IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol.-6, Issue-11, Nov 2018 E-ISSN: 2347-2693

Model Transformation of Platform Specific Model to Vanilla Model – A Proposed Platform Independent Model for Declarative User Interface

Smita Agarwal[1], Alok Aggarwal [2*], S. Dixit[3,] , Adarsh Kumart[4]

[1]Department of Computer Science, Mewar University, Chittorgarh (Raj), India
[2]School of Computer Science, University of Petroleum & Energy Studies, Dehradun, India
[3]Department of Computer Science, Mewar University, Chittorgarh (Raj), India
[4]School of Computer Science, University of Petroleum & Energy Studies, Dehradun, India

[*]Corresponding Author: alok289@yahoo.com, Tel.: 7906230838

Available online at: www.ijcseonline.org

Accepted: 18/Nov/2018, Published: 30/Nov/2018

***Abstract*—** In classic software re-engineering, the user interface are considered to be Platform Specific. Hence were always excluded from the process of software re-engineering. The user interface were re-written for the target platform and integrated with business application in the end. In this paper we propose a Vanilla Model – A Platform Independent Model for Declarative User Interface and algorithm for model transformation using Vanilla Model for Declarative User Interface. This approach will preserve the source artifact user interface will make re-engineering user interface part of main process,. This transformation is then applied to five of the popular libraries such as SWING, HTML5, and more recent libraries of Android and Python- Tkinter.

***Keywords*—** Model Transformation, Declarative User Interface , Platform Specific Model, Platform Independent Model

I. INTRODUCTION

The OMG (Object Management Group) has defined the Model Driven Architecture(MDA) as part of its response to the increasing complexity, heterogeneity and evolutionary issues of information systems[1]. It solved these issues through the rising the level of abstraction by adopting models instead of objects as a first measure and the separation of the business logic of an information system from the implementation of that logic on a specific technological platform as a second one. Thus, the simple principle of MDA is the elaboration of platform independent models (known as PIMs) and their transformation into platform specific models for a given platform (known as PSMs). The techniques used are essentially modeling techniques and model transformation techniques

In classical software re-engineering, the reusability of user interfaces across development platforms is not possible. In addition, in software re-engineering based on MDA, they are integrated only after making the transformation of the PIM to the PSM since they belong to the target platform and hence have the same problem. They are considered part of the PSM, which deprives us from reusing them as we do for the business logic.

In this research paper, in Section 2 , we discuss a thorough literature study on Model Transformation, in Section 3, we propose our Vanilla Model- Platform Independent Model to build Declarative User Interface , its Element Library, Model Hierarchy, algorithm for Model Transformation and criteria for successful Model Transformation. In Section 4, we map the Vanilla Model Element Library with the various popular User Interface Library like, Swings, HTML5, Android and Python. In the last section we conclude and discuss the future scope.

II. STATE OF ART

Software Re-engineering that uses model based representation of the existing systems giving a comprehensive understanding is termed as Model Driven Re- Engineering[2].

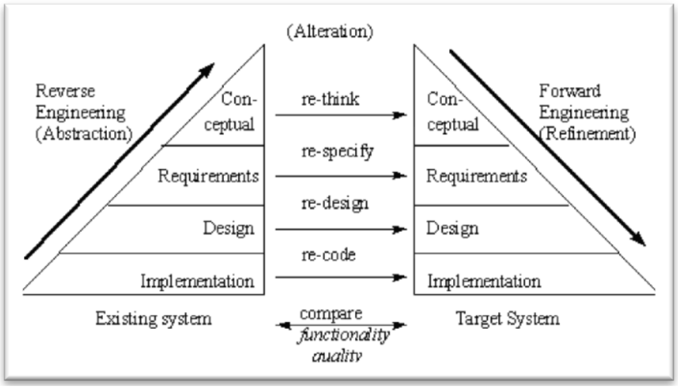


Figure 1: Software Re-engineering

They syntax and semantics of the modeling language is expressed by the Meta-model. For Example – UML Meta-model is expressed using class diagrams. And semantics is described by well- formed rules and natural language.

