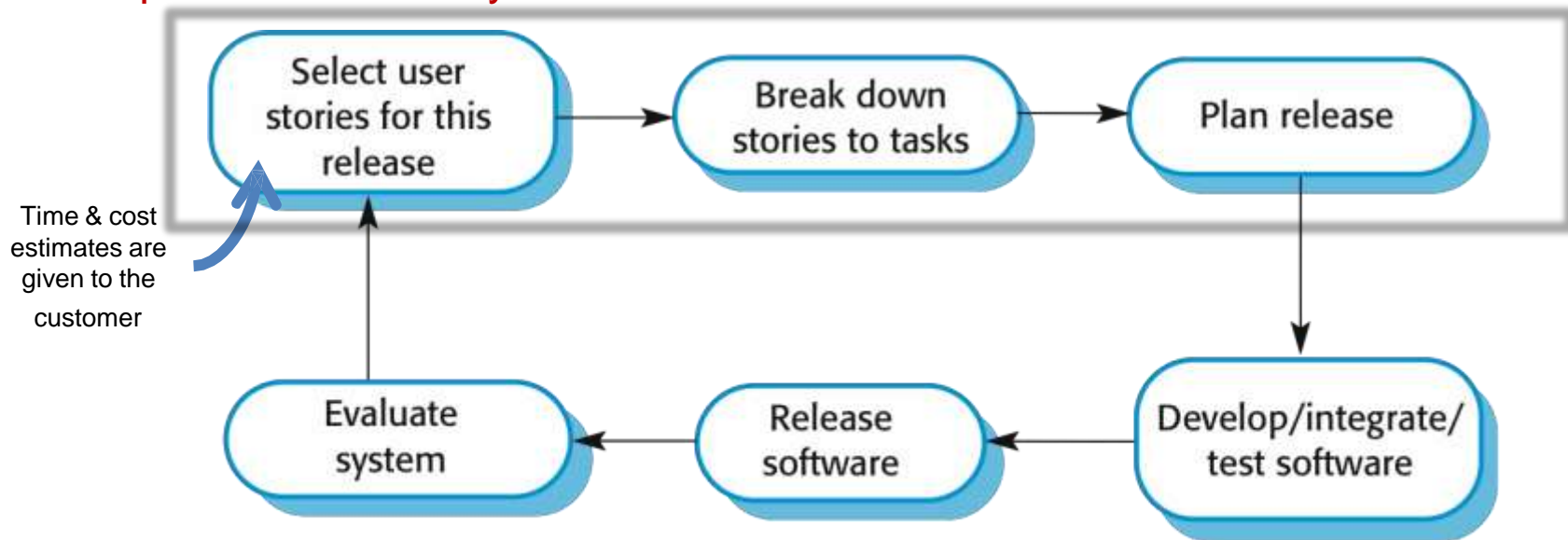
Extreme Programming

www.utm.my

- A very influential agile method, developed in the late 1990s, that introduced a range of agile development techniques
- Extreme Programming (XP) takes an 'extreme' approach to iterative development:
 - **New versions** may be built several times per day
 - **Increments** are delivered to customers every 2 weeks
 - **All tests** must be run for every build and the build is only accepted if tests run successfully

The Extreme Programming Release Cycle

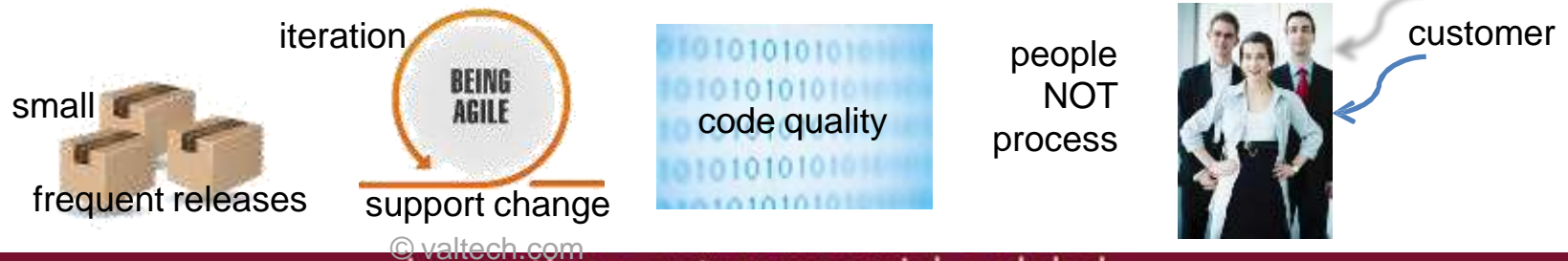
Requirements & Analysis



XP and Agile Principles

www.utm.my

- **Incremental development** is supported through small, frequent system releases
- **Customer involvement** means full-time customer engagement with the team
- **People not process** through pair programming, collective ownership and a process that avoids long working hours
- **Change** supported through regular system releases
- **Maintaining simplicity** through constant refactoring of code



Influential XP Practices

www.utm.my

- Extreme programming has a **technical focus** and is not easy to integrate with management practice in most organizations
- Consequently, while agile development uses practices from XP, the method as originally defined is not widely used
- Key practices:
 1. User stories for specification
 2. Refactoring
 3. Test-first development
 4. Pair programming

1: User stories for specifications

www.utm.my

- User stories describe a feature from the user perspective
- In XP, a customer (or user) is part of the XP team and is responsible for making decisions on requirements
- User **requirements** are expressed as **scenarios or user stories**
- These are written on cards and the development team break them down into implementation tasks as the basis of schedule and cost estimates
- The customer **chooses the stories** for inclusion in the next release **based on their priorities** and the schedule estimates

Example: 'prescribing medication' Story

www.utm.my

User type –
Who wants to
benefit from
this
requirement

Field options

A **doctor** prescribing medication for a patient

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select '**current medication**' you will be asked to **check the dose**; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose '**new medication**', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose '**formulary**', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

