EFFICIENT GUI DEVELOPMENT USING WINDOWS PRESENTATION FOUNDATION FRAMEWORK AND MODEL-VIEW-VIEWMODEL PATTERN

[1]Mrs. Sangita Oswal, [2]Mr. Siddhesh D. Kushte

1Professor, Department of MCA, V.E.S. Institute of Technology, Mumbai, India, sangita.oswal@ves.ac.in
2 Student of MCA, V.E.S. Institute of Technology, Mumbai, India, siddhkushte@gmail.com

Abstract

Abstract- The paper focuses on flexible GUI development for trade events viewer. Here we focus on tool for trading system developed using WPF (Windows Presentation Foundation) and MVVM (Model-View-ViewModel) pattern. MVVM architecture is an indirect successor of MVC pattern and it successfully overcome on flaws of latter technique by removing dependency between model and controller by synchronizing View with ViewModel. In Financial Industry, various applications require flexible GUI because market value of financial products continuously changes. This paper explains use of WPF and MVVM pattern in UI development of financial application.

Index Terms- flexible GUI development, MVVM, WPF, trade

1. INTRODUCTION

Financial industry comprises complex networks of organizations, which primarily deal with management of money and create conditions for investors and corporations to flourish in the market. The growth of other sectors is closely dependent on this industry as it is a prime source of liquidity and thereby ensures the overall prosperity and economic stability. This multi-trillion dollar services industry comprises companies varying a great deal in size and their offering as well. The industry grouping is widely branched out to include companies providing varied services, preventing a simple categorization of this industry [1].

Financial industry derives new financial products for the management of money. Financial industry has a need for various applications due to various financial products. These applications require flexible GUI because market value of financial products continuously changes. In this paper I am going to talk about such needs for a software change by utilizing Windows Presentation Foundation (WPF) and Model-View-ViewModel (MVVM) design pattern to provide our customers with GUI of their choice while ensuring there is a minimum impact to the rest of the code base. This paper is organized as follows: first domain knowledge about the industry. Next several approaches how GUI is usually made are presented in Background, followed by a brief description about what WPF and MVVM are.

2. DOMAIN KNOWLEDGE OF PROJECT

Financial industry comprises of following segments:
1. Consumer Finance segment
2. Capital Markets segment
3. Diversified Financial Services segment

2.1 CAPITAL MARKETS

A capital market is one of the segments in financial industry. Establishments in this segment undertake activities, including trading, brokerage, strategic advisory, portfolio management, asset management and investment advice. They primarily work as intermediaries, either to provide or manage capital, thereby satisfying financial goals of institutions and individuals.
Companies operating under the segment can broadly be classified into three distinct categories:

- The first set includes investment banking and brokerage companies that provide services, such as underwriting of bond and stocks to raise capital, trading and broking of stocks, bonds, derivatives and commodities, as well as companies engaged in strategic advisory services.
- The second set comprises asset management firms, including companies that professionally manage large pools of money from individuals and institutions with an aim to satisfy a common investment goal.
- The third set includes companies classified as diversified capital markets, which provide more than two services and drive a majority of their revenues collectively from both of them with no significant proportion coming from only one service.

2.2 TRADE

Trade is a basic economic concept that involves multiple parties participating in the voluntary negotiation and then the exchange of one's goods and services for desired goods and services that someone else possesses. In financial markets, trading also can mean performing a transaction that involves the selling and purchasing of a security.

3. BACKGROUND

This section will explain some design patterns that are used to create GUI applications. Those patterns are still valid and widely used in application development, but WPF and MVVM design pattern provide a step forward which is going to described after this section and is the purpose of this paper.

3.1 MODEL-VIEW-CONTROLLER (MVC) PATTERN

Model View Controller (MVC) model began as a framework developed by Trygve Reenskaug [2] for the Smalltalk platform in the late 1970s. The working of MVC pattern was first described in [3]. In MVC pattern, a Model is used to represent data or business logic that the rest of the application requires. Model knows how to store, load, handle and transform data based on the inputs it gets. Model notifies Views through a Controller when any change of the data occurs so Views can update themselves accordingly. View is a component in the pattern that is used to display data to the actual user and had elements that allow user interaction such as buttons, textboxes, trees, etc.