Kleppe et al [3] defined model transformation. as an automatic generation of a target model from a source model. A transformation definition is a set of transformation rules that together describe how model in the source language can be transformed into a model in the target language. A transformation rule is a description of how one or more constructs in the source language can be transformed into one or more constructs in the target language.

Declarative UI is a UI that's designed in a declarative way i.e. one describes what UI should be like rather than an imperative way i.e. one codes the steps to create it. For example, in HTML one can describe that one wants an input field, but how and where this field will be placed at the UI is highly dependent on the browser. Declarative approach [4] is appealing as navigation through source model and management through traceability matrix are inherent Declarative transformations tend to be simpler to write and comprehend

III. PROPOSED MODEL

1. Vanilla Model – A Proposed Platform Independent Declarative User Interface Model

Different User Interface Platforms differ in design and richness, the underlying functionality of the basic input and output element remain the same. Hence the elements like labels , textboxes , checkboxes , buttons , radio buttons etc exists with different formats and names in different user interface frameworks but their concept remain the same. For Example a checkbox which is used to fetch user choice from multiple options is a JCheckBox in Java SWING (javax.swing.JCheckBox), checkbox in HTML, Checkbutton in Python Tkinter and CheckBox in Android View Widget. The elements having similar functionality although they are having different class name in different market libraries.

We propose Vanilla Model- A Platform independent Meta-Model for Declarative User Interface. The 25 common elements of User Interface are identified with their characteristic structure. Each element then mapped to elements / widgets of 4 open source market libraries of Graphical User Interface – Java Swing Library, HTML Widget Library, Android View Library and Python Tkinter Library. The transformation of source model(PSM) from one platform specific meta-model to Vanilla Model (PIM) is part of reverse engineering process. And the transformation from Vanilla Model (PIM) to target model(PSM) is part forward engineering. It is represented schematically in Figure2.

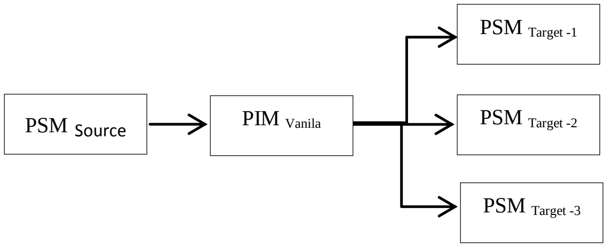
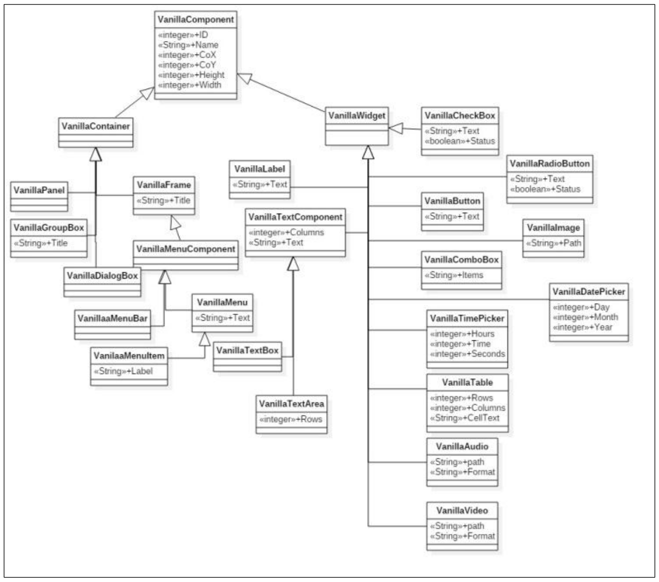


Figure 2 Proposed Transformations using Vanilla Model

2. The Vanilla Model Element Library



Figue 3. Vanilla Model Element Hierarchy

Table 1. Vanilla Model Element Library

S. No	Element Name	Parent Class	Attributes	Description
1	VanillaComponent	None	- id:int - name: String - cox:int - coy:int - height:int - width:int	Parent class of all the components of Vanillas Model
2	VanillaContainer	VanillaComponent	None	A special type of component that has capability to add component to itself.
3	Vanilla Widget	VanillaComponent	None	A component that helps user to interact with UI.
4	VanillaFrame	VanillaContainer	None	It is a container which has a collection of related widgets grouped together
5	Vanilla MenuComponent	VanillaFrame	- title:String	Parent class for all the menu controls.
6	Vanilla MenuBar	VanillaMenu Component	None	It provides a Menu Bar that is bound to a Frame.

