# CMPE 220

## Class 21 – Frameworks

# Types of System Software

# Building Software



High Level Language Source Code (e.g. C++) → **Compiler** → Assembly Language Source Code → **Assembler** → Binary Machine Code

*optional*

**Linker** → Executable Machine Code → **Loader** → In-Memory Code → **Hardware Execution**
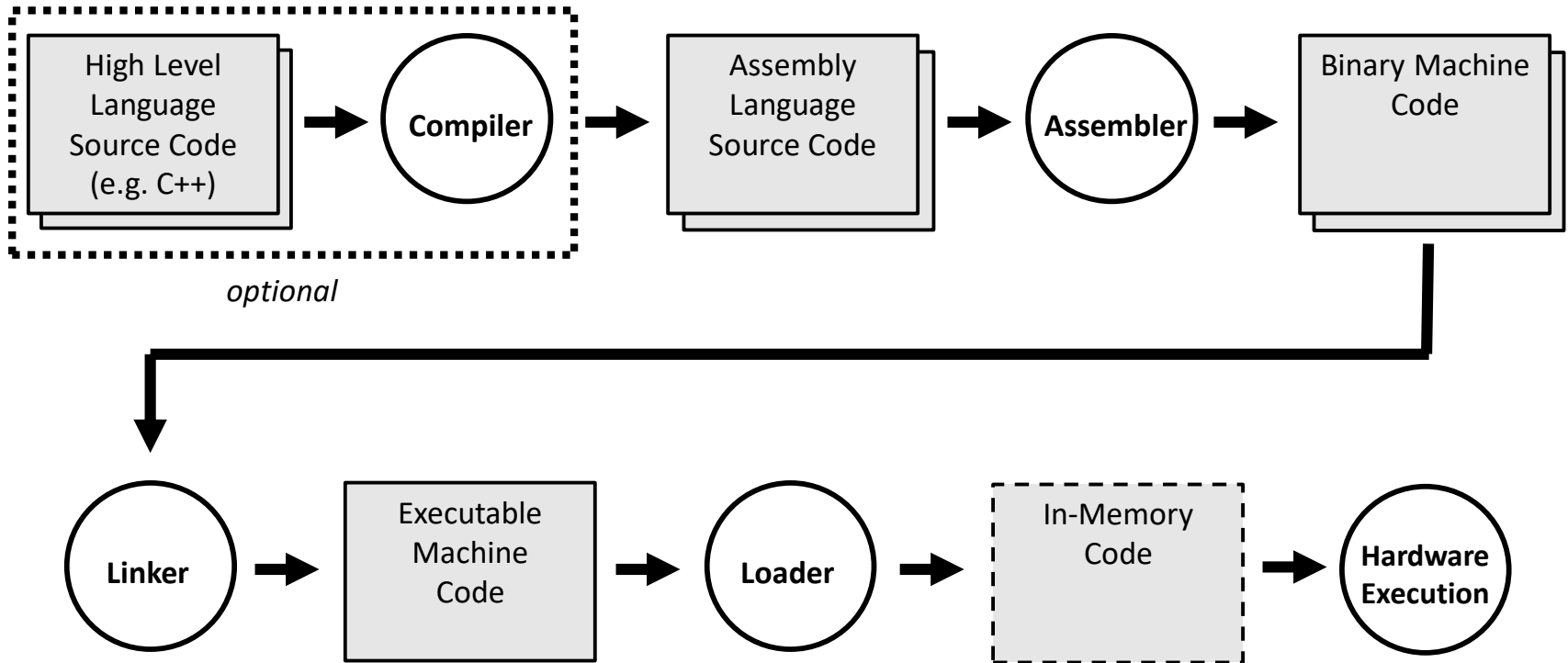
# Additional Development Tools

- Cross-Development Tools
- Emulators
- Integrated Development Environments (IDEs)
- Debuggers

# Servers and Services

**Servers**

- Web server
- DNS server
- FTP server
- Email server
- Database server
- Web application services
- Windowing systems

