

MULTIPLE DEVICE MARKUP LANGUAGE A RULE APPROACH

SE690 Research Seminar Final Report

*Paul D. Johnson & Jinesh Parekh
Winter 2003*

ABSTRACT	3
INTRODUCTION	3
PROBLEM.....	4
ARCHITECTURE.....	4
INITIAL ARCHITECTURE DIAGRAM	5
FINAL ARCHITECTURE DIAGRAM	5
RULE EXAMPLES.....	7
<i>Example 1</i>	7
<i>Example 2</i>	9
DISPLAY ENGINE	10
HANDLER/CODE GENERATION	12
PROFILE	13
DIMENSIONS	14
EVENT MODEL.....	17
CONCLUSION.....	18
FUTURE WORK.....	19
<i>Mobile Profile</i>	19
<i>Desktop Profile</i>	25
PROJECT SCHEDULE.....	26
REFERENCES	27
APPENDIX A – SWING RULE SPECIFICATION.....	28
APPENDIX B – SWT RULE SPECIFICATION.....	37
APPENDIX C – PROPERTY FILE.....	46
MDML COMPONENTS	48
<i>Label</i>	48
<i>Button</i>	49
<i>CheckBox</i>	50
<i>Group</i>	51
<i>ComboBox</i>	52
<i>TextBox</i>	53
<i>TabbedPane</i>	54
<i>Panel</i>	55
<i>Start</i>	55
<i>Window</i>	56
<i>Dialog</i>	57
<i>ProgressBar</i>	58
<i>Slider</i>	59
<i>List</i>	60
<i>Separator</i>	61
<i>FlowLayout</i>	61
<i>BoxLayout</i>	61
<i>GridLayout</i>	61
<i>MenuBar</i>	62
<i>Menu</i>	63
<i>MenuItem</i>	64
<i>ToolBar</i>	65
<i>Tree</i>	65
<i>SplitPane</i>	65
<i>Table</i>	66
APPENDIX D – WALKTHROUGH.....	67

Abstract

XML User Interfaces provide an effective and innovative means to separate presentation from logic and data. XML User Interfaces provide faster and more efficient methods to change and maintain code. Traditionally, a Graphical User Interface (GUI) modification involves editing the source code, compiling, and executing the application. Yet with an XML User Interface Language a developer simply edits the XML file and restarts the GUI application. Examples of this can be seen in applications that produce interchangeable GUIs or skins.

The Multiple Device User Interface project is based on XML User Interface Language (XUL). One goal is to take a XML description and with that create a framework based on rules that specify navigation, layout, and components. Yet another goal of this project is create a generalized framework for Graphical User Interface development and to define the rule specification. There are several devices and each is different from the other with regard to its memory, display capabilities and internal representation of the data. Although a general representation can be used for all devices, this is sometimes thought to be too simplistic, thus the direction the User Interface framework will follow will produce a different rule set for different devices or perhaps produce different rule sets for the same device.

Introduction

Traditional methods for Graphical User Interface (GUI) programming can be tedious, cumbersome, and error prone. Newer methods use Graphical User Interface Markup Languages that spur new ideas and challenges for software engineers and User Interface designers. Both User Interface development and XML have become interesting research topics. Examples of the ongoing projects include: LUXOR, UIML, VOICEXML, and THINLETS. Some interfaces are extremely complicated with varying widgets and functionality (i.e. SWING, SWT, C#), while others are simple and straightforward (J2ME). This project will develop a new approach to User Interface design by developing a general scheme to compose Graphical User Interfaces, Multiple Device Markup Language (MDML), and also produce a generalized framework to generate code based on a User Interface toolkit.

One goal of this project is to provide a method to separate the GUI from structure and behavior. In order to make this separation a framework must be built to close the gap between sophisticated User Interfaces and simple User Interfaces by generating a Graphical User Interface based upon a rule specification. The rule specification provides specific pieces of information like widget and attribute mappings, composition mapping, imports, accessibility functions, and functionality. MDML is a general markup language designed for generalizing both sophisticated graphical User Interface toolkits and simple graphical User Interface toolkits. The results of this research are important for other frameworks and projects that pursue a clear distinction between presentation, structure and behavior (i.e. ZOOM Project). The design of one interface for multiple platforms could revolutionize the way User Interfaces are built in future applications.

This framework provides a mechanism for developers to develop GUI applications in a rapid and fast paced environment. Since defining a process and building objects from scratch in every instance requires a large amount of time and

resources, not to mention the issue of cost, this framework will provide a simple way for developers to change, maintain and update their Graphical User Interfaces. The framework is developed using Java technologies and uses the ZOOM Project's Code Generator.

Problem

There is a problem however. Can a User Interface be designed for multiple platforms independent of any structure or behavior? As technology improves so do end-user demands. The Industry requires fast deployment to multiple target platforms without the maintenance and development overhead of storing and developing every possible implementation (4). There is a growing and urgent need for systems that can accomplish this task. Some projects that are related to this problem are the Alternate Abstract Interface Markup Language (AAIML), Abstract User Interface markup Language (AUIML), Extensible Interface Markup Language (XIML), Extensible User Interface Language (XUL), Microsoft Extensible Application Markup Language (XAML), User Interface Markup Language (UIML), and W3C XForms (4). All of these projects are based upon XML.