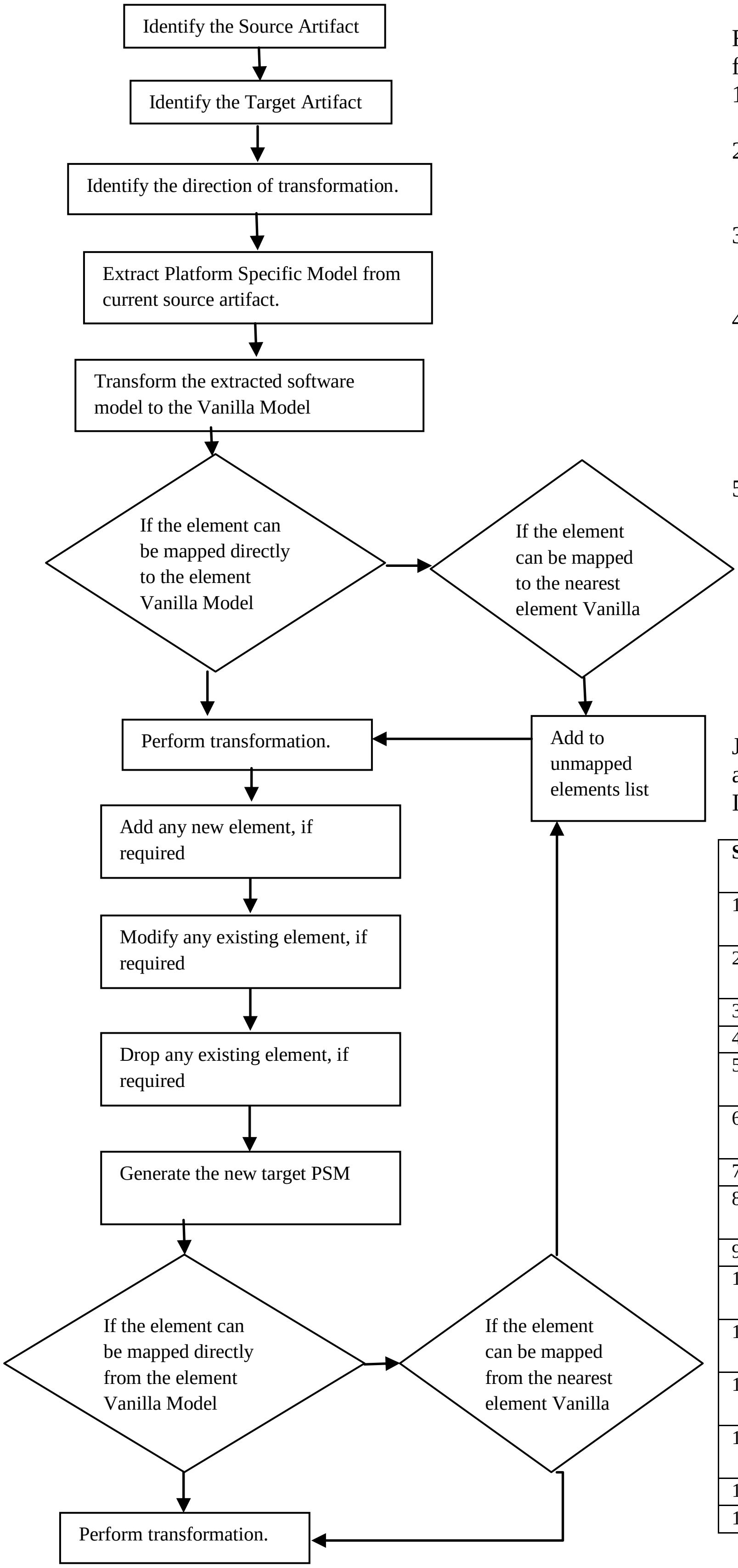
7	Vanilla Menu	VanillaMenu Component	- text:String	It is the pulled down menu component on the menu bar.
8	Vanilla MenuItem	VanillaMenu	- label:String	It is a simple labeled option in the menu
9	VanillaLabel	VanillaWidget	- text:String	It is a widget to put simple text on UI.
10	VanillaTextComponent	VanillaWidget	-column:int -text:String	Parent Class of any component that allows the editing of some text.
11	VanillaTextBox	VanillaText Component	- text:String	It is a text component that allows editing of single line of text.
12	VanillaCheckBox	VanillaWidget	- text:String -status:Boolean	It is a widget that has binary value true or false.
13	VanillaRadioButton	VanillaWidget	- text:String -status:Boolean	It is a widget that allows the user to select only one of a predefined mutually exclusive options.
14	VanillaButton	VanillaWidget	- text:String	It is labeled widget that generates an event when pressed.
15	VanillaTable	VanillaWidget	-rows:int -column:int -cell:String	It is two dimension widget of cells consisting of ows and columns.
16	VanillaImage	VanillaWidget	-path:String -format:String	It is a widget representing graphical image.
17	VanillaListBox	VanillaWidget	-items:String	It is a widget that provides a list of items from which the user can select.
18	VanillaComboBox	VanillaWidget	-items:String	It is a widget of drop down list which lets the user select from pre-defined options.
19	VanillaTextArea	VanillaText Component	-rows:int	It is widget for editing multi-line text.
20	VanillaDatePicker	VanillaWidget	-day:int -month:int -year:int	It is widget for choosing date in a simple manner.
21	VanillaTimePicker	VanillaWidget	-hours:int -min:nt -sec:int	It is widget for choosing time in a simple manner.
22	VanillaAudio	VanillaWidget	-path:String -format:String	It is widget for playing sound in UI.
23	VanillaVideo	VanillaWidget	-path:String -format:String	It is widget for playing video in UI.
24	VanillaDialogBox	VanillaContainer	-parentComponent : Component -message:String	It is window for taking input from user.