**Protocols**

- IP – Internet Protocol
- TCP/IP
- UDP
- FTP
- SMPT
- IMAP
- POP
- MySQL

# Operating Systems

**OS Functions**

1. Process Management
   - Interprocess Communications
2. Input / Output (I/O) Management
3. Memory Management
4. File System Management
5. System Functions and Kernel Mode
6. User Interaction – (maybe)
7. Networking

# OS Types

- Traditional Operation Systems
- Distributed Systems
- Networked Operating Systems
- Object Oriented Systems
- Virtual Systems

# Frameworks

# What is a Framework?

- A software framework used by software developers to implement the standard <u>structure</u> of application software
- A library
- A set of object classes

# Pros and Cons of Frameworks

## **Pros**
- A proven architecture
- Built-in functionality
- Developer communities

## **Cons**
- Performance penalty
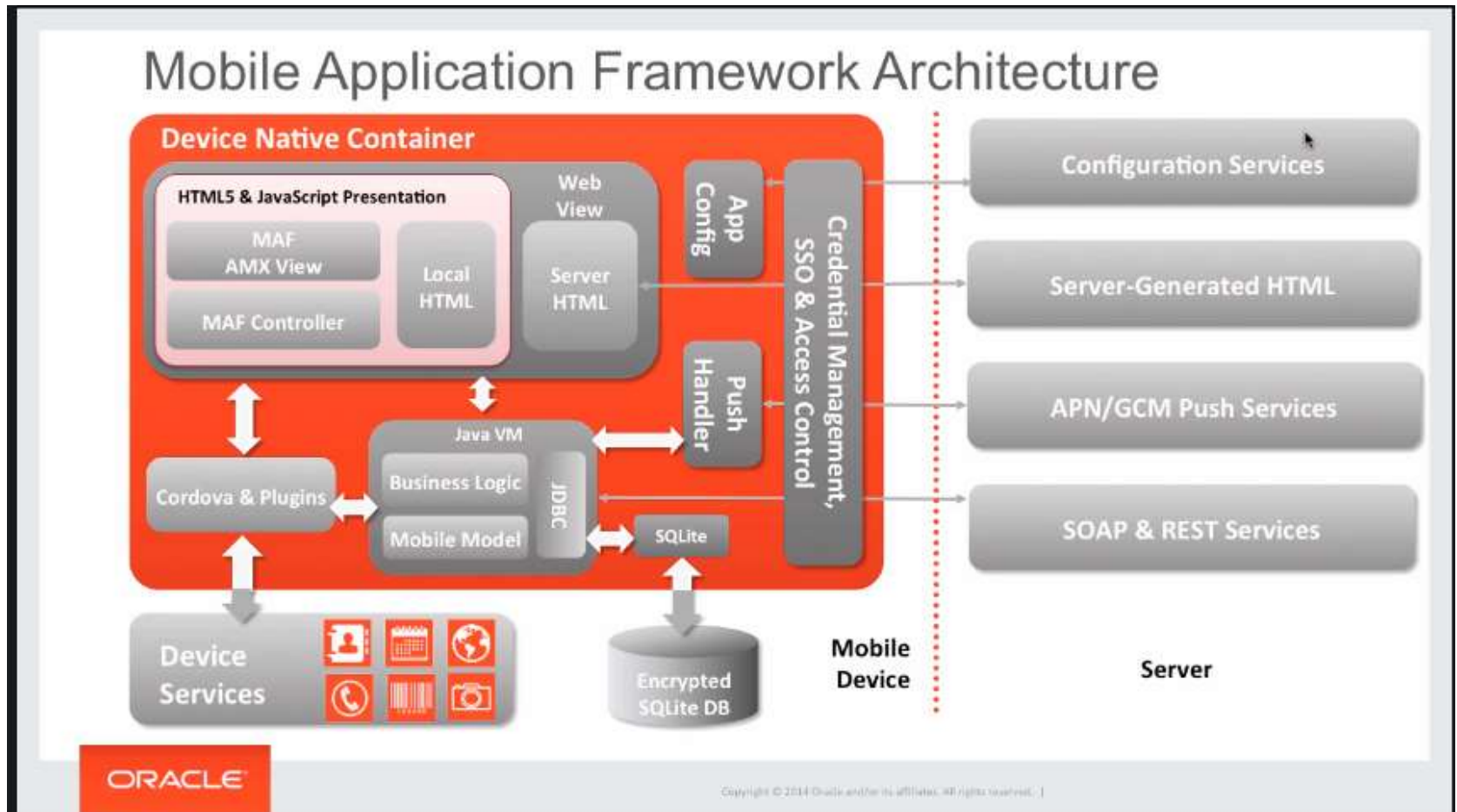- Potential obsolescence

# Early Frameworks

- Xerox PARC / SmallTalk – mid-70s?
- MacApp – 1985