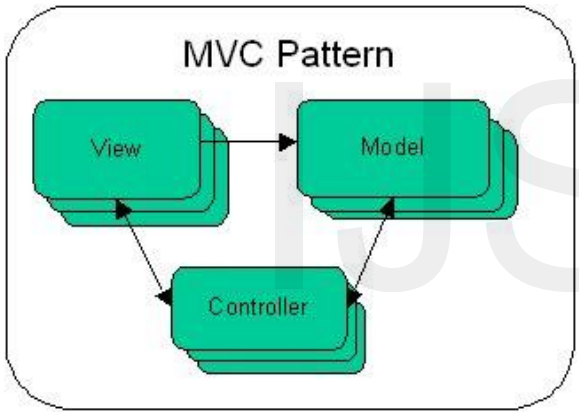

Fig. 1: MVC pattern

3.2 MODEL-VIEW-PRESENTER (MVP) PATTERN

Model View Presenter (MVP) developed in 1990s in Taligent Software Company. The working of MVP pattern was described in [4], which is similar to MVC pattern called MVP Passive View. In MVP pattern, Presenter receives events from the GUI, triggers and update to the Model. Model notifies the View about its change which is accomplished via data binding for .NET development [5]. View can be a WinForms application, which is a regular desktop application, or a web page, where Presenter has logic and data. View needs to display data properly. Result is that developers can greatly reuse their code and develop the application easily for 2 display mediums, and it also provides for automated testing of model and logic behind the View.
View responsibility is to show the data provided by presenter, and purpose of the presenter is to reach to model, retrieve the needed data, performs required processing and returns the UI prepared data to the view.

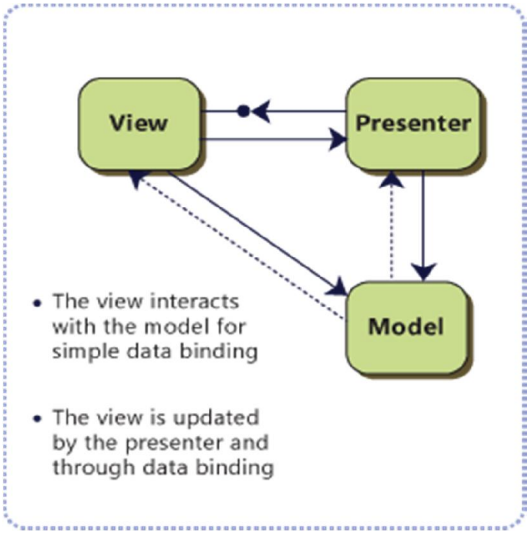

Fig. 2: MVP pattern

4. WPF AND MVVM PATTERN

4.1 WPF

Windows Presentation Foundation (WPF) is a new graphical subsystem for rendering and displaying of Microsoft Windows applications [6]. It builds upon DirectX for drawing and rendering content which gives developers and designers lots of tools to create graphically pleasing user interfaces. WPF introduces a common programming model and clearly separates presentation from logic. WPF provides graphics services (2D, 3D and vector graphics) which are rendered on the graphics card GPU leaving minimal impact to the CPU. WPF provide powerful data binding mechanism, media services, layout engine so that the application can be easily resized and/or displayed on various screen sizes. It does not use fixed point GUI widget placing as was the case before WPF, templates that are used to redefine how a GUI element looks (control template) or how data should be displayed (data template), animations, better text services, and so on. Thus, WPF is an extensive library full of features fir creating very expressive GUI's and is used in .NET development environment. Creation of GUI in WPF is done using XAML [7], which is XML-like declarative language, where GUI is created using XAML declarations and code logic behind GUI elements is created using one of the .NET languages such as C# and VB. Sample XAML snippet given below

```
<Button Width="100" Height="100">
    <TextBox Width="75">edit me</TextBox>
</Button> [8]
```

WPF however does not force a developer to use XAML and all GUI can be created from code if one chooses to do so. However with XAML application designers can contribute to the software development lifecycle where their GUI design can be seamlessly and immediately used in the actual application, developers do not need to develop a GUI per design as it was customary prior to WPF. Thus, software can be developed in parallel – designers develop GUI while

developers create code behind such as business logic and data management modules.

4.2 MODEL-VIEW-VIEWMODEL (MVVM) PATTERN

Model-View-ViewModel (MVVM) [9] pattern was developed at Microsoft by John Gossman as a variation of the MVP pattern to leverage benefits and features of WPF for application development. Model contains the data and does not know about the View or the ViewModel. ViewModel is an abstraction of the View, and contains all of its data and state. ViewModel does not have a reference to the View class, but a data binding is used where ViewModel is a data context of the View and View is bound to properties ViewModel has. If some ViewModel property changes, View receive a new value and if View issues some command, ViewModel executes the command. Binding as described is accomplished using dependency property in WPF and is deeply embedded inside WPF library and XAML.

