

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

200
1822 - 2022

Man Geni Addysg Uwch yng Nghymru
Where Higher Education Began in Wales



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

200
1822 - 2022

Software Development & Design

*Network Programmability
(ACCB5029)*

Applied Computing - WISA

*Man Geni Addysg Uwch yng Nghymru
Where Higher Education Began in Wales*

www.uwtsd.ac.uk



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

200
1822 - 2022

Previously,

- ✓ Overview
- ✓ Review Linux and Python Skills
- ✓ Devnet Developer Environment

This week,

- ✓ Software Development & Design

Previous week's Tutorial and Independent learning

- Setup Lab environments
- Review Linux and Python

This week's Tutorial and Independent learning

- Review Python installation, PIP, and Python virtual environments

Man Geni Addysg Uwch yng Nghymru
Where Higher Education Began in Wales

www.uwtsd.ac.uk

Previously, we discussed about Devnet skills, and course work briefing. I hope you had a clarity on network programmability and its importance. Your lab tasks to install DEVASC VM and exploring its GUI could have given you a better understanding. I would advice you to continue your independent learning activities (CCNA Devnet course and Explore different Devnet resources). This week, lets continue to understand Software Development life cycle and its development methodologies.

- ✓ Independent learning
 - Explore Devnet Resources
 - Continue with Devnet Course in Cisco Networking Academy



Software Development and Design

- The software development process is also known as the software development life cycle (SDLC).
- SDLC is **more than just coding** and includes **gathering requirements**, creating a **proof of concept**, **testing**, and **fixing bugs**.
- Although the **waterfall methods** is still widely used today, it's gradually being superseded by **more adaptive, flexible methods that produce better software, faster, with less pain**. These methods are collectively known as “**Agile development.**”



SDLC

- The **requirements and analysis phase** involves exploring the stakeholders' current situation, needs and constraints, present infrastructure, and so on, and **determining the problem to be solved by the software.**
- After gathering the requirements, the team analyzes the results to determine the following:
 - ✓ Is it possible to develop the software according to these requirements, and can it be done on-budget?
 - ✓ Are there any risks to the development schedule, and if so, what are they?
 - ✓ How will the software be tested?
 - ✓ When and how will the software be delivered?
- At the conclusion of this phase, the classic waterfall method suggests creating a Software Requirement Specification (SRS) document, which states the software requirements and scope, and confirms this meticulously with stakeholders.

SDLC

- During the **Design phase**, the **software architects and developers design the software** based on the provided SRS.
- At the end of the phase, the team creates High-Level Design (HLD) and Low-Level Design (LLD) documents.
- The **implementation phase** is also called the **coding or development phase**.
- As all the components and modules are built during this phase, it is the longest phase of the life cycle.
- At the end of the phase, the functional code that implements all customer's requirements is ready to be tested.

Testing

- In this phase, code is installed in the test environment
- Functional testing, integration testing, performance testing and security testing is performed.
- Testing continues until all the codes are bug free and pass all the tests. At the end of this phase, a high quality, bug-free, working piece of software is ready for production.

Deployment

- During this phase, the software is installed into the production environment.
- At the end of the phase, the product manager releases the final piece of software to end users.

Maintenance

- During the maintenance phase, the team:
 - ✓ Provides support to customers
 - ✓ Fixes bugs found in production
 - ✓ Works on software improvements
 - ✓ Gathers new requests from the customer
- At the end, the team works on the next iteration and version of the software.



Software Development Methodologies

A software development methodology is also known as **Software Development Life Cycle model**.