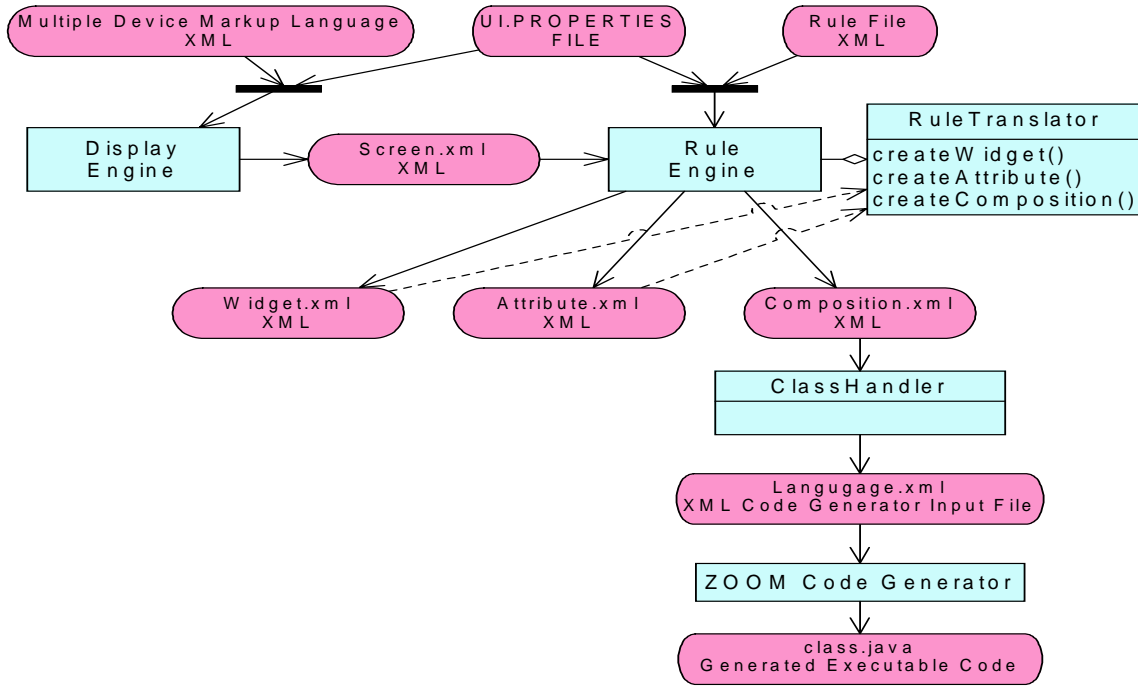
The Multiple Device Markup Language (MDML), the language introduced by this project, is based on XUL. MDML along with the framework does not re-implement XUL nor does it provide a one to one mapping for any toolkit similar to the LUXOR project. The parser is a general parser unlike the LUXOR parsers, which are tightly coupled to a toolkit. This project requires the user to define mappings between MDML tags and GUI toolkit components. This mapping is called the rule specification.

Architecture

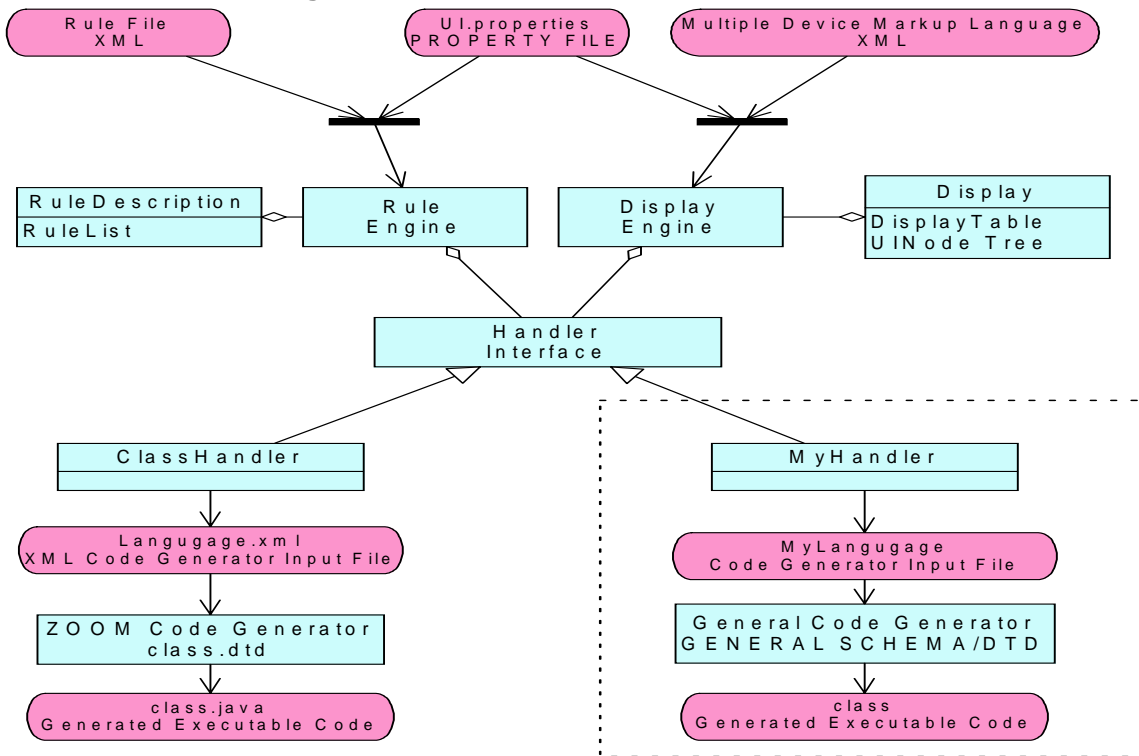
The initial architecture was based on a series of XML transformations to produce the final language file for input to the code generator. The architecture had two major problems. Firstly, the XSL transformations became complicated requiring a lot of templates, and second, composition of advanced widgets became increasingly difficult to generalize. The later is the reason the architecture changed since it often required special case scenarios in the rule engine. In general if special case scenarios are coded then coupling becomes tighter between a MDML tag and a GUI toolkit widget, which leads to difficulty for other devices. So, the decision was made to alter the initial architecture thereby using java instead of XSL transformations.

There are four parts to the final UI architecture rule engine, display engine, handler, and code generation. The rule engine will parse the rule file. The display engine will parse the MDML file and handle the layout. The handler will generate a XML file based on the generalized class – class.dtd. The code generation will generate an executable language file. All three parts are Java based applications. The properties file contains specific pieces of information needed by the rule and display engine see Appendix C for details.

Initial Architecture Diagram



Final Architecture Diagram



Rule Engine

Rules provide a mechanism to prevent errors and enhance User Interface design. The rule engine is responsible for reading the rule file -- i.e. rule.XML. The rule file has three sections <profile>, <tag>, and <event>. The <profile> tag may contain these tags <import>, <toolkit>, <accessor> and <topfunction>. The <import> tag is used for any imports or includes that must be specified at the beginning of any language file. The <toolkit> tag will specify a GUI toolkit which is required by the framework to be based on a OO language i.e. Swing, SWT, or C#. The <accessor> tag will produce a method that will allow a back end source to access any specific component. The <topfunction> is the function used display the initial window or frame. See Appendix A.

The <event> section has six tags <action>, <focus>, <window>, <mouse>, <mousemotion>, and <selection>. The <action> tag is for any mouse click, keyboard pressed or released, or menu selection. The <focus> tag is for a component when it gains or loses the focus. The <window> tag is used when a window event occurs, i.e. window closes. The <mouse> tag is used when a user presses the mouse button over a component. The <mousemotion> tag is used when a user moves the mouse over a component. Last the <selection> tag is used when an element is selected inside a component. This is all that is required for any desktop profile. Other profiles can build from these mapping i.e. PDA can substitute can use a pen event for the <mouse> tag.