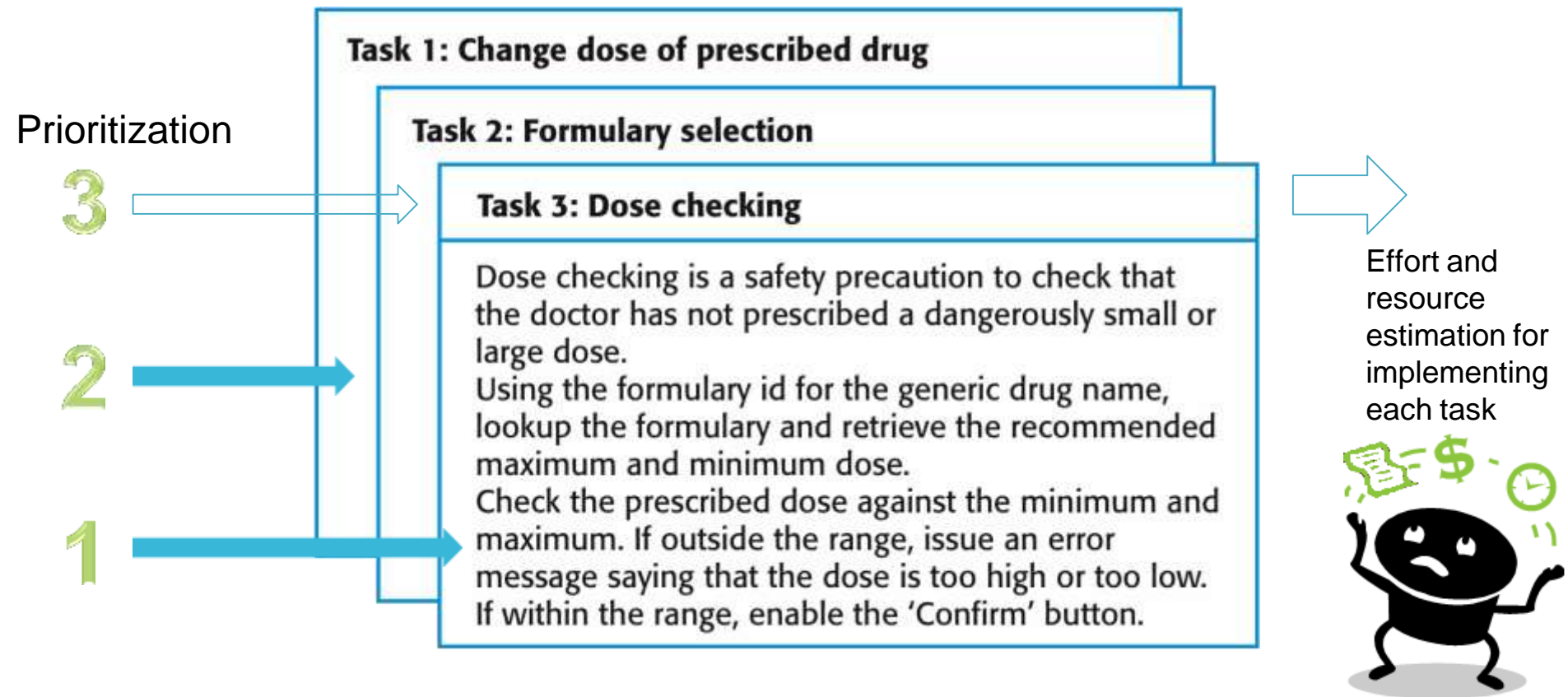
In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

Functionality

– Outlines
what the
software
should do

Example: Task cards for prescribing medication



2: Refactoring...

www.utm.my

- Conventional wisdom in software engineering is to **design for change**
- It is worth spending time and effort anticipating changes as this reduces costs later in the life cycle
- XP, however, maintains that this is **not worthwhile** as changes cannot be reliably anticipated
- Rather, it proposes **constant code improvement (refactoring)** to make changes easier when they have to be implemented

2: Refactoring

www.utm.my

- Programming team looks for **possible software improvements** and make these improvements even where there is no immediate need for them
- This practice **improves the understandability** of the software, thus reduces the need for documentation
- Changes are easier to make because **the code is well-structured and clear**
- However, some changes **require architecture refactoring** and this is much more expensive

Example of Refactoring

www.utm.my

- Re-organization of a class hierarchy to remove duplicate code
- Tidying up and renaming attributes and methods to make them easier to understand
- The replacement of inline code with calls to methods that have been included in a program library

3: Test-First Development

www.utm.my

- Testing is central to XP and XP has developed an approach where **the program is tested after every change** has been made
- XP testing features:
 - Test-first development
 - Incremental test development from scenarios
 - User involvement in test development and validation
 - Automated test harnesses are used to run all component tests each time that a new release is built

4: Pair Programming...

www.utm.my

- Pair programming involves programmers **working in pairs**, developing code together
- This helps develop **common ownership of code and spreads knowledge** across the team
- It serves as an **informal review** process as each line of code is looked at by more than 1 person
- It **encourages refactoring** as the whole team can benefit from improving the system code

4: Pair Programming

www.utm.my

- In pair programming, programmers **sit together** at **the same computer** to develop the software
- Pairs are created **dynamically** so that all team members work with each other during the development process
- The **sharing of knowledge** that happens during pair programming is very important as it **reduces the overall risks** to a project when team members leave
- Pair programming is not necessarily inefficient and there is some evidence that suggests that a pair working together is **more efficient** than 2 programmers working separately

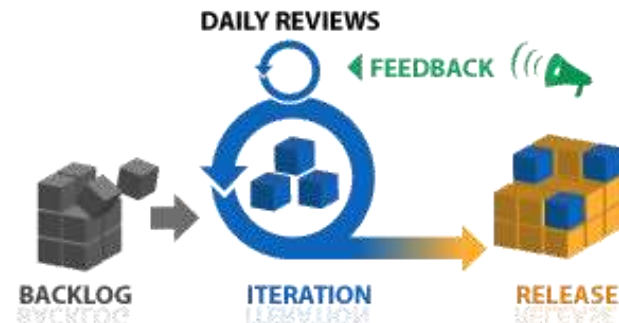
AGILE PROJECT MANAGEMENT

Agile Project Management

www.utm.my

- The principal responsibility of **software project managers** is to manage the project so that the software is delivered on time and within the planned budget for the project
- The standard approach to project management is **plan-driven**
- Managers draw up a plan for the project showing **what** should be delivered, **when** it should be delivered and **who** will work on the development of the project deliverables
- Agile project management requires a different approach, which is **adapted to incremental development** and the practices used in agile methods

