ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

Industrial Automation using Windows Presentation Foundation & Model View View-Model Pattern

Sameer Soni[1], Pranali Dhete[2], Shirish Patil[3] and Dr B.B. Meshram[4]
Department of Computer Technology
Veermata Jijabai Technological Institute, Mumbai, India

Abstract — **The paper focuses on flexible GUI development for industrial tool automation. Here we focus on tool for CNC machine automation developed using WPF (Windows Presentation Foundation) & MVVM (Model View View-Model) Pattern. The scheme allows flexible development of tool, better management of business intelligence and parallel code development without overlapping of concepts. The paper also considers the custom software tool development by multiple operators that share a commonality. MVVM architecture is an indirect successor of MVC pattern and it has successfully churned out the flaws of latter technique by removing dependency between model and controller by synchronizing View with ViewModel[1]. The main advantage now available is that parallel development & execution can be supported because ViewModel always has the status of View available with it. Industrial tool development often requires time and multiple developers for timely completion of project, and the approach suggested here is able to achieve the goal desired.**

Index Terms—**Industrial Automation, CNC Tool, WPF, MVVM, Parallel development, Flexible GUI development**

I. INTRODUCTION

Gear Metrology Industry is the essence of all tool based industries, and deals with accuracy and precision of manufactured tools like gears, hobs, shavers, etc. These tools being the basic component of machinery, needs to be accurate in dimensions at micron level. For example, gears used for power transmission, has several angular teeth and have a complicated design. For the machine to work properly, their accuracy is prime necessity.

Now if the tool manufactured is quite accurate, then also it needs to avoid human errors, and so here we propose a system for CNC machines. The software tool that we have developed will not only automate the manual machine, but also raise the precision level to microns.

The software tool will provide Graphical User Interface for user to feed in values as per gear tool to be analyzed, and the user will then select the option for checking the tool. The application would then instruct the physical testing machine to firstly identify the axis alignment and then test the tool for lead, profile, pitch and other parameters.

After testing of tool, a report would be generated which will depict the variations between the actual tool and the expected values. So far this has been the overall functionalities that are to be performed using the software application tool.

For making the desired application tool, a robust development architecture and methodology is required that is able to allow changes and modification at later stage of development. These modifications include hardware and software modifications that often involve GUI changes and additions on top of the basic design. In this paper I am going to talk about how Prime Technologies copes with such needs for a software change by utilizing Windows Presentation Foundation (WPF) and Model View ViewModel (MVVM) design pattern to provide its customers with GUI of their choice while ensuring there is a minimum impact to the rest of the code base, and thus supporting Industrial Automation. This paper is organized as follows: first several approaches how GUI is usually made are presented in Background, followed by a brief description about what WPF and MVVM are. Next a concrete example is provided that shows how WPF and MVVM can be used beneficially in Industrial Automation, followed by a sample application and conclusion of the paper.

II. BACKGROUND

This section will present some techniques that are used to create applications having a GUI. Those techniques are still valid and extensively used in contemporary application development but WPF and MVVM provide a step forward which is going to be described after this section and is the purpose of this paper.

A. Model View Controller (MVC) pattern

MVC Pattern was created in late 1970's[2]. In MVC pattern, a Model is used to represent data that the rest of the application needs. Physical storage is not a concern of the MVC pattern, but it is assumed that 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, tabs, etc.

Model encapsulates the core data and functionality. View encapsulates the presentation of the data there can be many views of the common data. Controller accepts input from the user and makes request from the model for the data to produce a new view.

B. Model View Presenter (MVP) pattern

Model View Presenter (MVP) developed in 1990s and pattern is similar to MVC pattern called MVP Passive View. In Supervising Controller MVP variant, Presenter receives events from the GUI, triggers and update to the Model and Model notifies the View about its change which is accomplished via data binding. MVP is used among other things as the basis design for .NET development where View can be a Window/User-Control application, which is a regular desktop application, or a Web page, where Presenter has logic and data View needs to display itself 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.

III. WPF AND MVVM PATTERN

A. WPF

Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications[3]. 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, vector graphics) which are rendered on the graphics card GPU leaving minimal impact to the CPU, provide powerful data binding mechanism, media services, layout engine so that the application can be easily resized and/or displayed on various screen sizes and 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 for creating very expressive GUI's and is used in .NET development environment. Creation of GUI in WPF is done using XAML [5], 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#. Sample XAML snippet can be seen in Fig 1[3] below. 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.

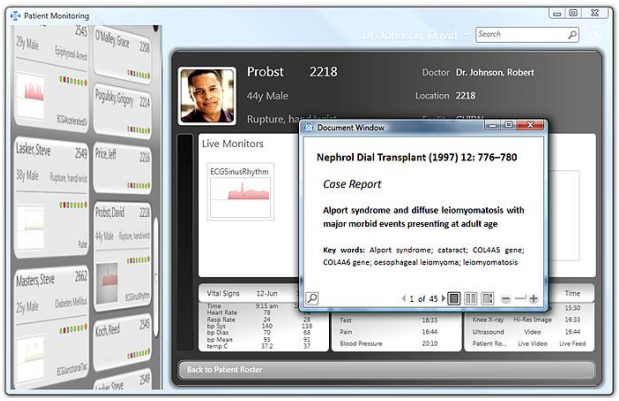

Fig 1. Sample WPF Application[3]

B. MVVM

Model View View Model (MVVM) [4] 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 mechanism 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 receives a new value and if View issues some command, ViewModel executes the command. For example, ViewModel for a motion axis can have properties as follows: axis name, position, speed, error condition. View can have GUI elements such as labels or text boxes that are bound to those properties. If a position is changed while motor is driving the axis, position property value will change and View will always have correct value to display.