The <tag> section is compromised of all the MDML tags that need to mapped or ignored in some cases. Each <tag> section will contain multiple <rule> tags. Each <rule> tag will contain a MDML tag i.e. <TextBox>. The MDML tag section will contain four tags <widget>, <attribute>, <define>, and <composition>. The <widget> tag must specify the GUI toolkit mapping and optionally whether or not if a constructor is required. The <attribute> tag will specify how the MDML attribute is mapped to the GUI toolkit attribute. The <define> tag is used to specify any required definitions before or after the initial widget is defined. All methods under the <define> tag have generated names. The <composition> tag is used to specify how a particular component or element is added to its <widget> tag. See Appendix A for further details.

In general to specify a method or constructor under the <widget>, <attribute> and <composition> tags use the <method> and <constructor> tag. A <method> is used in the <attribute>, <define>, and <composition> tags to specify the method used in each case. The <constructor> tag is only used in the <widget> and <composition> case to specify that normal definition is not appropriate but use constructor definition instead. Each <method> and <constructor> can have any number of <Parameter> tags. Common variables used with <parameter> tags are *\$Parent*, *\$TopFunction*, *\$Default*, *\$(MDML Tag)*, and *\$Counter*. See Appendix A for further details.

Rule Examples

Example 1

The Swing toolkit provides a JTREE component for displaying hierarchical components; provide a rule for this component.

Step 1

Find existing rule file (i.e. rule.XML) or create new rule file. If this is a new rule file remember to update ui.properties file with the rule filename. Edit rule file with a text editor or XML specific editor.

Step 2

Add the new rule with the <rule> tag and specify an MDML tag association.

```
<rule>
  <Tree>
</Tree>
</rule>
```

Step 3

Add the widget mapping using the <widget> tag. In Swing it is tempting to write this...

```
<rule>
  <Tree>
    <widget> JTree </widget>
  </Tree>
</rule>
```

but in Swing one can not add elements to a JTREE. Elements are not added directly to a JTREE they are added to a DefaultMutableTreeNode. So in this case the correct rule should look like

```
<rule>
  <Tree>
    <widget> DefaultMutableTreeNode </widget>
  </Tree>
</rule>
```

Step 4

Add the attribute information using the <attribute> tag. Each attribute section contains 1 or more <Method> tags. The Method in this case needs two attributes name and type. Name is the name that the rule file can use to associate it with the MDML attribute (attribute mapping). Type is required since the type will determine how the attribute from the MDML tag is formed. I.e. String will contain quotes "", text and numbers will contain no quotes.

```

<rule>
  <Tree>
    <widget> JTree </widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
  </Tree>
</rule>

```

Step 5

Define the JTree Component. Remember we have just defined the DefaultMutableTreeNode so elements can be added but we still need to define the JTree component because only the JTree can be added to a container in Swing. Use the <define> tag to define other functionality that needs to be associated with a MDML tag TREE.

```

<rule>
  <Tree>
    <widget> JTree </widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
    <define>
      <Method name="myTree" order="post" type="JTree">
        <Parameter>$Tree</Parameter>
      </Method>
    </define>
  </Tree>
</rule>

```

Step 6

Define the composition mapping. How do we add anything to our TREE component? In MDML a tree is composed of NODE(s). So we must tell the framework how to attach these NODE(s). In our case use the <METHOD> tag again and specify a method by using the name attribute.

```

<rule>
  <Tree>
    <widget> JTree </widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
    <define>
      <Method name="myTree" order="post" type="JTree">
        <Parameter>$Tree</Parameter>
      </Method>
    </define>
    <composition>
      <Node>
        <Method name="add"/>
      </Node>
    </composition>
  </Tree>
</rule>

```

Example 2

Provide the rule for a MDML Node using Swing.

Step 1

Create a rule with a MDML tag.

```
<rule>
  <Node>
  </Node>
</rule>
```

Step2

Provide the MDML tag association with a swing component. In Swing A DefaultMutableTreeNode can be created using a constructor – DefaultMutableTreeNode node = new DefaultMutableTreeNode("String"). The rules provide a convenient way to do this using a <Constructor> tag similar to the <Method> tag. Here the \$text is a lets the framework know, a variable must exist named text. In this case the MDML file contains a text attribute so it will map to whatever the user sets the text to.

```
<rule>
  <Node>
    <widget>DefaultMutableTreeNode
      <Constructor>
        <Parameter type="String">$text</Parameter>
      </Constructor>
    </widget>
  </Node>
</rule>
```

Step3

Provide the composition. Composition exists because a node can contain other nodes.

```
<rule>
  <Node>
    <widget>DefaultMutableTreeNode
      <Constructor>
        <Parameter type="String">$text</Parameter>
      </Constructor>
    </widget>
    <composition>
      <Node>
        <Method name="add"/>
      </Node>
    </composition>
  </Node>
</rule>
```

Display Engine

The display engine is responsible for reading the initial MDML file into memory and changing layout as desired by the rules. The display engine uses Document Object Model (DOM) to initially capture the MDML file into memory. Every tag is translated into a User Interface Node (UINode). All display engine objects, component(s), bargroup(s), container(s), and layout(s), will inherit from the UINode creating a UINode Tree.

The display engine contains two data structures. One is a tree that mimics the DOM with extended functionality, and the other is a HashTable that provides fast lookup by id for any UINode object. The DOM is not sufficient to use in producing the code since there are a number of things that are needed to be automatically generated and added to the tree i.e. id number. The HashTable is useful when composing a component on the container since reading the whole tree again is redundant, use a hashing mechanism to perform fast lookups for any UINode.

The input is a MDML file that is a valid XML file and is based on the UISchema.xsd. The objective of MDML is to define a complete general set of tags for different and varying platforms and provide simple and convenient documentation and usage examples. See Appendix D for explanation and details of each MDML component. There are two approaches for defining a complete set of tags for any GUI toolkit. One approach includes the minimum set of functionality provided by the toolkit and the other includes the maximum set of functionality. The minimum set is a subset of the maximum set.

One advantage in using the minimum set of functionality is the mapping of tags to the larger set is trivial since the minimum set is a subset of the larger set. A disadvantage to this approach is the majority of maximum set of functionality remains useless; rendering non-fancy GUIs on a desktop. A disadvantage to using the maximum set of functionality is that the minimum set may not map directly to the maximum set, thus some work is required to enhance the mapping. But, this is justified, as the complete set gains the advantages of the maximum set of functionality while still being able to transform to the minimum set. So, if a tag does not exist in the minimum set a useable tag(s) must be mapped or omitted to that individual tag i.e. <toolbar> is omitted. Thus, this project will use the maximum set of functionality to describe a complete set of tags for the MDML description. Below is a sample calculator written in with MDML.