© <http://www.planbox.com/resources/agile-project-management>



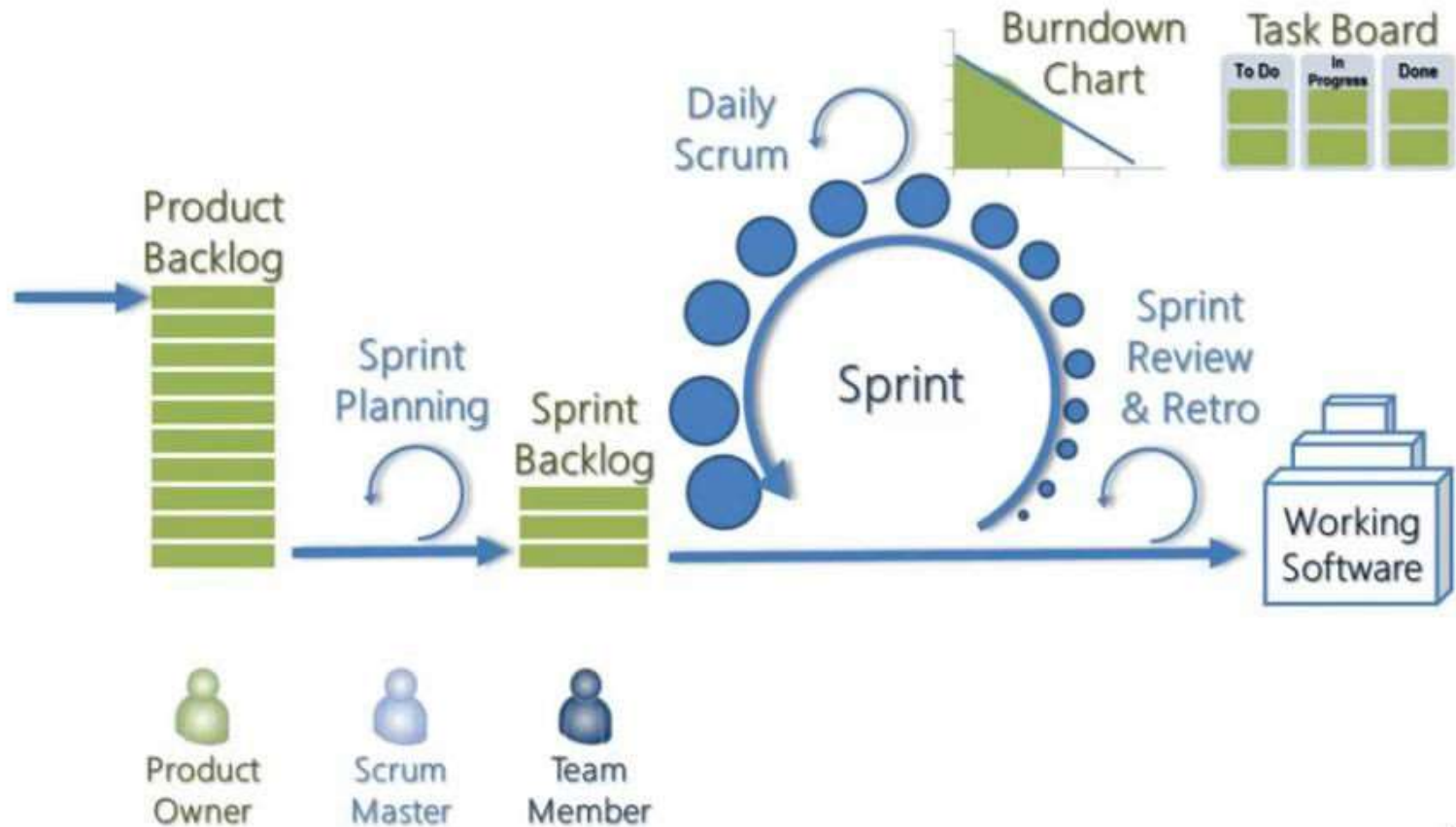
Scrum

www.utm.my

- Scrum is an agile method that focuses on **managing iterative development** rather than specific agile practices
- There are **three phases** in Scrum:
 - The **initial phase** is an outline planning phase where the team establishes the general objectives for the project and design the software architecture
 - This is followed by **a series of sprint cycles**, where each cycle develops an increment of the system
 - The project closure phase **wraps up the project**, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project