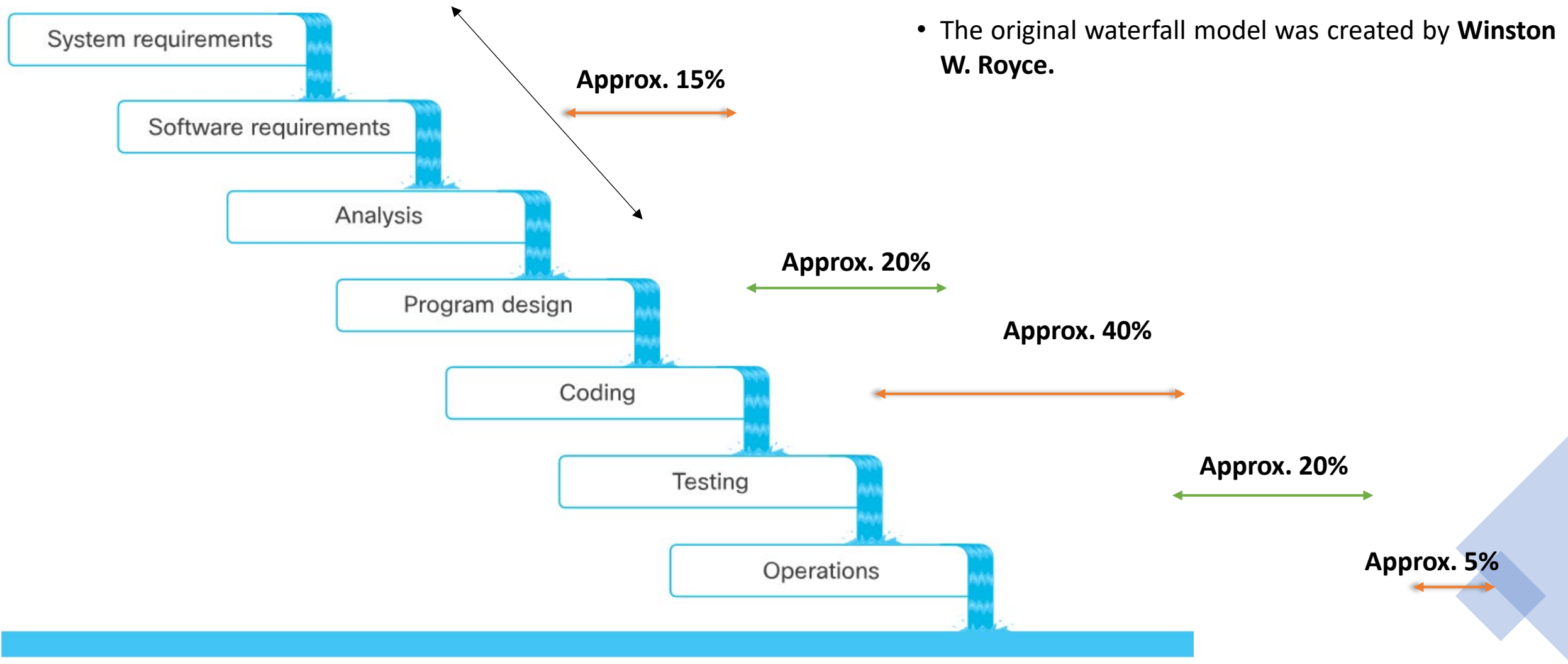
The three most popular methodologies are:

- Waterfall
- Agile
- Lean

The type of methodology to be used depends on the:

- Type of the project
- Length of the project
- Size of the team.

Software Development Methodologies – Waterfall Software Development



Software Development Methodologies – Agile Software Development

Agile method is **flexible and customer-focused**.

A group of 17 software developers produced the Manifesto for Agile Software Development, also known as the **Agile Manifesto**, in **2001**.

According to the Agile Manifesto, the values of Agile are:

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



1st Iteration



2nd Iteration



3rd Iteration



4th Iteration

Agile Manifesto Principles			
Customer focus	Collaboration	Working software	Simplicity
Embrace change and adapt	Motivated teams	Work at a sustainable pace	Self-organizing teams
Frequent delivery of working software	Face-to-face conversations	Agile environment	Continuous Improvement

Software Development Methodologies – Agile Methods

Agile Scrum

The Scrum focuses on *small, self-organizing teams that meet daily for short periods and work in iterative sprints*.