Calculator Application Example MDML

```
<?XML version="1.0" encoding="UTF-8"?>
<Start name="CalculatorGUI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="J:\eclipse\workspace\Zoom\src\resources\ui\UISchema.xsd">
  <Window id="String" name="Calculator" show="true">
    <MenuBar name="menuBar1">
      <Menu text="Edit">
        <MenuItem mnemonic="c" text="Copy"/>
        <MenuItem mnemonic="p" text="Paste"/>
      </Menu>
      <Menu text="View">
        <MenuItem mnemonic="n" text="Standard"/>
        <MenuItem mnemonic="u" text="Scientific"/>
        <MenuItem mnemonic="c" text="Digital Grouping"/>
      </Menu>
      <Menu text="Help">
        <MenuItem mnemonic="h" text="Help Topics"/>
        <MenuItem mnemonic="a" text="About Calculator"/>
      </Menu>
    </MenuBar>
    <Panel>
      <BoxLayout axis="1"/>
      <Panel>
        <FlowLayout/>
        <TextBox type="text" editable="false" columns="40"/>
      </Panel>
      <Panel>
        <GridLayout rows="1" columns="4" icolumns="4"/>
        <Button text="/">
        <Button text="Backspace"/>
        <Button text="CE"/>
        <Button text="C"/>
      </Panel>
      <Panel id="Calculator Panel" name="JCalcPanel">
        <GridLayout rows="4" columns="6" icolumns="6"/>
        <Button text="MC"/>
        <Button text="7"/>
        <Button text="8"/>
        <Button text="9"/>
        <Button text="/">
        <Button text="sqrt"/>
        <Button text="MR"/>
        <Button text="4"/>
        <Button text="5"/>
        <Button text="6"/>
        <Button text="**"/>
        <Button text="%/>
        <Button text="MS"/>
        <Button text="1"/>
        <Button text="2"/>
        <Button text="3"/>
        <Button text="-"/>
        <Button text="1/x"/>
        <Button text="M+"/>
        <Button text="0"/>
        <Button text="+/-"/>
        <Button text="."/>
        <Button text="+"/>
        <Button text="="/>
      </Panel>
    </Panel>
  </Window>
</Start>
```

Handler/Code Generation

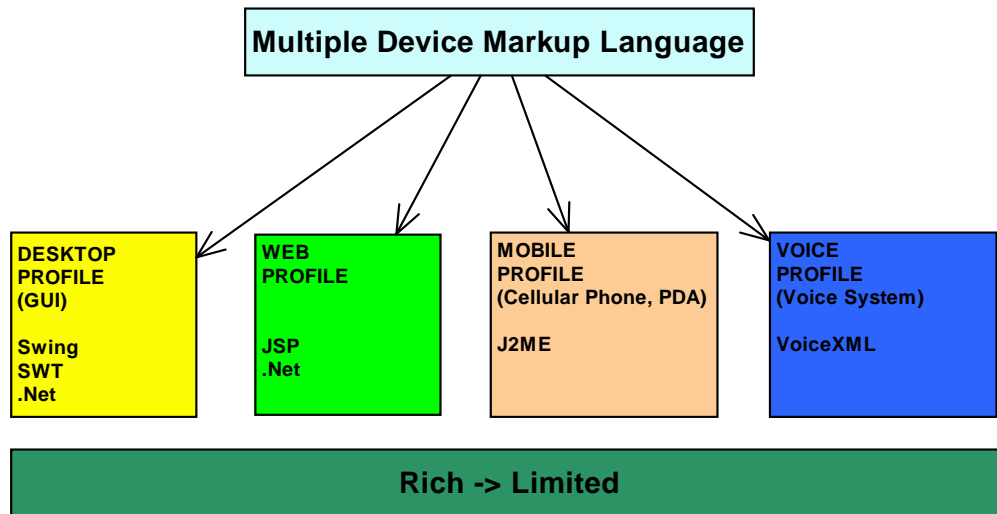
The handler is general interface that contains access to the rule set and the MDML display object. The classHandler is a handler that inherits from the handler class. Its responsibility is to generate a language specific XML file based on the class.dtd. This file produced is called Language.XML. The Language.XML file is passed to the code generator. To produce executable code this project uses the ZOOM Code Generator. The ZOOM Code Generator can generate executable code from a valid class.dtd or class-swing.dtd. This project uses the class.dtd since the toolkit is not known before the profile is read. The class.dtd is general schema for any Java class.

Language.XML/Swing GUI Code

```
<Method isStatic="false" accessibility="public" returnType="void"
name="createComponents">
  <CodeTemplate>
    <CodeBody>
      <![CDATA[Calculator = new JFrame();]]>
      <![CDATA[Calculator.setTitle("Calculator");]]>
      <![CDATA[menuBar1 = new JMenuBar();]]>
      <![CDATA[Calculator.setJMenuBar(menuBar1);]]>
      <![CDATA[menuBar1.setName("menuBar1");]]>
      <![CDATA[JMenu4 = new JMenu();]]>
      <![CDATA[menuBar1.add(JMenu4);]]>
      <![CDATA[JMenu4.setText("Edit");]]>
      <![CDATA[JMenuItem5 = new JMenuItem();]]>
      <![CDATA[JMenu4.add(JMenuItem5);]]>
      <![CDATA[JMenuItem5.setText("Copy");]]>
      <![CDATA[JMenuItem6 = new JMenuItem();]]>
      <![CDATA[JMenu4.add(JMenuItem6);]]>
      <![CDATA[JMenuItem6.setText("Paste");]]>
      <![CDATA[JMenu7 = new JMenu();]]>
      <![CDATA[menuBar1.add(JMenu7);]]>
      <![CDATA[JMenu7.setText("View");]]>
      <![CDATA[JMenuItem8 = new JMenuItem();]]>
      <![CDATA[JMenu7.add(JMenuItem8);]]>
      <![CDATA[JMenuItem8.setText("Standard");]]>
      <![CDATA[JMenuItem9 = new JMenuItem();]]>
      <![CDATA[JMenu7.add(JMenuItem9);]]>
      <![CDATA[JMenuItem9.setText("Scientific");]]>
      <![CDATA[JMenuItem10 = new JMenuItem();]]>
      <![CDATA[JMenu7.add(JMenuItem10);]]>
      <![CDATA[JMenuItem10.setText("Digital Grouping");]]>
      <![CDATA[JMenu11 = new JMenu();]]>
      <![CDATA[menuBar1.add(JMenu11);]]>
      <![CDATA[JMenu11.setText("Help");]]>
      <![CDATA[JMenuItem12 = new JMenuItem();]]>
      <![CDATA[JMenu11.add(JMenuItem12);]]>
      <![CDATA[JMenuItem12.setText("Help Topics");]]>
      <![CDATA[JMenuItem13 = new JMenuItem();]]>
      <![CDATA[JMenu11.add(JMenuItem13);]]>
      <![CDATA[JMenuItem13.setText("About Calculator");]]>
      ....
      <![CDATA[Calculator.show();]]>
    </CodeBody>
  </CodeTemplate>
</Method>
```

```
public void createComponents() {
    Calculator = new JFrame();
    Calculator.setTitle("Calculator");
    menuBar1 = new JMenuBar();
    Calculator.setJMenuBar(menuBar1);
    menuBar1.setName("menuBar1");
    JMenu4 = new JMenu();
    menuBar1.add(JMenu4);
    JMenu4.setText("Edit");
    JMenuItem5 = new JMenuItem();
    JMenu4.add(JMenuItem5);
    JMenuItem5.setText("Copy");
    JMenuItem6 = new JMenuItem();
    JMenu4.add(JMenuItem6);
    JMenuItem6.setText("Paste");
    JMenu7 = new JMenu();
    menuBar1.add(JMenu7);
    JMenu7.setText("View");
    JMenuItem8 = new JMenuItem();
    JMenu7.add(JMenuItem8);
    JMenuItem8.setText("Standard");
    JMenuItem9 = new JMenuItem();
    JMenu7.add(JMenuItem9);
    JMenuItem9.setText("Scientific");
    JMenuItem10 = new JMenuItem();
    JMenu7.add(JMenuItem10);
    JMenuItem10.setText("Digital Grouping");
    JMenu11 = new JMenu();
    menuBar1.add(JMenu11);
    JMenu11.setText("Help");
    JMenuItem12 = new JMenuItem();
    JMenu11.add(JMenuItem12);
    JMenuItem12.setText("Help Topics");
    JMenuItem13 = new JMenuItem();
    JMenu11.add(JMenuItem13);
    JMenuItem13.setText("About Calculator");
    ...
}
```