The Scrum Process

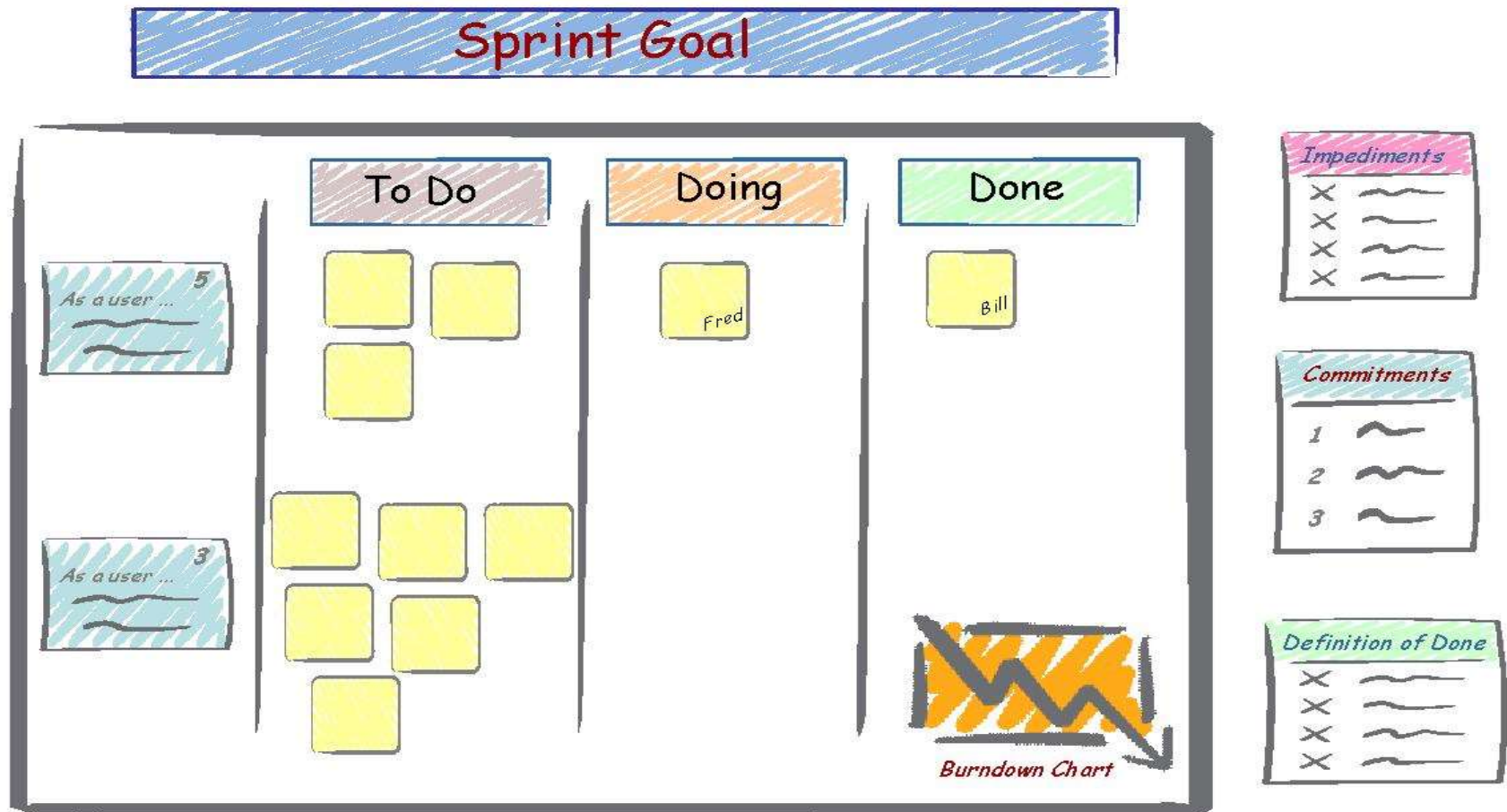
www.utm.my



Task Board

www.utm.my

Task board



The Scrum Sprint Cycle

www.utm.my

- Sprints are fixed length, normally 2–4 weeks
- The starting point for planning is the **product backlog**, which is the list of work to be done on the project
- The **selection phase** involves all the project team members who work with the customer to select the **features and functionality** from the **product backlog** to be developed during the sprint

The Sprint Cycle

www.utm.my

- Once these are agreed, the team organize themselves to develop the software
- During this stage the **team is isolated from the customer and the organization**, with all communications channelled through the so-called 'Scrum master'
- The role of the **Scrum master** is to protect the development team from external distractions
- At the **end of the sprint**, the work done is **reviewed and presented** to stakeholders
- The next sprint cycle then begins

Teamwork in Scrum



www.utm.my

- The '**Scrum master**' is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team
- The **whole team attends short daily meetings** (Scrums) where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day
- This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them

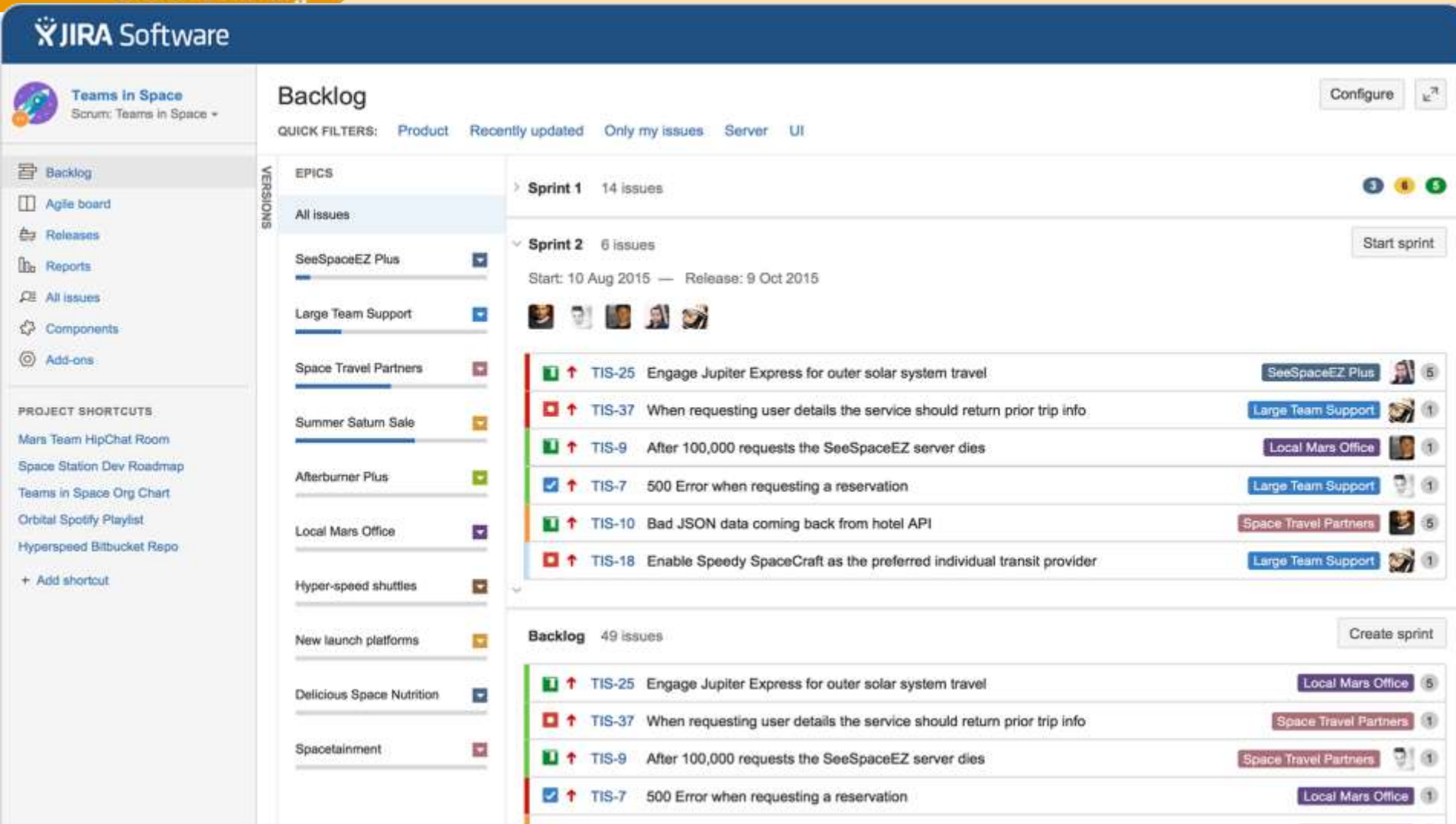
Scrum Benefits

www.utm.my

- The product is broken down into a set of **manageable** and **understandable** chunks
- Unstable requirements do not hold up progress
- The whole team have **visibility** of everything and consequently team communication is improved
- Customers see **on-time delivery** of increments and gain feedback on how the product works
- **Trust between customers and developers** is established and a positive culture is created in which everyone expects the project to succeed

Example of Agile Project Management: JIRA Software

www.utm.my



JIRA Software

Teams in Space
Scrum: Teams in Space

Backlog

QUICK FILTERS: Product Recently updated Only my issues Server UI

EPICS

- All issues
- SeeSpaceEZ Plus
- Large Team Support
- Space Travel Partners
- Summer Saturn Sale
- Afterburner Plus
- Local Mars Office
- Hyper-speed shuttles
- New launch platforms
- Delicious Space Nutrition
- Spacetainment

