Group Name

GroupWon

Team Members

Destoni Baldwin
Michael Curtis
Jordan Leibman
Zachary Lowery
Troy McMillan

Won Word Processor

Project Overview

- We are working on a simple word processor to save us from the dreaded Libre Office.
- We are focusing on minimalistic functionality that works well rather than a broad scope of function that performs sub-par at best.
- We are creating this program using C#, XAML, and a MVVM programming design pattern.
  - ➢ We believe that using an MVVM pattern is overkill for this project, but it is more desirable than traditional methods.
  - ➢ The majority of our team mates use Visual Studio 2015/2017 for an IDE, and also use Windows 7 – 10 as a design platform.

Key Architectural Drivers

There were three main requirements that drove our design decisions:

- It had to be fast.
- It had to be responsive.
- It had to be up to date.
  - As an honorable mention, it had to be simple.

- It had to be fast, because we needed freedom from Libre’s slow execution speed.
- It had to be responsive, because we can’t abide an editor that makes you guess at what it is doing while you’re doing it.
- And it needed to be up to date, because processing stale information is pointless.

Architectural Style Choices

There were two potential candidates for our architecture:

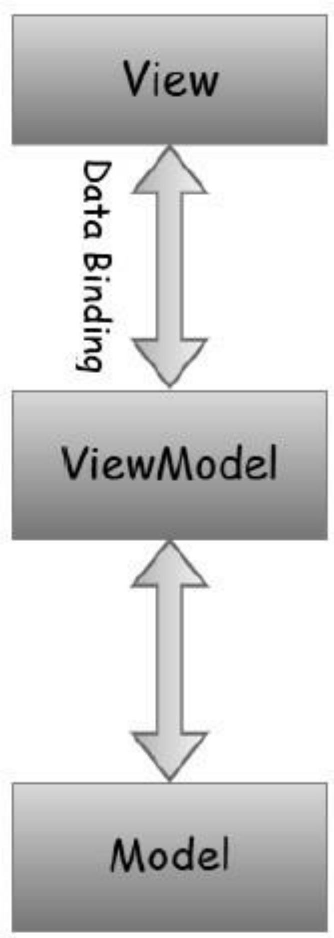
- Publish-Subscribe.
- Layered Systems.

- The Publish-Subscribe architecture was a candidate because it allowed for up to date information in a responsive manner due to its event-driven nature.
- The Layered Systems approach was considered because it naturally adhered to the typical structure of a word processor as a user-driven piece of software.
  - ➢ We eventually chose the Publish-Subscribe architecture.

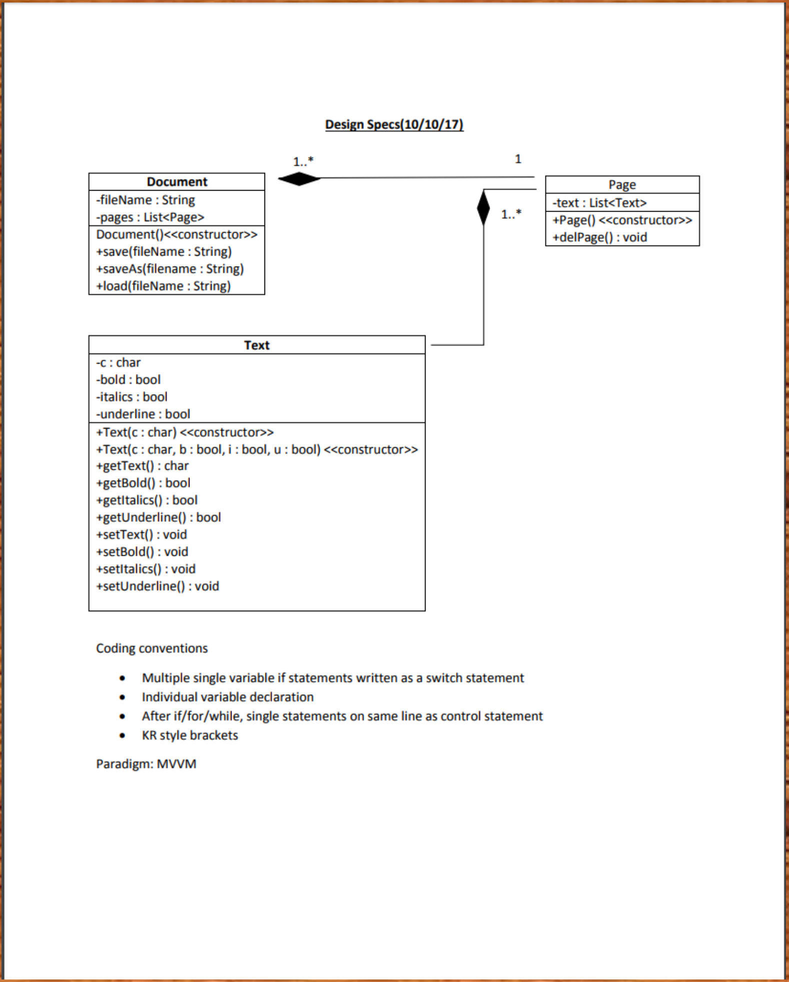
Our Architecture

Typical MVVM Architecture

(Image Courtesy of www.tutorialspoint.com)



Our Back end (Model)



Design Specs(10/10/17)

Document
-fileName : String
-pages : List<Page>
Document()<<constructor>>
+save(fileName : String)
+saveAs(filename : String)
+load(fileName : String)

Page
-text : List<Text>
+Page() <<constructor>>
+delPage() : void

Text
-c : char
-bold : bool
-italics : bool
-underline : bool
+Text(c : char) <<constructor>>
+Text(c : char, b : bool, i : bool, u : bool) <<constructor>>
+getText() : char
+getBold() : bool
+getItalics() : bool
+getUnderline() : bool
+setText() : void
+setBold() : void
+setItalics() : void
+setUnderline() : void

Coding conventions

- Multiple single variable if statements written as a switch statement
- Individual variable declaration
- After if/for/while, single statements on same line as control statement
- KR style brackets

Paradigm: MVVM

Conclusion

We are using a Publish-Subscribe model to capture the necessary aspects of a word processor.

Risks include:

- Complexity of development (Publish-Subscribe is hard to test because it is primarily event driven)
- Lack of speed, the publish-subscribe architecture doesn't cater to speed, so we will have to make up for it through minimal, functional design.

Interesting Notes:

During the implementation phase, our lead programmer discovered that C# has certain features and classes that greatly ease development, so our design is undergoing a revamp.

- ➢ We are discussing using a separate design pattern and are revamping the architecture/implementation as a result.