IV. PROPOSED MODEL TRANSFORMATION

The transformation will be a two Step process 1. Model transformation from source model to Vanilla Model will be reverse engineering and 2. Model transformation from Vanilla Model to target model.
for forward engineering, thus completing re-engineering of User Interface. The process for Model transformation is as follows:

1. Identify the source and target artifacts for Model transformation.
2. Identify the direction of transformation i.e. from concrete to abstract (in case of reverse Engineering) or from abstract to concrete (in case of forward engineering).
3. Extraction of Platform Specific Model from current source artifact.
4. Transform the extracted software model to the Vanilla Model platform independent meta-model.
5. During the transformation from PSM to PIM, for each element that exist in PSM Model.
   a. Check if the element can be mapped directly to the Vanilla Model, if yes then perform transformation.
   b. Identify the elements that cannot be mapped to vanilla model. Either identify the nearest elements which can replace that elements or add them to unmapped element list.
6. Add any new elements to mapped vanilla model to add any new features required
7. Modify the existing elements as per new specification, if required.
8. Drop the existing components which are no longer required in the target artifact.
9. Finalize the Vanilla Meta Model.
10. Generate the new target Platform Specific meta-model from Vanilla model.
11. During the transformation from PIM (Vanilla) to target Platform Specific Model
    a. Check if the element can be mapped directly from the Vanilla Model to Platform Specific Model , if yes then perform transformation.
    b. Identify the elements that cannot be mapped from Vanilla model. Either identify the nearest elements which can replace that elements or add them to unmapped element list.
12. Manually modify the model to add elements from un-mapped element list
13. Finalise the new Platform Specific Meta Model.
14. Repeat Steps Vii and X to for multiple Platform Specific target Model.
15. Generate code for each of the target Platform specific model

FLOWCHART


Identify the Source Artifact
Identify the Target Artifact
Identify the direction of transformation.
Extract Platform Specific Model from current source artifact.
Transform the extracted software model to the Vanilla Model
If the element can be mapped directly to the element Vanilla Model
If the element can be mapped to the nearest element Vanilla
Perform transformation.
Add to unmapped elements list
Add any new element, if required
Modify any existing element, if required
Drop any existing element, if required
Generate the new target PSM
If the element can be mapped directly from the element Vanilla Model
If the element can be mapped from the nearest element Vanilla
Perform transformation.