- Microsoft Foundation Classes (MFC) – 1992
- ColdFusion (web application framework) - 1995

# Types of Frameworks

- GUI Application Frameworks
- Web Application Frameworks (server)
- Web Application Frameworks (client)
- Mobile Application Frameworks (iPhone / Android)

# System Software?

# Development Using Frameworks

- The framework consists of a set of object classes
- To write applications, developers *subclass* the framework classes, and *override* their methods
- Most frameworks use a Model-View-Controller (MVC) architecture
- Top-level class is the *App*
  - Override the *init* method to set up the application
  - Create other objects used by the app
  - Register callbacks

# Model-View-Controller Architecture (MVC)

- MVC is an ideal architecture for GUI applications.

- Design goal: Identify which application components are <u>model</u>, <u>view</u>, or <u>controller</u>.
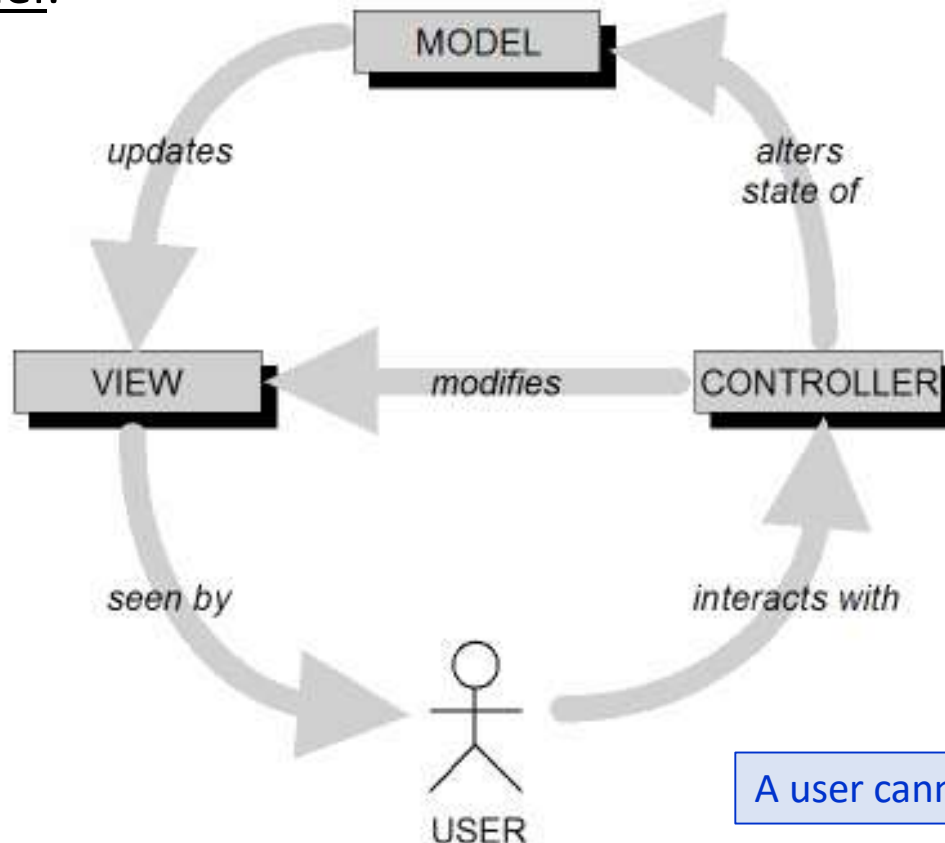


A user cannot directly modify the model.