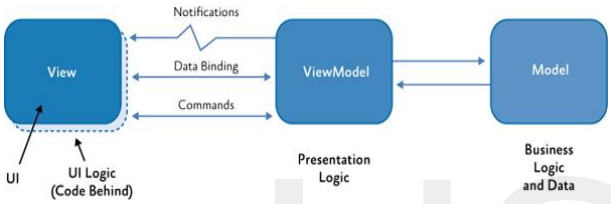

Fig. 3: MVVM pattern

5. FLEXIBLE GUI IN TRADING APPLICATION

Trade events viewer generates tree structure based on various operations performed on block trade. Various operations performed on trade are:

1. Split
A corporate action in which a company divides its existing trade into sub accounts. Although the number of shares remains the same compared to pre-split amounts, because the split did not add any real value.

2. Assignment
An assignment of trade is a term that is used to describe a situation in which one of the parties involved in a trade decides to assign that trade to a party that was not part of the original deal.

3. Novation
A novation is a transaction in which a 'transferor' transfers to a 'transferee', with the consent of the 'remaining party', all of its rights and obligations under the contract in respect to the novated amount. The effect of the agreement is that for the 'novated amount' (i.e. all or part of the outstanding notional amount), the old transaction between the transferor and the remaining party is terminated, and a new transaction is executed between the remaining party and the transferee with economic terms identical to those of the old transaction.
4. Unwind
To close out a position that has offsetting investments or the correction of an error. Unwinds occur when, for example, a broker mistakenly sells part of a position when an investor wanted to add to it. The broker would have to unwind the transaction by selling the erroneously purchased stock and buying the proper stock.
Trader can perform any above operation at any point of time. Trade Event Viewer shows sequence of operation performed in tree - like structure.

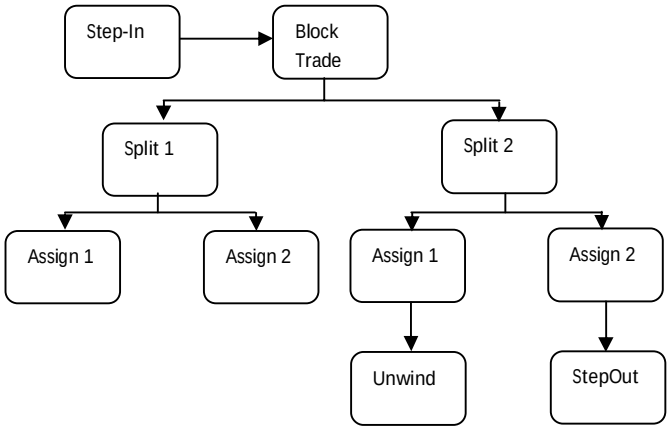

Fig. 4: General tree scenario

The sequence of trade operation in not pre-defined. Therefore, structure of tree varies for every trade.

5.1 NEED FOR FLEXIBLE GUI

During trade life cycle, Trader can perform any operation on trade. Hence, the structure of trade event tree changes for different trade and after trade operation performed on it. Thus, it is essential that application is flexible enough to accommodate those changes while not affecting other software code parts to prevent introducing the bugs as a side-effect of the changes.

5.2 WPF AND MVVM SOLUTION FOR FLEXIBLE GUI

MVVM pattern greatly decouple GUI from the rest of the code to a far greater extent than it was possible using MVC/MVP pattern for GUI development. In Trade Events Viewer, I design all GUI components in XAML and declarative nature of XAML speeds up GUI design process. Using WPF and XAML to define a GUI we can experiment easily and change/redefine the GUI quicker. Using XAML, we can define complex GUI components. Following code shows complex structure TreeView component

```
<TreeView ItemsSource="{Binding TreeRoot.ChildNodes}">
<HierarchicalDataTemplate DataType="{x:Type Model:Node}"
ItemsSource="{Binding ChildNodes}">
<StackPanel>
<TextBlock Text="{Binding TicketId}" DataContext="{Binding}"
HorizontalAlignment="Center"/>
<ListView ItemsSource="{Binding EventTypes}"/>
</StackPanel>
</HierarchicalDataTemplate>
</TreeView>
```

E.g. 1: Sample code for customized TreeView

In the above code, I added TextBlock and ListView component in TreeView. To design such a GUI is difficult in other languages. Using XAML, we can create such a complex code easily.

MVVM pattern greatly improved productivity and drastically reduced side-effect errors when a modification had to be done. With MVVM pattern we were able to produce self contained modules consisting of Model and ViewModel parts for a certain object in our system. ViewModel exposes properties and command objects a GUI (a View in the MVVM pattern) can display or act upon. GUI changes or modification do not require any changes of the ViewModel that has logic.