Profile



The framework will establish a profile for each target platform. The profiles are desktop, web, mobile, and voice. The desktop profile contains the maximum set of functionality with powerful, and rich toolkits Java Swing, Java SWT and .NET (C#, C++, VB). These APIs may be used to develop very sophisticated graphical User Interfaces i.e. Microsoft Outlook and Eclipse. They have access to the complete set of widgets. Space and memory are not limitations in these environments for implementation purposes; these issues however do have performance impacts. Screen length and width are usually 15" to 21" (diameter) so visual layout is limitless – within scope. This project will focus on the DESKTOP profile and the rule specification. See Future Work for further information.

The second profile is the WEB profile which is a subset of the first profile because the WEB has some constraints. The web toolkits HTML, XHTML, JSP can delivery fancy interfaces although with certain limitations. Widgets used on a desktop application might cause performance issues on WEB presentations since the presentation is tied to a particular browser rendering capability. Other widgets might look awkward or non-functional i.e. a toolbar on a WEB page, this looks great on a DESKTOP application but awkward on a WEB page since the browsers already has a toolbar. Space and memory are issues since the WEB applications cannot store enormous amounts of information due to session limitations. Also, constant database hits could cause the WEB applications to be slow and useless. Screen length and width depend on the browser; but usually this is not an issue for display.

The third profile is the MOBILE profile. This profile can be separated into two separate profiles those with less than 2MB memory and those with 2MB or greater memory. Both profiles have limited user input devices; keyboard and/or mouse are typically not available (2). Space and memory are limited due to size of the device and available power source on the device (2). Since the CPU has limited processing any instructions are expensive so should only be done if there

is a strong justification (2). “A wireless device has limited bandwidth, compared to a wired device, for transmitting and receiving data” (2). Also, intermittent connections are typically not reliable since a persistent point-to-point connection is difficult.

The last profile is the VOICE profile. This profile has gained much attention in the past year with the introduction of voiceXML. “Its major goal is to bring the advantages of web-based development and content delivery to interactive voice response applications” (3).

Dimensions

There are three independent dimensions to any profile layout, language, and widgets. The layout dimension is concerned with space, look, and feel. The language dimension handles different choices of object oriented (OO) languages used to display the User Interface. The widget dimension involves mapping MDML tags to actual User Interface toolkit specific components or methods.

The layout dimension is compromised of four different layout managers desktop, web, mobile and voice. The desktop layout manager produces the exact same layout as described MDML and is the superset. All other layouts are subsets. There is a 1:1 mapping between the MDML and the layout generation for the desktop platform. The web layout manager is similar but there are some restrictions on layout compared to desktop. Toolbars are generally not common in web applications so they are omitted. Menu bars are generally not at the top of web applications but they cannot be omitted because the user needs to navigate through the application. Menu bars need to be created but placed on the left side of the page’s content. Another major difference is web applications use tables for managing layout whereas desktop applications use layout managers i.e. flow, box, or grid. If the table tag is used then there is 1:1 mapping if the <box> tag is used then a conversion needs to happen between the <box> tag and the HTML table. If the <layout> tags are used they should be converted in some way to HTML’s particular table layout. This is difficult because dynamic content is not known at runtime so the size of the table must be assumed or displayed column-by-column or cell-by-cell.

The mobile layout manager is the most restrictive layout manager. There are two issues that involve layout for mobile units display and input. Limited display or space is a challenge to any mobile device since the display is small compared to any desktop monitor. The power of mobile technology is in part due to its compactness, but application development is not the main goal of these devices. With the J2ME MIDP, usually full scale applications do not exist but rather smaller applications called MIDlets, which require limited input because entering text is awkward; there is usually no keyboard only a numeric pad. Mobile User Interface programming is difficult, but not impossible because technology is changing rapidly.

The voice layout manager is simple; there is no layout in terms of display. But there is a layout in terms of navigation. A user should only be able to do things in some hierarchical order. For example, if a user does not give the

correct username and password then they should not be allowed to proceed any further, not only for security purposes but for application integrity and flow.

The desktop will focus on the Java language. In theory, our description is generic and could choose other languages that include C#. This project will concentrate on the Java language because Java provides a cross platform mechanism and easy to learn GUI toolkits. Java has three main GUI toolkits AWT, Swing, and SWT. This paper will provide examples for Swing and SWT toolkits. Web application use HTML for static contents but for dynamic content there are a multitude of chooses JSP (Java), ASP (.NET), or DHTML. Mobile applications commonly use C, Java (J2ME), or OS specific languages for development.

All languages with GUI toolkits have a set of widgets. Widgets are components or controls used in GUI applications. The MDML defines general tags based on XUL (XML User Interface Language) transformed into some GUI toolkit specific widget or component. In the next chart shown there is a mapping from general MDML tag to Java based GUI toolkit component.