# MVC Implementation: Loose Coupling

- Keep the implementations of the three object types separate.

- Each type of objects does not depend on how the other types are implemented.

- Your application is
  - easier to develop and maintain
  - faster to develop
  - more <u>robust</u> (resilient to runtime errors)

# MVC <u>Model</u> Objects

- Represent the <u>persistent information</u> maintained by your application.
  - State


- The information can be kept in a database.

# MVC <u>View</u> Objects

- View objects represent <u>user interface components</u>.
  - Input components such as text fields and checkboxes.

- In each use case, users interact with at least one view object.

- A view object <u>collects information</u> from users in a form that the model and controller objects can use.

# MVC Controller Objects

- Coordinate the model and view objects.

  - Often have <u>no physical counterpart</u> in the real world.
  - Collect information from view objects for dispatch to model objects.
  - This is how user-entered data can update the model.

# Advantages of MVC Architecture

- Classes are Cohesive
- Classes are Loosely Coupled
  - Principle of Information Hiding
- Functionality and data are Encapsulated
  - Changes are isolated and localize
- Provides a clear separation of business logic, UI (display) logic, and input logic.
- Each component can be tested separately

# Model-View-ViewModel (MVVM)

- MVVM is an architectural pattern, invented by Microsoft architects Ken Cooper and Ted Peters

- MVVM (Model-View-ViewModel) cleanly separates the business logic of an application from the user interface

- The ultimate goal of MVVM architecture is to make the view completely independent from the application logic