Kanban

The Kanban focuses on *minimizing the bottlenecks and improve the workflow*. It visualizes the work on its go by tracking them.

Lean

The Lean method emphasizes on *elimination of wasted effort in planning and execution*, and reduction of programmer cognitive load.

Extreme Programming (XP)

XP deliberately addresses the *specific kinds of quality-of-life issues* faced by the software development teams.

Software Development Methodologies – Agile Methods

Feature-Driven Development (FDD)

FDD prescribes that software development should proceed in terms of an ***overall model, broken out, planned, designed, and built feature-by-feature.***

Backlog The backlog consists of all the features of the software, in a prioritized list.

Scrum Teams

- Scrum teams are **cross-functional, collaborative, self-managed and self-empowered.**
- The scrum teams should **not be larger than 10 individuals.**
- The scrum master should have a **daily stand-up meeting with the team** at a fixed time everyday for not more than 15 minutes.
- The goal is to go over **important tasks** that have been finished, are in progress, or are about to be started.

Software Development Methodologies – Agile Methods

Sprints

A sprint is a ***specific duration***, usually between 2-4 weeks, during which, each team takes on as many tasks (also known as user stories) as they feel they can accomplish.

- When the sprint is over, the software should be working and deliverable.
- The duration of the sprint is determined before the process begins and should rarely change.

User stories

A user story is a simple statement of what a user (or a role) needs, and why. Each user story should be small enough that a single team can finish it within a single sprint.

The suggested template for a user story is:

As a <user|role>, I would like to <action>, so that <value|benefit>

Software Development Methodologies – Lean Software Development

Lean software development is based on **Lean Manufacturing principles**, which are focused on minimizing waste and maximizing value to the customer.

The seven principles of lean, given in the book *“Lean Software Development: An Agile Toolkit,”* are as follows:

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. Optimize the whole

Empower the Team

Each person must be allowed to make decisions in the area of their own expertise.

Build Integrity In

Integrity for the software is when the software addresses the customer's needs as well as maintains the usefulness for the customer.

Optimize the Whole

The software must be built cohesively. The value of the software will suffer if each expert only focuses on their expertise and doesn't consider the ramifications of their decisions on the rest of the software.

Software Design Patterns

- Software design patterns are best practice solutions for solving common problems in software development.
- Design patterns are **language-independent**.
- In 1994, Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (known as the Gang of Four (GoF)) published a book called ***Design Patterns - Elements of Reusable Object-Oriented Software***.
- Patterns identified are:
 - Program to an interface, not an implementation.
 - Favor object composition over class inheritance.
- Software design patterns have already been proven to be successful, so using them can speed up development.

Software Design Patterns

The Gang of Four divided patterns into three main categories:

- Creational
 - Structural
 - Behavioural
-
- They listed 23 design patterns.
 - Two of the most used design patterns are:
 - ✓ The Observer design pattern (a Behavioural design pattern)
 - ✓ The Model-View-Controller (MVC)