Tags	Swing	SWT	J2ME
<checkbox>	JCheckBox	Button.SWT.CHECK	List
<textbox>	JTextField	Text	TextField
	JTextArea	Text SWT.MULTI	
	JEditorPane	StyledText	
	JTextPane		
<button>	JButton	Button.SWT.PUSH	Command
	JToggleButton	Button.SWT.TOGGLE	
<group>	ButtonGroup	Group	*
<label>	JLabel	Label	StringItem
<menu>	JMenu	MenuItem	Command/List
<menubar>	JMenuBar	Menu	List
<menuItem>	JMenuItem	MenuItem	Command
	JCheckBoxMenuItem	MenuItem SWT.CHECK	
	JRadioMenuItem	MenuItem SWT.RADIO	
<panel>	JPanel	Composite	IGNORED
	JScrollPane	ScrolledComposite	
<image>	ImageIcon	Image	ImageItem
<list>	JList	List	List
<combobox>	JComboBox	Combo	List
<boxlayout>	BoxLayout	FillLayout	IGNORED
<flowlayout>	FlowLayout	RowLayout	IGNORED
<gridlayout>	GridLayout	GridLayout	IGNORED
		GridBagLayout	
<toolbar>	JToolBar	ToolBarItem SWT.RADIO SWT.CHECK SWT.PUSH SWT.SEPARATOR	IGNORED
<progressbar>	JProgressBar	ProgressBar	*
<separator>	JSeparator	Label SWT.SEPARATOR	IGNORED
<slider>	JSlider	Slider	Gauge
<tabbedpane>	JTabbedPane	TabFolder	*
<table>	JTable	Table, TableColumn, TableItem	CustomItem
<window>	JFrame	Shell	Form
<frame>	JInternalFrame	*	*
<dialog>	JDialog	Shell	Form Alert
<splitpane>	JSplitPane	Sash	IGNORED
<tree>	JTree	Tree	*
<customecomponent>	*	*	*

Event Model

The event model is general and shares the Swing naming convention. In this framework there exist six basic event types – action, focus, selection, mouse, mouse-motion, and window. The user of the framework is responsible for implementing the functionality or classes for these events. The framework requires the name of these classes to be specified in the rule section. Each event type tag contains a <Field> tag. The <Field> tag contains two attributes name and method. Name specifies the variable name and method specifies the method to use with the components. Under the <Field> tag the user must specify the type. The <type> tag specifies the class that the backend functionality must implement. Also, if the functionality is in a different package or subsystem, that package or subsystem must be specified in the <import> section of the profile.

Example 1

Define rules for Swing Events. The <Field> is used here with two attributes name and method. Name is the variable name and method is the method used with the component. The <type> in this case is the class that the user must define outside the framework to handle any events.

```
<Event>
  <Action>
    <Field name="actionListener" method="addActionListener">
      <Type>ActionEventHandler</Type>
    </Field>
  </Action>
  <Focus>
    <Field name="focusListener" method="addFocusListener">
      <Type>FocusEventHandler</Type>
    </Field>
  </Focus>
  <Window>
    <Field name="windowListener" method="addWindowListener">
      <Type>WindowEventHandler</Type>
    </Field>
  </Window>
  <Selection>
    <Field name="selectionListener" method="addListSelectionListener">
      <Type>ListSelectionEventHandler</Type>
    </Field>
  </Selection>
  <Mouse>
    <Field name="mouseListener" method="addMouseListener">
      <Type>MouseEventHandler</Type>
    </Field>
  </Mouse>
  <MouseMotion>
    <Field name="mouseMotionListener" method="addMouseMotionListener">
      <Type>MouseMotionEventHandler</Type>
    </Field>
  </MouseMotion>
</Event>
```

Conclusion

Many projects approach building an XML User Interface by tightly coupling XML tags to components or methods (i.e. THINLETS, LUXOR). This project is loosely coupled since the rule specification provides the mapping between XML tags and components. For example, what does it mean to have a <MENUBAR> in our framework? With relation to a desktop profile it means a menu bar will be attached to the given window or dialog, yet in a mobile environment menu bar do not normally exist. So in this framework because the tags are loosely coupled it may mean a menu bar is mapped to a tree component or list component in order to provide some form of navigation to the end user displaying possible applications. This framework will allow the end user to change the component mapping through XML rules without modifying the implementation.

Another benefit of this framework is that a developer is not tied to one code generator. Although, this project uses the ZOOM code generator, one could develop an entirely new Handler from the *Handler* interface. The new Handler could produce the desired input for another generalized code generator or use a XSL Transformation to transform the *Language.xml* into a valid input for a new code generator. Lastly, with this framework, the ZOOM model of separation between User Interface, behavior, and structural is proven for the User Interface. The UI model can be generated independent of any behavior or structure.

Using this framework requires thinking about design first and code later since the code is automatically generated. Unlike the traditional approach code and run the design is extremely import in this framework since the idea is it can be used on multiple platforms and devices with varying degrees of richness and limitations. Also, to the novice, it might appear that this framework requires a lot of overhead, rule specification and xml description to complete an application. But, once the rule specification is written generally it doesn't have to be changed unless the application rendered doesn't conform to the initial design.

Future Work

Mobile Profile

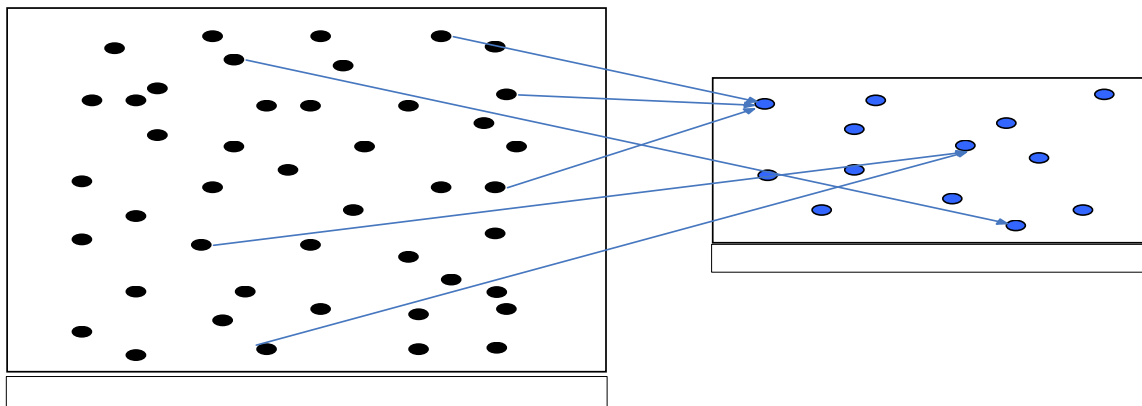
As stated earlier the goal of the project is to produce a GUI for multiple platforms and devices not only to high-end desktop devices (Desktop Profile) but also support the low-end devices like MIDP-Mobile Information Device Profile. This section discusses extending the framework to support low-end devices using the MIDP Profile.