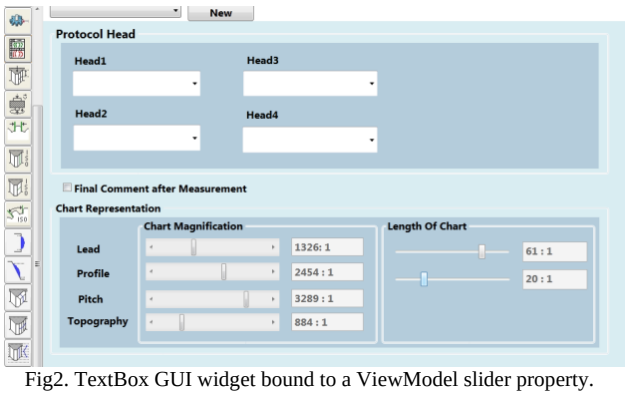

Fig2. TextBox GUI widget bound to a ViewModel slider property.

Binding as described is accomplished using dependency property in WPF and is deeply embedded inside WPF library and XAML. For example XAML in Fig 2 above defines such a text box, where values change on moving sliders.

IV. INDUSTRIAL AUTOMATION USING WPF & MVVM

Gear Testing Industries check and verify various industrial tools like gears and hobs, for multiple parameters like lead, profile, pitch, etc and prepare their report for their manufacturing accuracy and quality. These reports are then used by the production department for improving their manufacturing process.
For gear testing, these industries have been using manual machines which involve measuring probes mounted on horizontal and vertical axis, and their position and movement is regulated by hand held drives. The process is though delivers sufficient results, but requires quite amount of hand skilled proficiency, and involves huge amount of risk of human error.
To overcome this flaw of human error, idea of automated machines i.e. CNC machines have been proposed, and are they are widely used these days. Although it involves a huge investment at initial stage, but proves its real worth in longer run. It removes human error, is time efficient and involves least manual operations and thus supports Industry to automate.
To support this automation in the CNC machine, the background software and the embedded systems play a vital role which provide these machines the logic and controls to behave as per commands.

A. Need for Flexible Design

Each manipulator produced on a custom basis for a particular customer shares a lot of commonality in terms of hardware, electronics and software but is also customized for a particular customer according to their specifications and expected needs. [6]Often times, there are requests to modify the GUI to suit specific customer need, add additional control component for a specific custom application or make several GUIs to be used with the same equipment depending on the actual operator role. Thus, it is essential that software 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. Lastly, in today's market of this specific industry, all parts of the system including software need to be fully localized to the native language and it is essential that technology used to develop such software support this requirement seamlessly.

B. WPF and MVVM Solution for Flexible GUI

WPF was designed to 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. Here, we 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. Our designers can develop GUI design and work with the customer and application departments to come up with a intuitive design while development team focuses on logic and other modules of the software application. Lastly, WPF support for localization in terms of auto sizing of GUI widgets and layouts helps us in localizing our software, which was not possible using .NET WinForms where a widget had to be resized manually if localized text did not fit.

Using MVVM pattern greatly improved our 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, fully test it using testing harnesses. ViewModel exposes properties and command objects a GUI (a View in the MVVM pattern) can display or act upon. GUI changes or modifications do not require any changes of the ViewModel that has logic,etc and View does not have any code behind – it is pure XAML having only GUI look design declaration and data binding specification.
Thus, to customize a GUI for a specific customer we can only change the XAML portion of the application to expose or hide a certain property or command binding in case more or less data is needed. To change physical layout and design XAML is also easily changed. To do a more involved change, we can create another ViewModel and corresponding XAML and other parts of the system do not have to be modified because XAML can use DataTemplate to select dynamically a View for a certain ViewModel object instance. Thus, main window XAML is minimally changed to register the new ViewModel and rest of the application code remains unchanged. It should be noted that a bigger change may require changing of the other ViewModels and performing their testing again, but at Inetec using WPF and MVVM pattern we have reduced our time and effort to do changes to the GUI by about 80% comparing with when WinForms .NET application was developed and used. Effort in doing the localization and making sure all GUI elements display correctly was reduced drastically to a similar percentage.

C. Parallel Development

The Tool to be developed has several modules and each module has various components in them. For a faster progress of events, these modules have been divided among various team members, and then each of them worl on their own individual component[7]. A common repository has been maintained at server where every developer commits his part of work, and thus the other developer can now access this updated work in his working copy simply by updating the component.
Thus, the cycle progresses and developments proceed in distributed yet common fashion.