Sprint 1 14 issues

Sprint 2 6 issues
Start: 10 Aug 2015 — Release: 9 Oct 2015


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


Backlog 49 issues

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

Example of Agile Project Management: JIRA Software

www.utm.my





Teams in Space
Scrum: Teams in Space ▾

Backlog

Agile board

Releases

Reports

All issues

Components

Add-ons

PROJECT SHORTCUTS

Mars Team HipChat Room

Space Station Dev Roadmap

Teams in Space Org Chart

Orbital Spotify Playlist

Hyperspeed Bitbucket Repo

+ Add shortcut

TIS-70 Scrum Board


QUICK FILTERS: Critical partners Only my partners Recently updated

12 To do

2 In progress

3 Done

▼ TIS Developer Love 3 issues





TIS-37

↑ Service should return prior trip details and info

SeeSpaceEZ plus

2







TIS-10

↑ Bad JSON data coming back from hotel API

SeeSpaceEZ plus







TIS-8

↑ Requesting flights is now taking > 5 seconds

SeeSpaceEZ plus




▼ Everything Else 21 issues




TIS-68

↑ Homepage footer uses an inline style-should use class

Large Team Support







TIS-20

↑ Engage Saturn Shuttle lines for group tours

Space Travel Partners


3





TIS-12


⊘ Create 90 day plans for all departments in Mars office



TIS-17

↑ Engage Saturn's Rings Resort as preferred


Space Travel Partners




TIS-56

↑ Add pointer to main css file to create child themes

Large Team Support






TIS-45

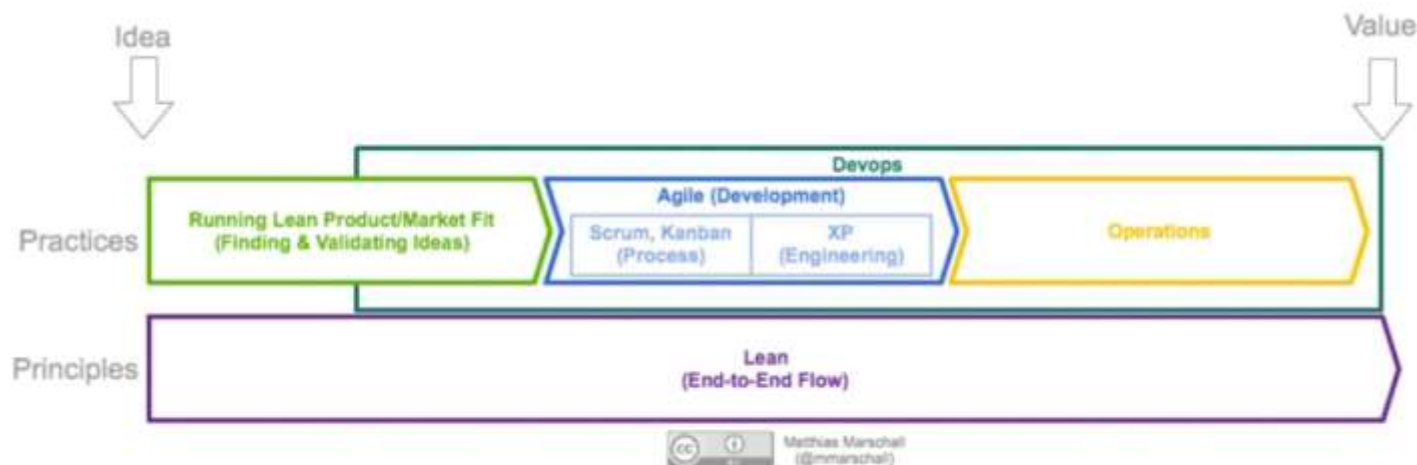
↑ Email non registered users to sign up with TIS

SeeSpaceEZ plus



DevOps: Agile & Lean to Operations Work

- DevOps: a practice of **operations** and **developments** engineers participating together in the entire service lifecycle from design through development process to production support