To support the mobile profile (J2ME) the display engine must be enhanced to produce an alternate layout from the original MDML file. There are two options, let the user produce an alternate MDML file for the mobile profile, or by adding new rules dynamically produce an alternate layout. In other frameworks like UIML the first option is used to generate GUI's across profiles. In this framework, the latter is desired, since the rule specification contains the toolkit characteristics.

There are two approaches to dynamically produce convert MDML file into an application for low-end devices. The first approach uses direct widget mapping technique. All widgets and layouts that can be defined in the MDML are mapped to the widgets of the MIDP API or any other API that is to be used. So for example if the MDML defines a button then the corresponding behavior could be mapped onto the Action button on the mobile devices. Similarly text box definition in the MDML file could be mapped to the corresponding element of the MIDP API.

PROBLEM:

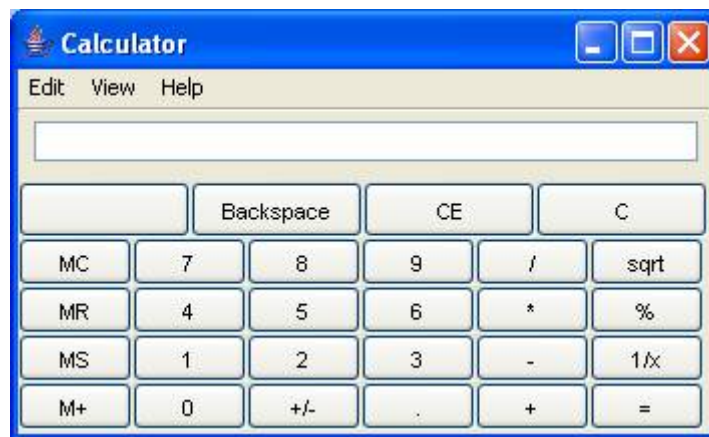
The first approach has a severe drawback. Remember MDML is required to map between sophisticated GUI (Desktop) and non-sophisticated GUI (Mobile). The MIDP API does not support most of a these widgets in a sophisticated GUI. In order to map all the widgets defined in the MDML schema, several MIDP widgets and layouts have to map to a single or no widgets in the MIDP profile. The following diagram shows a MDML tag to MIDP Widget mapping.



The number of widgets in the MDML clearly out number the number of widgets in the MIDP API and there is no one to one mapping; instead there is a group of widgets in the MDML that map to the same widgets in the MIDP set. This indicates that one or more functionalities in the MIDP will be delivered using the same MDML Tag. So the first approach mapping is naïve, the number of widgets that is required to display the GUI on the low-end device is not equal to the number of widgets that is defined in the GUI definition using MDML.

Another drawback is the limited screen size of the low-end device. It is absolutely impossible to display a sophisticated desktop GUI on a single page/screen of the mobile devices. So it is obvious that the single page desktop application must be divided into multiple pages on the mobile device. The MDML schema can provide a way to tell the rule engine about the page separations but there is no exact way to dynamically create these page separations. This could lead to a complete disaster.

Consider the following calculator application from the MDML that is displayed on the desktop.



This application is a simple desktop application. However it is impossible to map the complete application on the display to a small mobile device. This application could show six to eight widgets on a single screen as opposed to the 41 widgets that shows up on the desktop application. Thus it is clear that this application needs to be separated into multiple pages. If the rule engine is allowed to perform a naïve break down, then it will show a few buttons on one page and a few buttons on the other pages thus partitioning the other buttons in the same manner.

Thus naïve mapping does not break the application logically, thus making it difficult to use.

Though the application only demonstrates the problems with widgets, the same applies to layouts and other unsupported widgets like the menu bar. Also the application developer must develop the application recalling that eventually it will be required to be mapped to more than one profile, in this case a low-end device.

The above examples demonstrate major difficulties that must be addressed in order to make a sensible and usable transition from a desktop application to a mobile application. It clearly indicates that the rule engine below could be naïve and rendering a complete useless application so the framework must incorporate reasonable amount of intelligence.

Solution:

The above examples are summarized below:

- Number of widgets required in the MIDP is equal to the number of widgets in the MDML definition.
- Logical elements are not grouped together

The solution lies in *shrinking* the desktop application. Instead of mapping the widgets naively, an attempt must be made to map the functionality as compared to the display in a way such that it reduces the number of widgets that are required for display. In this approach, the intelligence is incorporated inside the rule engine to produce the desired output. There are a number of intricate details about the device that the rule engine profile must know. Some of them are listed below:

- Size of the MIDP display.
- The widget list that it supports.
- Constraints under which the rule engine should work.
- Other details such as the support for color and other GUI applications.

The process that is described below should be used iteratively until the application could be easily displayed on the mobile device or the application could not go any further reduction.

Below is the list of steps that the application should iterate through:

1. Chunking
2. Reduction
3. Smart Mapping
4. Elimination

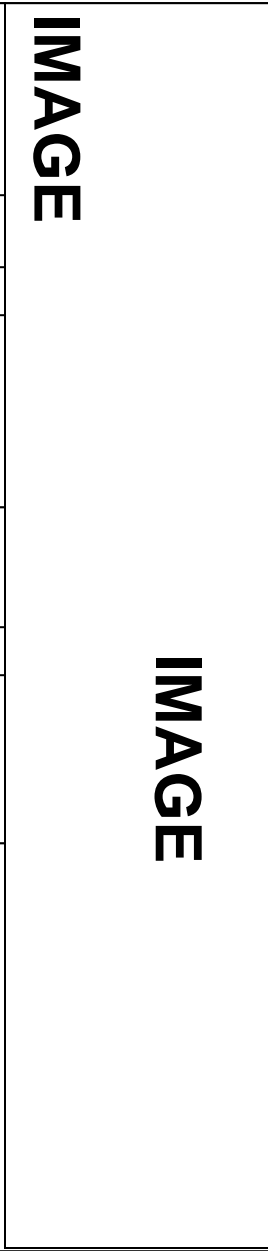
Each of these steps is described below using an example. Consider the below desktop application that is required to be shrunk into a mobile application.

Heading with some Graphical Effects

250 lines of text describing some article. Mobile device would display at the most half line per page and the user will have to scroll 1000 times in order to read the whole article. The user will surely be unhappy.

Another 150 lines of text. The user might not even reach to this as he might have already quit after reading some 5 to 10 pages of the above title.