# Model-View-ViewModel (MVVM)
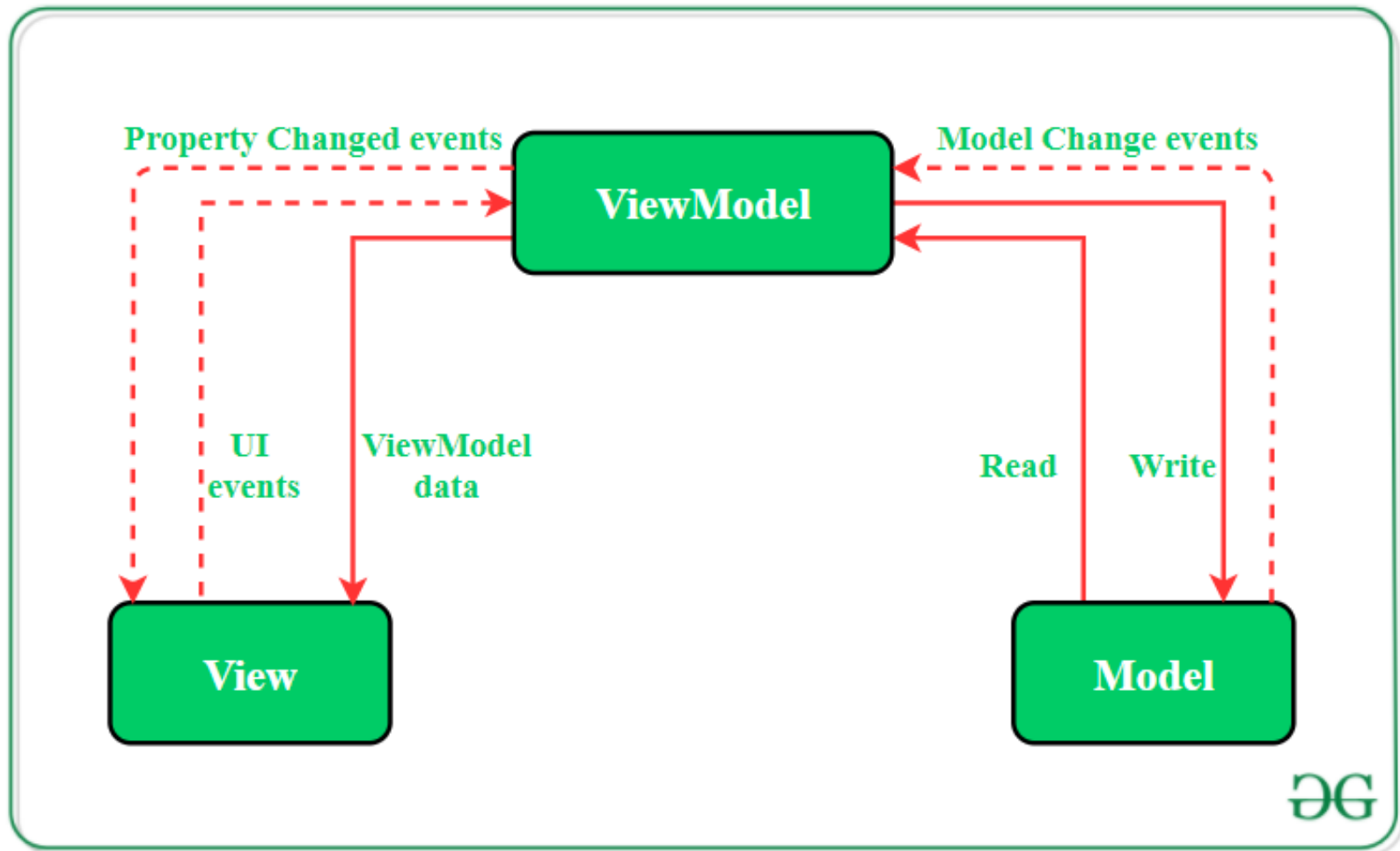
# Advantages of MVVM

- MVVM has a **unidirectional data and dependency flow**
- Dependency is one way, thus it is easier to decouple it needed

# MVC versus MVVM

- In MVC, the controller is the entry point to the Application
- In MVVM, the view is the entry point to the Application
- In the MVC Model, the Model itself serves as the mediator between the View and the Model
- In the MVVM model, the ViewModel serves as the mediator between the View and the Model, and can have a one-to-many relationship with views

  Because the ViewModel is responsible for exposing the data and logic of the Model to the View, it can be easier to test the ViewModel separately from the View.

# Inversion of Control

- The *framework* controls the execution flow

- The developer registers callback functions, mostly as event handlers, with the framework

- After initializing objects and registering callbacks, the developer surrenders control to the framework

- The framework invokes the callback functions at the appropriate times, such as in response to events

# Callbacks

- Callback functions are registered to:
  - Process events
    - Button clicks
    - Menu selections
    - Timer events
  - Data Transformation
    - Validate
    - Filter
    - Format

# Popular Frameworks

**GUI / Windowing**

- Chromium (C/C++)
- GTK (C/C++)
- MKS (C/C++)
- QT (C/C++)
- wxWidgets (C/C++)

- Apache Pivot (Java)
- JavaFX (Java)
- Swing (Java)

- Bottle (Python)
- Django (Python)
- CherryPy (Python)
- Flask (Python)
- Web2Py (Python)

# Popular Frameworks

**Web Applications**

- Angular – JavaScript
- CodeIgniter – PHP
- Django – Python
- Laravel – PHP
- Rails – Ruby (aka RoR or Ruby on Rails)
- Spring – Java

# Popular Frameworks

**Web Client Frameworks**

- Angular

- Backbone.js

- Ember

- jQuery

- React

- Svelte

- Vue.js

# Popular Frameworks

**Frameworks for Mobile Apps**

- Swiftic
- React Native
- Xamarin
- Ionic
- Mobile Angular
- Onsen UI
- Flutter
- Corona (Solar2d)

# Assignment 8

**Securing a software service**

- You've developed a new software service. It will be accessible over the Internet, using accounts created by the users.

- It will be hosted on a dedicated server that you manage.

- The data handled by this service is very sensitive.

- Write a short description of all the things you will do to secure your service.

- Submit this assignment as a pdf file.


- Due next Tuesday