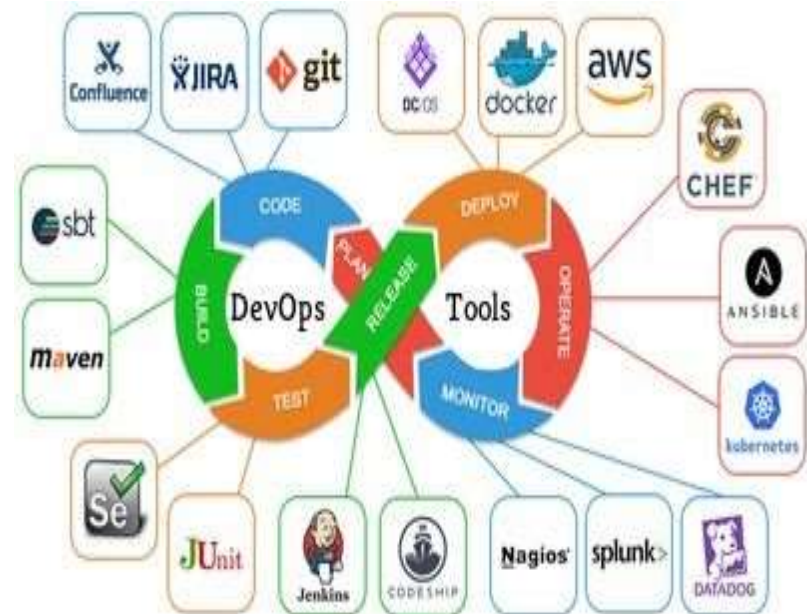


Source: <https://theagileadmin.com/what-is-devops/>

DevOps and Tools

www.utm.my

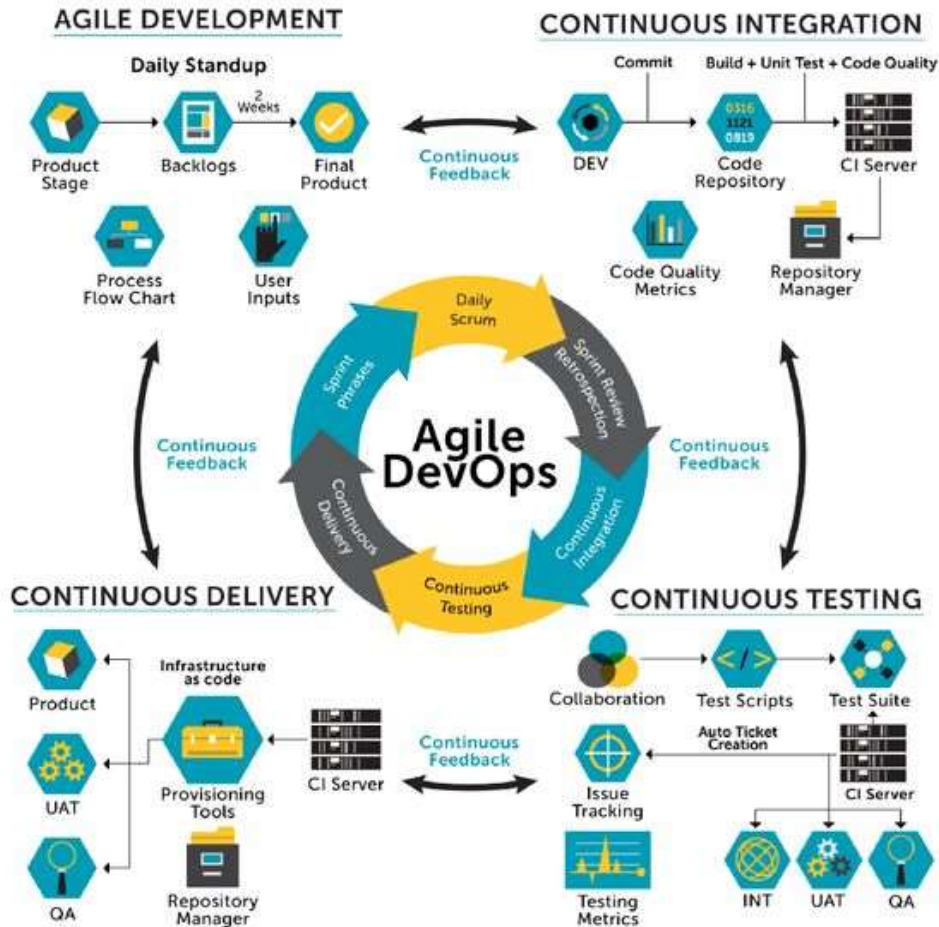
- DevOps is a set of practices that emphasizes the **collaboration and communication** of both software developers and other information technology (IT) professionals, while **automating the process** of software delivery and infrastructure changes, its implementation can include the definition of the **series of tools** used at various stages of the lifecycle
- There is no one product that can be considered a single DevOps tool



Source: <https://dzone.com/articles/an-agile-introduction-to-devops-what-is-devops-any>

Agile DevOps...

www.utm.my



<https://www.educba.com/agile-devops/>

- Agile along with DevOps has a **collaborative working style**, irrespective of the method implemented
- Both the methodologies rely on **continual feedback and routine updates** about the work progress from internal and external stakeholders
- Both Agile and DevOps focus on **developing the product at a fast pace**, by keeping smaller teams and using a risk-free approach
- Both methods **adapt to the business requirements** and continually improve the products to fulfill customer expectations

Agile DevOps

www.utm.my

- Agile DevOps is an **extended agile methodology** for product development where DevOps is a succession to agile and not a replacement
- In a nutshell, Agile works towards software development by making **continuous alterations**, adapting and developing products as per customer expectations
- While DevOps through **automated processes** and **bug detection** at an early stage of software development focusses primarily on deploying products of high quality

Source: <https://www.educba.com/agile-devops/>

SCALING AGILE METHODS

Practical Problems with Agile Methods

www.utm.my

- The informality of agile development is **incompatible with the legal approach** to contract definition that is commonly used in large companies
- Agile methods are **most appropriate for new software development** rather than software maintenance
- Yet the majority of software costs in large companies come from maintaining their existing software systems
- Agile methods are designed for **small co-located teams** yet much software development now involves worldwide distributed teams

Scaling Agile Methods

www.utm.my

- Agile methods have proved to be **successful for small and medium** sized projects that can be developed by a small co-located team
- However, the usage of Agile methodologies in large corporations has been growing steadily over the past decade. Here are some statistics based on various surveys and reports:
 1. According to the 15th Annual State of Agile Report by VersionOne, 97% of respondents indicated that their organization is practicing Agile development methods, up from 84% in 2015
 2. A survey by Agile Alliance found that 79% of respondents reported using Agile methodologies in large organizations, up from 64% in 2015

Scaling Agile Methods

www.utm.my

3. A report by McKinsey & Company found that 94% of respondents reported that their organizations had adopted Agile methods, and 76% reported that they had successfully scaled Agile beyond a single team
4. According to a survey by Deloitte, 68% of respondents reported that their organization had adopted Agile methodologies, up from 42% in 2016
5. The 2020 State of Agile Marketing Report found that 63% of respondents reported using Agile methodologies in large organizations, up from 50% in 2019

Scaling Agile Methods

www.utm.my

- It is sometimes argued that the success of these methods comes because of **improved communications** which is possible when everyone is working together
- **Scaling up agile methods** involves changing these to cope with larger, longer projects where there are multiple development teams, perhaps working in different locations

Large Systems Development...

www.utm.my

- Large systems are usually collections of separate communicating systems, where **separate teams** develop each system. Frequently, these teams are working in different places, sometimes in different time zones.
- Large systems are ‘brownfield systems’, that is they **include and interact with a number of existing systems**. Many of the system requirements are concerned with this interaction and so don’t really lend themselves to flexibility and incremental development.
- Where several systems are integrated to create a system, a significant fraction of the development is **concerned with system configuration** rather than original code development.

Large System Development

www.utm.my

- Large systems and their development processes are often constrained by **external rules and regulations** limiting the way that they can be developed.
- Large systems have a **long procurement** and development time. It is **difficult to maintain coherent teams** who know about the system over that period as, inevitably, people move on to other jobs and projects.
- Large systems usually have a **diverse set of stakeholders**. It is practically impossible to involve all of these different stakeholders in the development process.

Scaling Out and Scaling Up

www.utm.my

- ‘Scaling **up**’ is concerned with using **agile methods for developing large software systems** that cannot be developed by a small team.
- ‘Scaling **out**’ is concerned with how **agile methods can be introduced across a large organization** with many years of software development experience.
- When scaling agile methods, it is essential to maintain **agile fundamentals**: Flexible planning, frequent system releases, continuous integration, test-driven development and good team communications.

Scaling Up to Large Systems

www.utm.my

- For large systems development, it is not possible to focus only on the code of the system. You need to do **more up-front design and system documentation**.
- Cross-team communication mechanisms have to be designed and used. This should involve **regular phone and video conferences** between team members and frequent, short electronic meetings where teams update each other on progress.
- Continuous integration, where the whole system is built every time any developer checks in a change, is practically impossible. However, it is essential to **maintain frequent system builds and regular releases** of the system.