3. Criteria for Successful Model Transformation

For any transformation tool to be a success, it should fulfill following functional requirements -

1. The tool should be able to create, modify retrieve and drop transformations.
2. One can reutilize the transformation model defined for one transformation from source platform to target platform to other set of source and target platform.
3. The transformation model must clearly define the termination condition and the output obtained from the transformation should be unique.
4. The transformation must be complete for each element in the source model; there should be a corresponding element in the target model that is created by a model transformation. A Traceability Matrix can be maintained to trace each and every element from source to target.
5. The tool should be relevant which means that it should be able to serve the practical purpose for which it is designed.

V. MODEL TRANSFORMATION USING VANILLA MODEL

1. Java Swing

Java Swing is part of Oracle JFC(Java Foundation Classes) and is a lightweight widget toolkit for Graphical User Interface.

Table 3. Vanilla to Swing mapping

S.No	Vanilla GUI – PIM	JAVA-Swing-PSM	Swing Class
1	VanillaComponent	SwingComponent	javax.swing.JComponent
2	VanillaContainer	SwingContainer	javax.swing.Container
3	VanillaWidget	NA	NA
4	VanillaFrame	SwingFrame	javax.swing.JFrame
5	VanillaMenuComponent	NA	NA
6	VanillaMenuBar	SwingMenuBar	javax.swing.JMenuBar
7	VanillaMenu	SwingMenu	javax.swing.JMenu
8	VanillaMenuItem	SwingMenuItem	javax.swing.JMenuItem
9	VanillaLabel	SwingLabel	javax.swing.JLabel
10	VanillaTextComponent	SwingTextComponent	javax.swing.text.JTextComponent
11	VanillaTextBox	SwingTextBox	javax.swing.JTextField
12	VanillaCheckBox	SwingCheckBox	javax.swing.JCheckBox
13	VanillaRadioButton	SwingRadioButton	javax.swing.JRadioButton
14	VanillaButton	SwingButton	javax.swing.JButton
15	VanillaTable	SwingTable	javax.swing.Jtable

16	VanillaImage	SwingImage	java.awt.Image
17	VanillaListBox	SwingListBox	javax.swing.JList<E>
18	VanillaComboBox	SwingComboBox	javax.swing.JComboBox<E>
19	VanillaTextArea	SwingTextArea	javax.swing.JTextArea
20	VanillaDatePicker	SwingDatePicker	javax.swing.JSpinner and javax.swing.SpinnerDateModel
21	VanillaTimePicker	SwingTimePicker	javax.swing.JSpinner and javax.swing.SpinnerDateModel
22	VanillaAudio	SwingAudio	can be played using library javax.sound.*
23	VanillaVideo	SwingVideo	can be Java Media APIs
24	VanillaDialogBox	SwingDialogBox	javax.swing.JDialog

*NA – No such component available in the Library

2. HTML-5

HTML5 (Hypertext Markup Language) [5] is the core Markup Language of the World Wide Web. The building blocks of HTML are HTML elements. The constructs of HTML include text box, button, check box, radio button, images , audio , video and many more.

