ARCHITECTURE

Model-view-viewmodel (MVVM)

May 7, 2024

Muhammad Fozan Bin Mohsin | FA21-BSE-115

SOFTWARE DESIGN & ARCHITECTURE

Sir Mukhtiar Zamin

MVVM

MVVM stands for Model-View-ViewModel and it's a software design pattern used to separate the different aspects of an application. Here's a breakdown of the three parts:

- **Model:** This layer represents the data and the core business logic of your application. It handles things like data retrieval, manipulation, and validation. You can think of it as the brain of your application.
- **View:** This layer is all about the user interface (UI). It displays the data to the user and captures any user interaction. The View should be focused on presentation and not contain any business logic.
- **ViewModel:** This layer acts as a bridge between the Model and the View. It prepares the data from the Model in a way that's easy for the View to understand and display. It also handles user interactions from the View and updates the Model accordingly. Essentially, the ViewModel translates between the Model and the View.

There are several advantages to using MVVM architecture:

- **Separation of Concerns:** By separating the UI, data, and logic, MVVM makes your code more maintainable and easier to understand. Developers working on the UI don't need to worry about the business logic, and vice versa.
- **Testability:** Each layer of MVVM can be tested independently, making it easier to identify and fix bugs.
- **Reusability:** Since the View and ViewModel are decoupled from the specific data source, they can potentially be reused across different parts of your application.

Overall, MVVM is a popular design pattern for building clean, maintainable, and testable applications.

Success Example of MVVM

A successful example of MVVM architecture is Microsoft's WPF (Windows Presentation Foundation) toolkit, released in 2006. WPF is a UI framework for building desktop applications with rich graphics and user interactions.

Here's why MVVM is a good fit for WPF and contributes to its success:

- **Improved Maintainability:** WPF applications built with MVVM are easier to maintain because the UI logic is separated from the data and business logic. This allows developers to make changes to one layer without affecting the others.
- **Flexibility:** MVVM allows for greater flexibility in designing UIs. Developers can easily create reusable UI components that can be bound to different data models.
- **Testability:** The separation of concerns in MVVM makes WPF applications easier to test. Unit tests can be written to isolate and test the ViewModel logic independently of the UI.
- **Data Binding:** WPF supports data binding which allows the View to automatically update itself whenever the underlying data changes in the ViewModel. This simplifies development and reduces the need for manual code to update the UI.

Overall, MVVM's emphasis on separation of concerns and data binding makes it a powerful tool for building complex and maintainable WPF applications. This has likely contributed to WPF's continued use in building desktop applications.

Failure Example of MVVM

Unfortunately, pinpointing a specific software example of MVVM failure that resulted in a commercially unsuccessful product is difficult due to the following reasons:

1. **Multiple Factors Contribute to Failure:** A software product's success or failure depends on many factors beyond its architecture. Market competition, poor execution, or lack of a clear value proposition can all contribute to failure, making it hard to isolate MVVM as the sole culprit.
2. **Internal Development Information:** Specific details about internal development choices, like architecture patterns used, are not always publicly available.

However, we can discuss a hypothetical scenario where MVVM implementation might have contributed to a project's struggles:

- **Project: Fitness Tracker App (Year: 2020)**
    - **MVVM Implemented:** The development team adopted MVVM for a new fitness tracker app.
- **Reasons for Potential Failure:**

- **Over-engineered for Simplicity:** The app's core functionality was relatively simple - tracking steps, calories burned, etc. MVVM might have introduced unnecessary complexity to the development process.
- **Learning Curve Impact:** If the development team lacked prior MVVM experience, the learning curve could have slowed down development and increased costs.
- **Focus Shifted:** The focus might have shifted from core functionalities like accurate step tracking to implementing complex MVVM patterns, potentially delaying time to market.

Important to Note:

- This is a hypothetical scenario. MVVM can still be a good choice for fitness tracker apps, especially if the app has additional features like workout routines or social integrations.
- The key takeaway is that MVVM is not a one-size-fits-all solution. For simpler applications, the potential drawbacks of over-engineering and learning curve can outweigh the benefits.

Video Explanation: -

https://www.mediafire.com/file/aibgz13648y32o4/short_explanation.mp4/file

Note: Sorry for the bad video and audio quality because I have recently done my eye laser surgery, so I have to wear sunglasses and have to record video in dark room.