Scaling Out to Large Companies

www.utm.my

- Project managers who do not have experience of agile methods may be **reluctant to accept the risk** of a new approach.
- Large organizations often have **quality procedures and standards** that all projects are expected to follow and, because of their **bureaucratic nature**, these are likely to be incompatible with agile methods.
- Agile methods seem to work best when **team members have a relatively high skill level**. However, within large organizations, there are likely to be a wide range of skills and abilities.
- There may be **cultural resistance to agile methods**, especially in those organizations that have a long history of using conventional systems engineering processes.

Agile Methods and Software Maintenance

www.utm.my

- Most organizations **spend more on maintaining existing software** than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development.
- Two key issues:
 - Are systems that are developed using an agile approach **maintainable**, given the emphasis in the development process of minimizing formal documentation?
 - Can agile methods be **used effectively for evolving a system** in response to customer change requests?
- Problems may arise if original development team cannot be maintained.

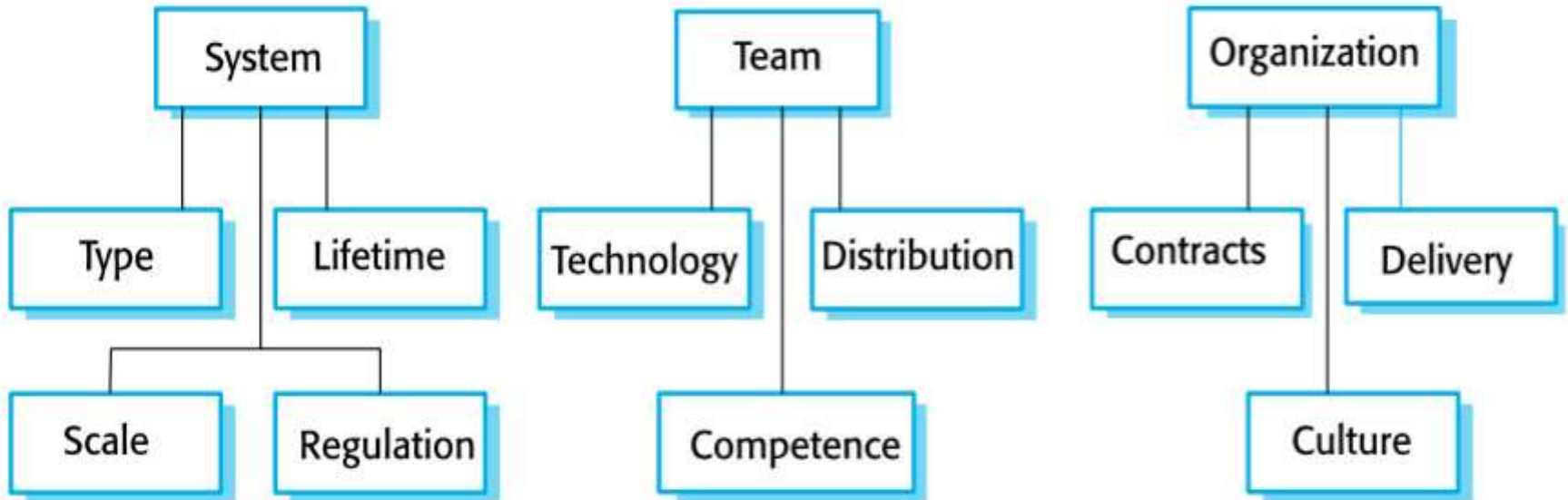
Agile Maintenance

www.utm.my

- Key **problems** are:
 - Lack of product documentation
 - Keeping customers involved in the development process
 - Maintaining the continuity of the development team
- Agile development **relies on the development team** knowing and understanding what has to be done
- For **long-lifetime systems**, this is a real problem as the original developers will not always work on the system

Agile and Plan-Based Factors

www.utm.my



Agile and Waterfall: AgiFall

www.utm.my

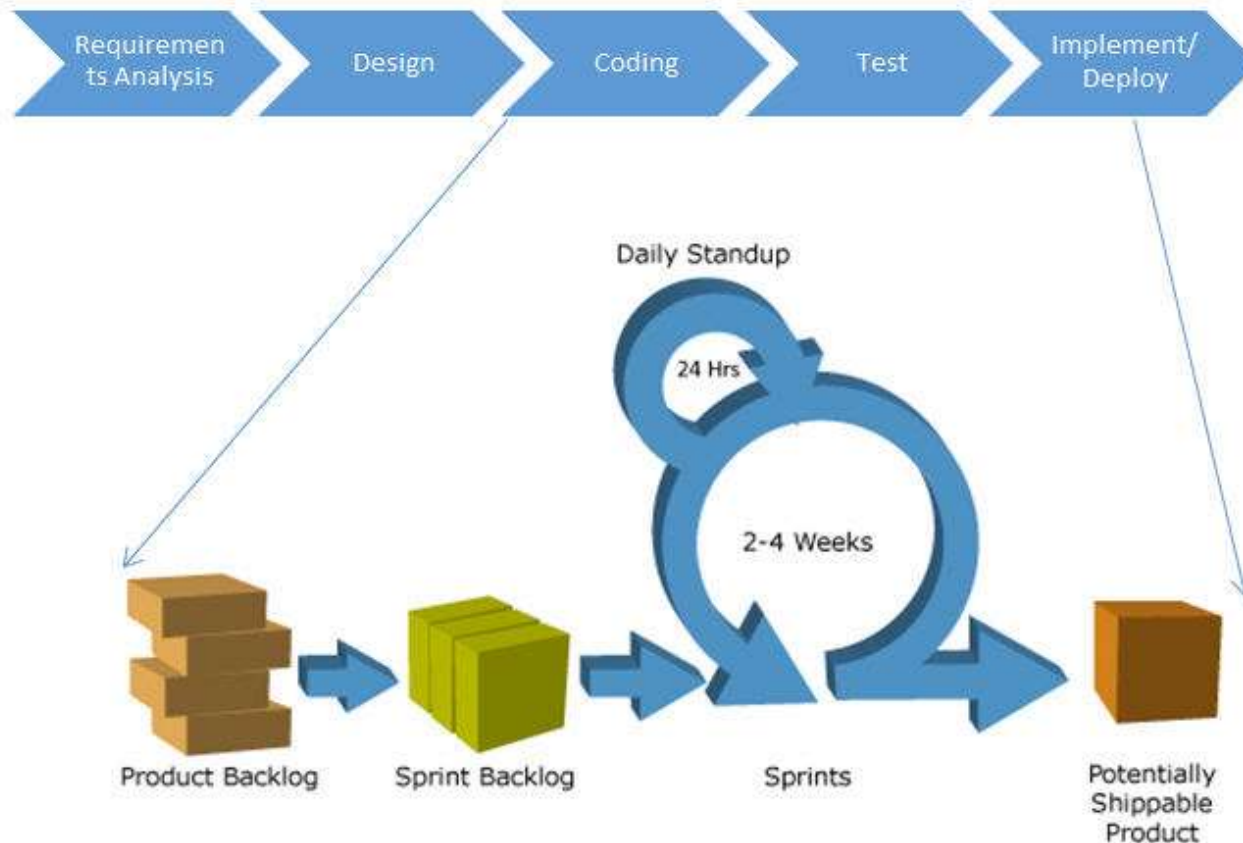
- How to make Agile and Waterfall work together?