Table 4. Vanilla to HTML mapping

S.No	Vanilla GUI – PIM	HTML- PSM	HTML Elements
1	VanillaComponent	HTMLComponent	DOM - Document <html>
2	VanillaContainer	HTMLContainer	<Form>
3	VanillaWidget	NA	NA
4	VanillaFrame	HTMLFrame	<frame>
5	VanillaMenuComponent	NA	NA
6	VanillaMenuBar	HTMLMenuBar	<div class="navbar">
7	VanillaMenu	HTMLMenu	<menu>
8	VanillaMenuItem	HTMLMenuItem	<menuitem>
9	VanillaLabel	HTMLLabel	Label text is directly added between body tags
10	VanillaTextComponent	NA	NA
11	VanillaTextBox	HTMLTextBox	<input type="text">
12	VanillaCheckBox	HTMLCheckBox	<input type="checkbox"
13	VanillaRadioButton	HTMLRadioButton	<input type="radio">
14	VanillaButton	HTMLButton	<button type="button">
15	VanillaTable	HTMLTable	<table>
16	VanillaImage	HTMLImage	<img>
17	VanillaListBox	HTMLListBox	<ul>, <ol>, <dl>
18	VanillaComboBox	HTMLComboBox	<select>
19	VanillaTextArea	HTMLTextArea	<textarea>
20	VanillaDatePicker	HTMLDatePicker	<input type="date"
21	VanillaTimePicker	HTMLTimePicker	<input type="time">
22	VanillaAudio	HTMLAudio	<audio>
23	VanillaVideo	HTMLVideo	<video>
24	VanillaDialogBox	HTMLDialogBox	<dialog>

3. Android –View Class

Android has very rich pre-defined built-in UI components library including layout objects and widgets to design and develop GUI for a mobile application. The View Class of the Android library is the parent class of entire widget toolkit to build an interactive application.

Table 5. Vanilla to Android mapping

S.No	Vanilla GUI – PIM	Android - PSM	Android Class
1	VanillaComponent	NA	NA
2	VanillaContainer	NA	NA
3	VanillaWidget	AndroidWidget	android.view.View
4	VanillaFrame	NA	NA
5	VanillaMenuComponent	NA	NA
6	VanillaMenuBar	NA	NA
7	VanillaMenu	AndroidMenu	android.view.Menu
8	VanillaMenuItem	AndroidMenuItem	android.view.MenuItem
9	VanillaLabel	AndroidLabel	android.widget.TextView
10	VanillaTextComponent	NA	NA
11	VanillaTextBox	AndroidTextBox	android.widget.EditText
12	VanillaCheckBox	AndroidCheckBox	android.widget.CheckBox
13	VanillaRadioButton	AndroidRadioButton	android.widget.RadioButton
14	VanillaButton	AndroidButton	android.widget.Button
15	VanillaTable	AndroidTable	TableLayout
16	VanillaImage	AndroidImage	android.widget.ImageView
17	VanillaListBox	AndroidListBox	android.widget.ListView
18	VanillaComboBox	AndroidComboBox	android.widget.Spinner
19	VanillaTextArea	AndroidTextArea	android.widget.EditText
20	VanillaDatePicker	AndroidDatePicker	android.widget.DatePicker
21	VanillaTimePicker	AndroidTimePicker	android.widget.TimePicker
22	VanillaAudio	AndroidAudio	android.media.MediaPlayer
23	VanillaVideo	AndroidVideo	android.media.MediaPlayer
24	VanillaDialogBox	AndroidDialogBox	android.app.AlertDialog

4. Python- Tkinter

Python has many GUI libraries for web development[6]. Tkinter is the most popular and de-facto GUI library for Python. It provides a powerful interface based on objectoriented concept to the Tcl/Tk widget set. It's portable across platforms line Windows, UNIX and Mac –OS.

Table 5. Vanilla to Android mapping

S.No	**Vanilla PIM**	**Python - PSM**	**Python- Tkinter**
1	VanillaComponent	NA	NA
2	VanillaContainer	PythonContainer	tkinter.Notebook
3	VanillaWidget	PythonWidget	tkinter.Widget
4	VanillaFrame	PythonFrame	tkinter.Frame
5	VanillaMenuComponent	NA	NA
6	VanillaMenuBar	NA	NA
7	VanillaMenu	PythonMenu	tkinter.Menu
8	VanillaMenuItem	PythonMenuItem	tkinter.Menubutton
9	VanillaLabel	PythonLabel	tkinter.Label
10	VanillaTextComponent	NA	NA
11	VanillaTextBox	PythonTextBox	tkinter.Entry
12	VanillaCheckBox	PythonCheckBox	tkinter.Checkbutton
13	VanillaRadioButton	PythonRadioButton	tkinter.Radiobutton
14	VanillaButton	PythonButton	tkinter.Button
15	VanillaTable	NA	NA
16	VanillaImage	PythonImage	PIL.Imagetk
17	VanillaListBox	PythonListBox	tkinter.Listbox
18	VanillaComboBox	PythonComboBox	tkinter.OptionMenu
19	VanillaTextArea	PythonTextArea	Tkinter.Text
20	VanillaDatePicker	PythonDatePicker	Tkinter.ttk.calendar
21	VanillaTimePicker	NA	NA
22	VanillaAudio	PythonAudio	simpleaudio.Wave Object
23	VanillaVideo	PythonVideo	cv2.VideoCapture
24	VanillaDialogBox	PythonDialogBox	Tkinter.tkMessage Box