Software Design Patterns – Observer Design Pattern

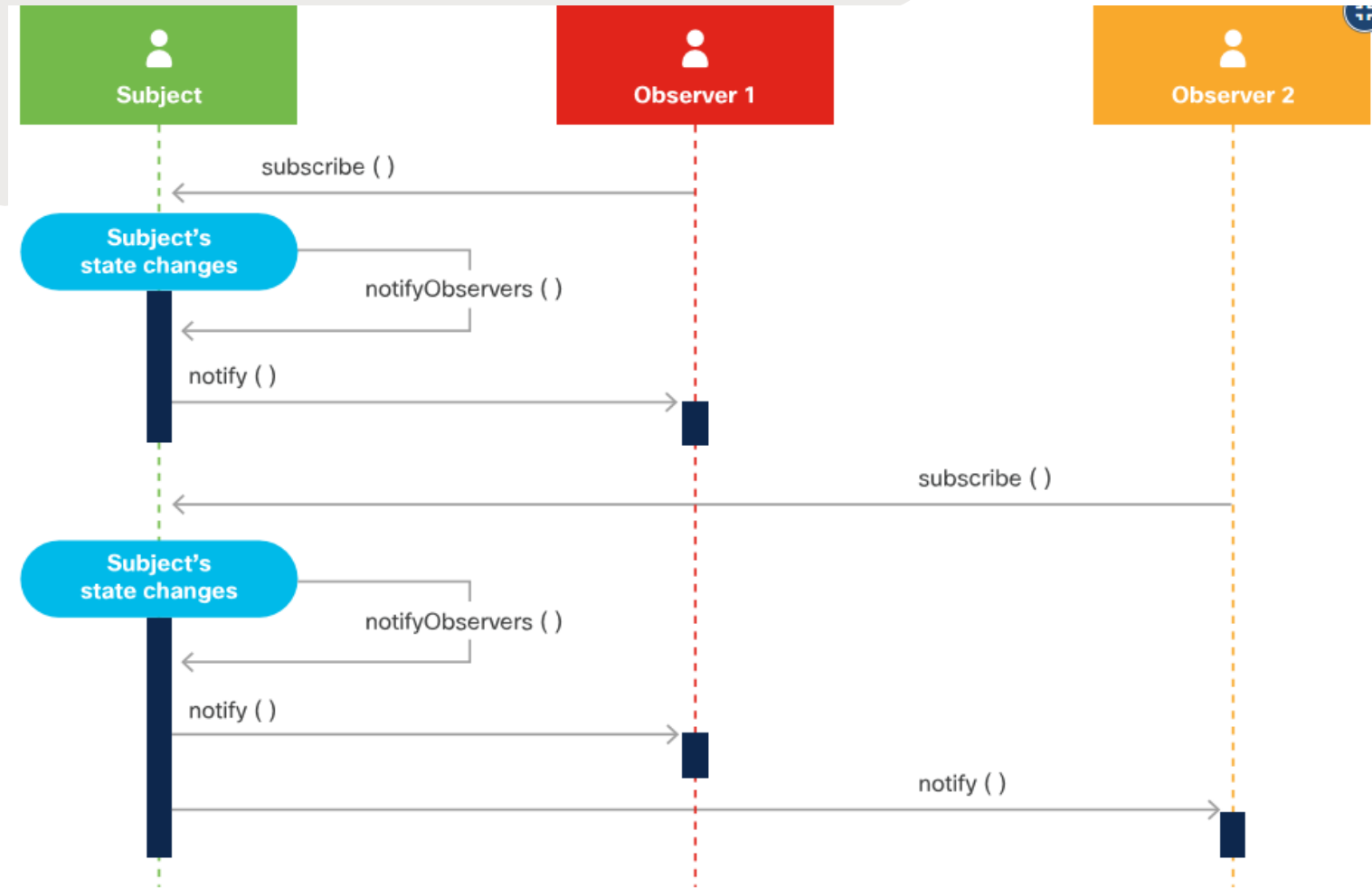
The observer design pattern is a subscription notification design that lets objects receive events when there are changes to an object they are observing.

To implement this subscription mechanism:

- The subject must have the ability to store a list of all its observers.
- The subject must have methods to add and remove observers.

The benefit of the observer design pattern is that observers can get real time data from the subject when a change occurs.

Subscription mechanisms always provide better performance than other options, such as polling.