Waterfall role	Meet half way	Agile role
Sponsor	Same	Sponsor
Product manager	Product manager would typically be accountable for the whole product or product line, and oversee potentially many product owners, or act as a product owner herself.	Product owner
Project manager	Project management would be elevated to oversee programs and dependencies of projects, leaving teams to manage more autonomously.	Development team
Project methodology champion (if one exists)	Same idea here, if either practices are applied, care should be taken to apply with true understanding of why they are applied to not dilute the intent and effect.	Scrum master (Scrum being by far the most widely adopted agile framework, hence this role used here)
End users	Active involvement from the beginning to the end. Traditional stage gates with wider presentation, key roles from core stakeholder groups involved continually.	End users
Other stakeholders	Address and respect the product owner.	Other stakeholders

Source: <https://www.itforbusiness.org/content/uploads/2016/01/Reconcile-Waterfall-and-Agile-for-Long-Term-Best-Results.pdf>

Agile-Waterfall Hybrid

www.utm.my



Collaborative (Hybrid) Model

Source: <http://www.softwaretestinghelp.com>

DSDM

www.utm.my

- **DSDM** (Dynamic Systems Development Method) which is specifically designed for large and complex software development projects.
- Here are some ways in which DSDM is suitable for large system development:
 - o **Incremental development:** emphasizes incremental development, which means that the project is broken down into smaller, manageable chunks that are developed and tested in iterations
 - o **Timeboxing:** uses timeboxing to manage the project schedule. It involves setting fixed deadlines for each iteration, which helps to ensure that the project stays on track and that deliverables are produced on time.
 - o **Emphasis on collaboration:** emphasizes collaboration between stakeholders, including the development team, the project sponsor, and end-users to ensure that the requirements are well-understood and meets the needs of the stakeholders.
 - o **Flexible approach:** flexible and can be tailored to the specific needs of the project. This is important in large system development, where the requirements and scope can change rapidly
 - o **Focus on quality:** emphasizes the importance of quality throughout the development process to ensure that the product meets the required quality standards

DSDM

www.utm.my

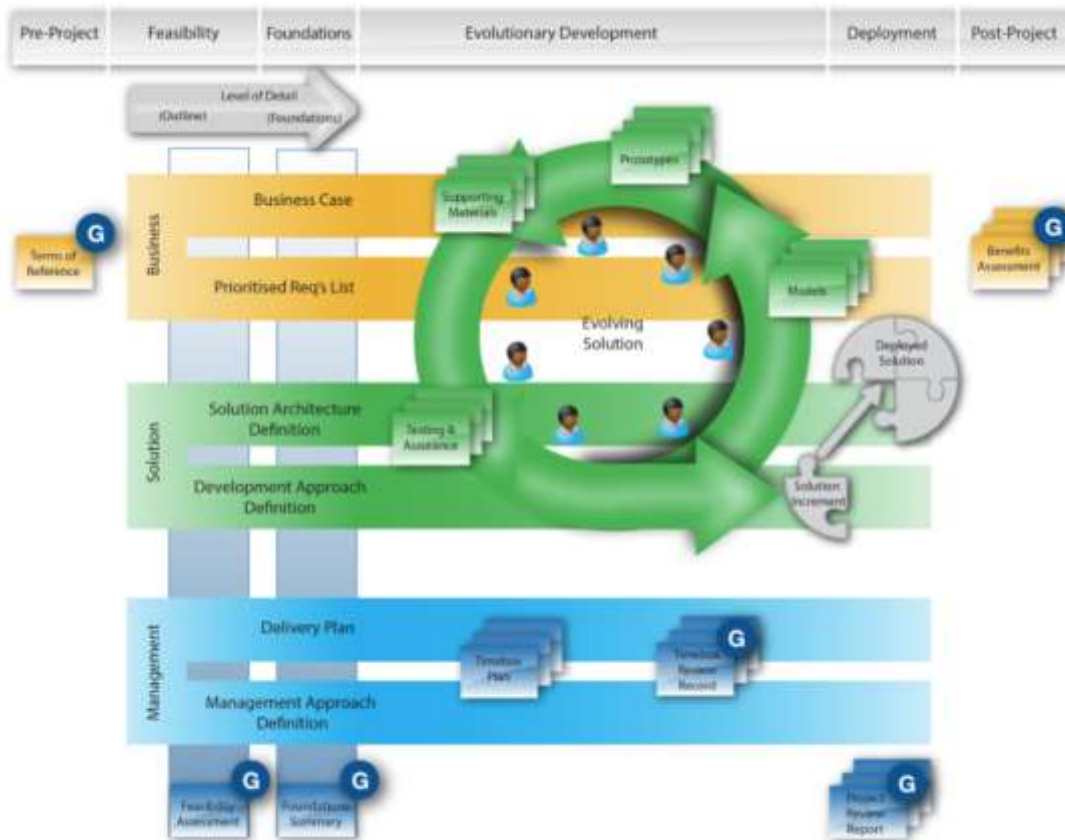


Figure 8a: DSDM products

Source: <http://www.softwaretestinghelp.com>

innovative • entrepreneurial • global

Key points

www.utm.my

- Agile methods are **incremental development** methods that focus on rapid software development, **frequent releases** of the software, **reducing process overheads** by minimizing documentation and producing high-quality code
- Agile development **practices** include:
 - User stories for system specification
 - Frequent releases of the software
 - Continuous software improvement
 - Test-first development
 - Customer participation in the development team

Key points

www.utm.my

- **Scrum** is an agile method that provides a project management framework
 - It is centered round a set of sprints, which are fixed time periods when a system increment is developed
- Many practical development methods are a **mixture of plan-based and agile development**
- Scaling agile methods for large systems is difficult
 - Large systems need up-front design and some documentation and organizational practice may conflict with the informality of agile approaches