VI. CONCLUSION AND FUTURE SCOPE

In this research paper, we proposed Vanilla Model – a Platform Independent Declarative Model for User Interface. In this approach, the user interface of web application is preserved and re-engineered with the help of Model Driven Architecture approach. We also proposed the steps to achieve complete re-engineering by first generating source Platform specific Model from source code. This source model is then transformed into the Vanilla Model. We can use Vanilla Model to generate any number of Platform specific Model and generating code from it.

REFERENCES

[1] A.R. Da Silva, "Model-driven engineering: A survey supported by the unified conceptual model." Computer Languages, Systems & Structures, 43, 139-155. (2015).

[2] R. Pérez-Castillo, I. G. R., de Guzmán, & M. Piattini. "Model-driven reengineering". In Emerging Technologies for the Evolution and Maintenance of Software Models (pp. 200-229). IGI Global.s(2012).

[3] A. Kleppe, J. Warmer,W. Bast. " MDA Explained , The Model - Driven Archietecture : Practise and Promise". Addison Welsey, 2003.

[4] of Lecture Notes in Computer Science. Springer- D. Akehurst, S. Kent. "A relational approach to defining transformations in a metamodel." 5th Int'l Conf. UML. Volume 2460 Verlag, 243–258.2002.

[5] Prince Singha, Aditya, Kunal Dubey, JagadeeswararaoPalli, "Toolkit for Web Development Based on Web Based Information System," Isroset-Journal (IJSRCSE), 6, no. 5, pp.1-5. 2018..

[6] Shubham, Deepak Chahal, LatikaKharb, "Security for Digital Payments: An Update," Journal (IJSRNSC), 6, no. 5 , pp. 51-54. 2018.

Authors Profile

Smita Agarwal has earned Bachelor's degree of Electronics & and Master's degree of Information Technology in 1998 &2001 respectively from University of Delhi. She is currently pursuing Ph.D. in Computer Science & Engineering.She has seven years of industry experience.

Alok Aggarwal received his bachelors' and masters' degrees in Computer Science& Engineering in 1995 and 2001 respectively and his PhD degree in Engineering from IITRoorkee, Roorkee, India in 2010. He has academic experience of 18 years, industry experience of 4 years and research experience of 5 years. He has contributed more than 150 research contributions in different journals and conference proceedings. Currently he is working with University of Petroleum & Energy Studies, Dehradun, India as Professor in CSE department.

Sarvottam Dixit did his Ph.D. in Physics (Material Science) from Dr. B.R. Ambedkar University Agra in 1990 and completed Post-Doctorate work from Tata institute of fundamental research (TIFR) Mumbai funded by DST in 1996 and M.E. in CSE. Current he is working as advisor to Chancellor and Professor in Faculty of Engineering in Mewar University. Earlier he was Pro-VC and acting Vice Chancellor Shri Venkateshwara University Gajurala (UP) and Venkateshwara Open University Arunachal Pradesh.

Dr. Adarsh Kumar received his Master degree (M. Tech) in Software Engineering from Thapar University, Patiala, Punjab, India, in 2005 and earned his PhD degree from Jaypee Institute of Information Technology University, Noida, India in 2016 followed by Post-Doc from Software Research Institute, Athlone Institute of Technology, Ireland during 2016-2018. Currently, he is working with University of Petroleum & Energy Studies, Dehradun, India as Associate Professor in School of Computer Science.