Software Design Patterns

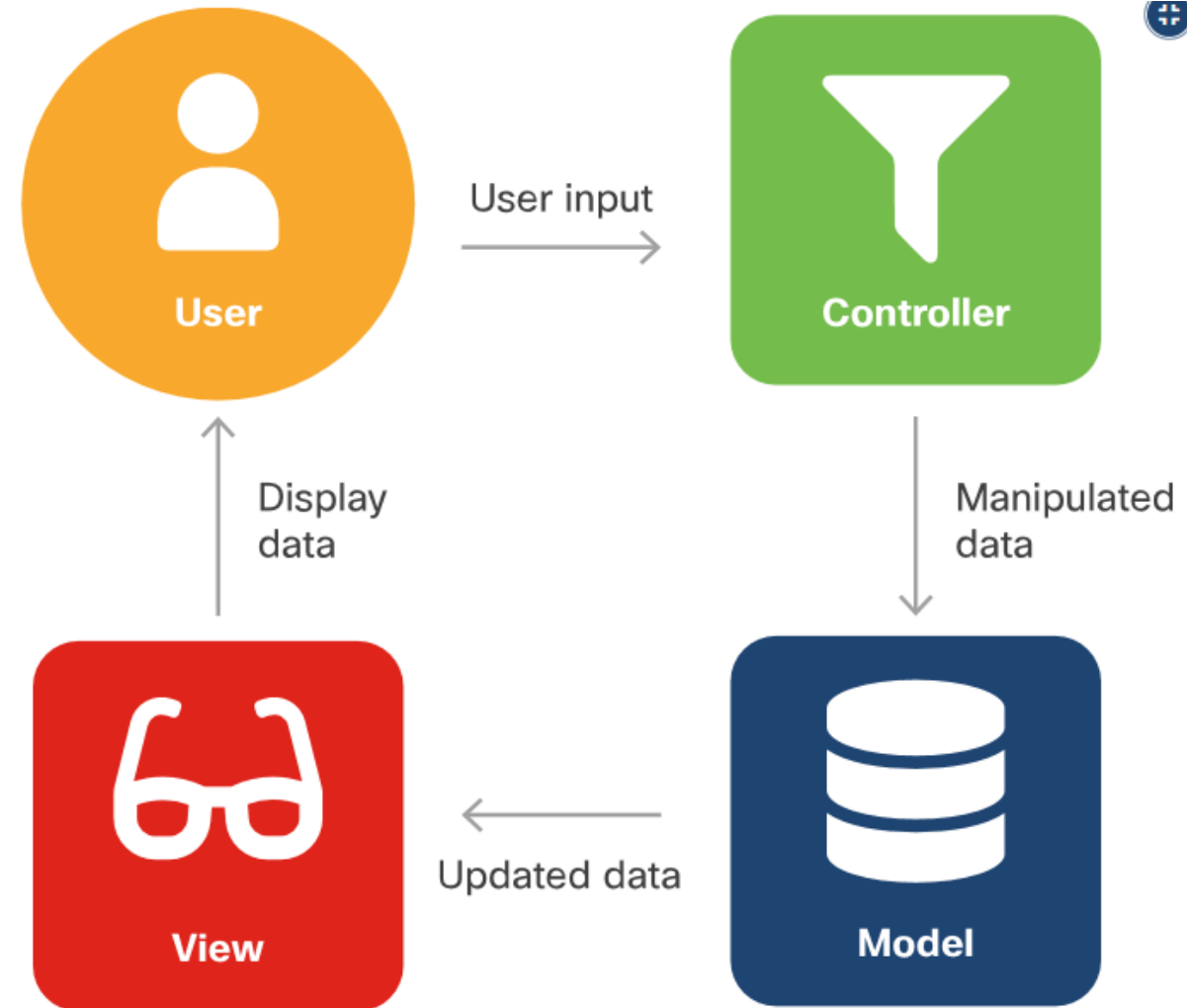
– Model-View-Controller

The Model-View-Controller (MVC) design pattern aims to simplify development of applications that depend on graphic user interfaces.

MVC abstracts code and responsibility into three different components:

- **Model:** The model is the application's data structure and is responsible for managing the data, logic and rules of the application. It gets input from the controller.
- **View:** The view is the visual representation of the data.
- **Controller:** The controller is the middleman between the model and view. It takes in user input and manipulates it to fit the format for the model or view.

The benefit of MVC is that each component can be built in parallel.





Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

200
1822 - 2022

Any Questions?

Network Programmability
(ACCB5029)

Applied Computing - WISA

Man Geni Addysg Uwch yng Nghymru
Where Higher Education Began in Wales

www.uwtsd.ac.uk