D. Sample Application

This section briefly describes a 4-axis CNC Gear Testing Machine using MVVM pattern. Finished application would look like in Fig 3. The Main Window will allow user to select various components like Gears, Hobs, Stylus, Settings, Macros, Reports, and then as per functionalities chosen, he can either feed in the actual values, or tests the desired tool against the selected standard. Following MVVM pattern, Main window has

its corresponding ViewModel that has a property list of multiple Screen ViewModels object instances and one ViewModel for system instance. For each ViewModel, there is a custom WPF user control created which corresponds to a View and is bound to properties of its particular ViewModel. In XAML on the application level, in a data template each ViewModel is registered to use a particular custom WPF user control. Thus, in MainWindow when <ItemsControl> GUI Widget is bound to property which is a list of ViewModels, it will populate its items list with instances of Views defined in the data template and would set the data context of View to a particular item in the list it was bound to – showing (items) as in Fig 3.

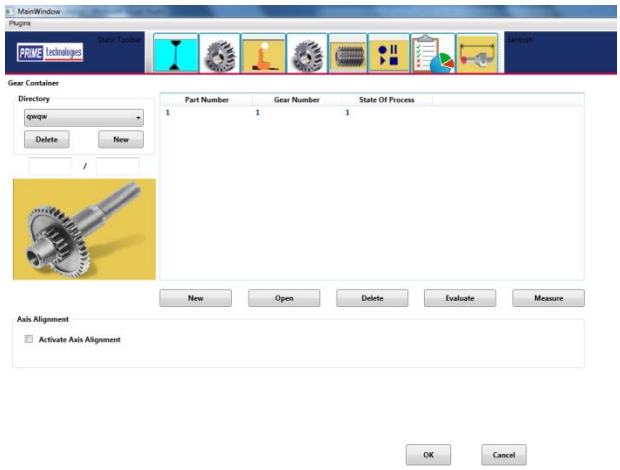

Fig 3. Sample application screenshot showing GUI structure.

As ViewModel data changes, for example input values changes, View will display correct position in the textbox automatically due to binding. Thus, once ViewModel is created along with corresponding user control implementing the View, things work automatically in WPF and MVVM framework.

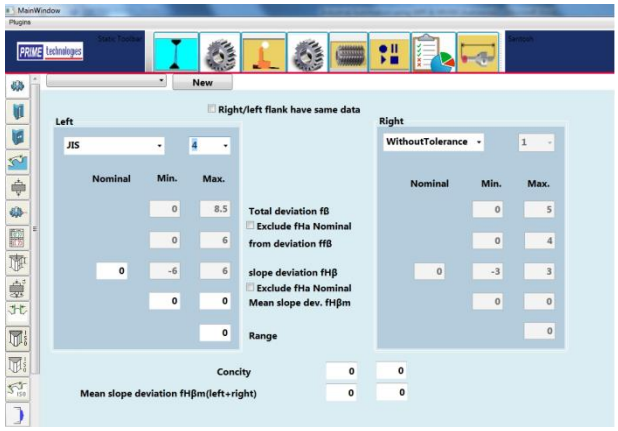

Fig 4. An intermediate Screen which allows selection of different standards for left and right flank

Also, it can be seen that different checking parameters can be applied on left and right flanks of tool to be tested. There are also several graphical changes which allow movements of ellipses, changes in contours, and other graphical changes without affecting the background logic code base.

Thus, WPF and MVVM can provide for a rapid prototype development to be used quickly with a simple and then GUI can be further refined in function, visual cues and graphic design with frozen and tested ViewModel code. This has proven to be greatly beneficial for custom software design, along with component reusability MVVM provides.

V. 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 Tool Testing Industry that have to be reliable but also flexible to accommodate customer changes, specific requirements and future changes which company develops. Traditional MVC/MVP patterns provide a degree of separation between logic and GUI code, but WPF and MVVM take this a step further. Utilizing these two 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, parallel development of GUI and code and reduced time to spend on localization.

REFERENCES

[1]. MVVM vs MVP vs MVC
http://nirajrules.wordpress.com/2009/07/18/mvc-vs-mvp-vs-mvvm/

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

[3]. Introduction to WPF .NET Framework 4
http://msdn.microsoft.com/en-us/library/aa970268.aspx

[4]. XAML in WPF
http://msdn.microsoft.com/en-us/library/ms747122.aspx

[5]. Introduction to Model/View/ViewModel pattern for building WPF apps – John Gossman
http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx

[6]. Flexible GUI in Robotics Applications Using Windows Presentation Foundation Framework and Model View ViewModel Pattern - Fran Jarnjak, IEEE 2010

[7]. Ergonomics Research and CNC machine tools in the interface design of the application - IEEE 2008

[8]. Microsoft Corporation. “Model View Presentter Pattern” (undated).
http://msdn.microsoft.com/en-us/library/cc304760.aspx

[9]. Microsoft Corporation, “Introducing Windows Presentation Foundation” (undated).
http://msdn.microsoft.com/enus/library/aa663364.aspx