View can have code behind, normally written in C# or VB. WPF reduces code behind coding by using features like data binding, command binding etc. One of the key principles for designing controls in WPF is separation of UI from the actual control implementation. We write control implementation as property in ViewModel, which helps to reduce code behind coding. In above code, Text property of TextBlock component bind with TicketId property of ViewModel. This will reduce code behind code. WPF provides a simple and powerful way to auto-update data between the business model and the user interface. This mechanism is called Data Binding. Everytime when the data of your business model changes, it automatically reflects the updates to the user interface and vice versa. The source of a data binding can be a normal .NET property or a DependencyProperty. The target property of the binding must be a DependencyProperty.

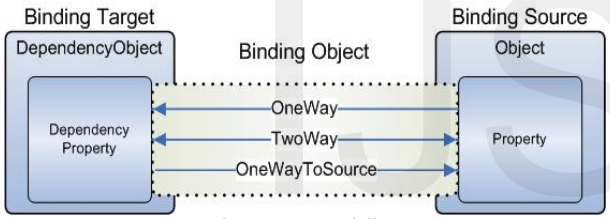

Fig. 5: Data Biding

5.3 SAMPLE APPLICATION

This section briefly describes Trade Events Viewer application using MVVM pattern. UI of application would look like in Fig. 6. A main window shows tree structure which describes various events performed on trade in hierarchical manner.

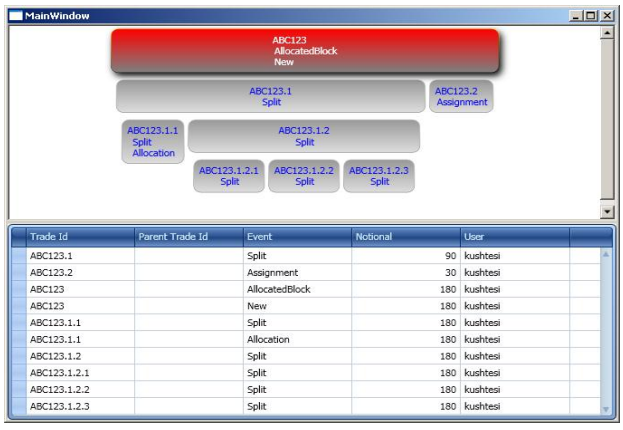

Fig. 6: Application Screenshot

Following MVVM pattern, Main window has its corresponding ViewModel that has a property list of TreeViewModel.

CONCLUSION

In this paper an application of new framework of WPF and MVVM has been presented that is more suitable to the needs and requirements of Trading applications that have to be reliable but also flexible to accommodate customer changes, specific requirements and future changes. Traditional MVC/MVP patterns provide a degree of separation between logic and GUI code, but WPF and MVVM take this a step further. WPF and MVVM separate design part of application from business logic (or development). So designer can work on view and developer on business logic using ViewModel independently. Utilizing these technologies proven to be greatly beneficial in our organization in terms of increased component reusability, fast response time to customer change requests, reduced side-effects of such changes helping in maintenance of software and shared code base library and parallel development of GUI and code.

REFERENCES

[1] John W. Molka III, “Financial Services Industry – Subprime and Credit crisis weighs heavy 2008 Edition”
Available:https://www.advisen.com/downloads/advisenIndustryReportDivFinConsFinCapMrkt.pdf

[2] G Trygve M. H. Reenskaug, “MVC XEROX PARC 1978-79”
Available:http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

[3] Steve Burbeck, “Applications Programming in Smalltalk-80™: How to use Model-View-Controller (MVC)”, March 4, 1997.
Available:http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

[4] Mike Potel, “MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java”. Taligent Inc, 1996.
Available: http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

[5] Microsoft Corp., “Model View Presenter Pattern”.
Available:http://msdn.microsoft.com/en-us/library/cc304760.aspx

[6] Microsoft Corp., “Introducing Windows Presentation Foundation”.
Available: http://msdn.microsoft.com/en-us/library/aa663364.aspx

[7] Microsoft Corp., “XAML Overview”.
Available: http://msdn.microsoft.com/en-us/library/ms752059.aspx

[8] Chris Sells & Ian Griffiths, “Programming WPF, second edition”. O’Reilly publications, 2007, pp 17

[9] Arlen Feldman, Maxx Daymon, “WPF in Action with Visual Studio 2008”. Manning Publications Co, 2009, pp 283-285