0				
1	2	3	+	-
4	5	6	*	/
7	8	9	Text <input type="text"/>	



Below is the a description of the above steps to shrink the application

Chunking

In the first step the application needs to be separated into multiple logical pages. The application has four logically distinct and separate parts, the title part, two text articles and the small calculator application. So the chunking process should mark the application into four logically separate parts.

Reduction

The text articles are way too long to be transported to the mobile application. It is surely possible to reduce that one. There are various automatic text reduction algorithms that could work around the problem based on various things and reduce the text to some key words that would merely give the gist of the article. A good way is to take the help of the publisher of the article and reduce it. It is certainly not possible to shrink it to only key words, so some of them could be a link to a detailed description optionally that could be directed to but only if the reader wishes to.

Smart Mapping

This is the most interesting and important thing that should be understood very well. As mentioned earlier it might not be possible to display all the forty-one widgets in the calculator application on the same page. It is now required to reduce the number of widgets. Given a thought the number widgets could be mapped to a single widget – the text box. Also the four operation widgets could be mapped with just one widget – the command button thus transforming the application into a three widget application from forty-one that could be easily displayed on a single page. It is not always possible to reduce an application so drastically, but it is always possible to map a large number of widgets to a fewer one which helps in reducing the number of widgets to be displayed.

Elimination

If the image is not giving any information like a logo than it could be eliminated, as it would just prove to be an overhead in the mobile application. Also the title could be displayed without any graphical effects as a trade off for efficiency and space.

This process could be iteratively used to shrink the application. Also there are specialized web browsers that could aid in displaying these images. These browsers also have capabilities to display large images providing a magnifying widget that rolls over the small image on the horizontal plane.

Below is the final look at the application after iterative process.

Part 1

Heading

Some 25 lines of text. The user will have to scroll 10 times. The user will be happy.

Part 2

Another 15 lines of text. The user this.

Part 3

Input Text

Command

Div

Mul

Sub

Add

The first approach that was used to shrink the application is a software engineering technique. However there is a second approach that merges software engineering techniques with the field of Human Computer Interaction (HCI). In order to accomplish this task, this project will branch off into several different projects to complete the work for the mobile profile. These projects will concentrate on solving the above problems using methods and theories from HCI.

Desktop Profile

Desktop Profile Itemized Future Updates:

1. MDML/RuleEngine - Support Enhanced Layout Manager i.e. BorderLayout, FrameLayout
2. Event Handling - Support events within each rule.
A user might not want to code for general event cases. Allow for specific listeners or event handlers.

Project Schedule

1. Initial Groundwork (Summer)
 - a. Initial Presentation (June)
 - b. Language Decision – Swing, SWT (June)
 - c. Research components, attributes, composition, etc (June, July, August)

2. Architecture (Summer)
 - a. Architecture Design (August)
 - i. XSLT
 - ii. Java
 - b. Initial Architecture Complete (September)
 - c. Intermediate Presentation (Fall)

3. Software Design (Fall)
 - a. Diagrams (October)
 - b. Coding (October, November, December)

4. Architecture Re-work (Fall)
 - a. Architecture redesigned (December / 1 Week)
 - b. Final Architecture Complete (December 12th)

5. Code (Fall)
 - a. Final Implementation (December 2 weeks, January 2 weeks)
 - b. Complete (January 10th)

6. Final Deliverable (Winter)
 - a. Testing examples (January)
 - b. Final Paper (February 6th)
 - c. Final Presentation (February 6th)

The initial project was scheduled for six months. Although, one might questions does a software engineering project ever end; this project deliverable were extended two months.

References

1. Daconta, Michael C. and Bullard, Vaughn and Smith, Kevin T. *Essential XUL Programming*. New York: John Wiley & Sons, Inc., 2001.
2. B'Far, Reza and Richards, Roger and Ditlinger, Stephen. *Designing Effective User Interfaces For Wireless Devices*. DevX.
3. VoiceXML Reference. 2000. Systinet.
<http://www.zvon.org/xxl/VoiceXMLReference/Output/>.
4. Abrams, Marc., Phanouriou, C., Batongbacal A. L., Williams, S. and Shuster, J. *UIML: An Appliance-Independent XML User Interface Language*,
<http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html>.

Appendix A – Swing Rule Specification

```
<?XML version="1.0" encoding="UTF-8"?>
<rules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Profile>
    <Import name="Import">
      <Statement>javax.swing.*</Statement>
      <Statement>javax.swing.event.*</Statement>
      <Statement>javax.swing.border.*</Statement>
      <Statement>java.awt.*</Statement>
      <Statement>java.awt.event.*</Statement>
      <Statement>javax.swing.tree.*</Statement>
    </Import>
    <DisplayMethod>show</DisplayMethod>
    <Toolkit>SWING</Toolkit>
    <Accessor>
      <Method name="getComponent" returnType="Object">
        <Parameter type="String">name</Parameter>
      </Method>
    </Accessor>
  </Profile>
  <Event>
    <Action>
      <Field name="actionListener" method="addActionListener">
        <Type>ActionEventHandler</Type>
      </Field>
    </Action>
    <Focus>
      <Field name="focusListener" method="addFocusListener">
        <Type>FocusEventHandler</Type>
      </Field>
    </Focus>
    <Window>
      <Field name="windowListener" method="addWindowListener">
        <Type>WindowEventHandler</Type>
      </Field>
    </Window>
    <Selection>
      <Field name="selectionListener" method="addListSelectionListener">
        <Type>ListSelectionEventHandler</Type>
      </Field>
    </Selection>
    <Mouse>
      <Field name="mouseListener" method="addMouseListener">
        <Type>MouseEventHandler</Type>
      </Field>
    </Mouse>
    <MouseMotion>
      <Field name="mouseMotionListener" method="addMouseMotionListener">
        <Type>MouseMotionEventHandler</Type>
      </Field>
    </MouseMotion>
  </Event>
  <Tag>
    <rule>
      <ToolItem>
        <widget>JButton</widget>
        <attribute>
          <Method name="name" type="String">setTitle</Method>
        </attribute>
      </ToolItem>
    </rule>
    <rule>
      <Screen>
        <!-- define the control mapping -->
        <widget>Class</widget>
        <!-- define the attribute mapping -->
        <attribute>

```

```

        <Method name="title" type="String">setTitle</Method>
        <Method name="name" type="String">setName</Method>
    </attribute>
    <!-- define the composition mapping -->
</Screen>
</rule>
<rule>
    <Window>
        <!-- define the control mapping -->
        <widget>JFrame</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="name" type="String">setTitle</Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition>
            <SplitPane>
                <Method name="getContentPane().add"/>
            </SplitPane>
            <Panel>
                <Method name="getContentPane().add"/>
            </Panel>
            <button>
                <Method name="getContentPane().add"/>
            </button>
            <FlowLayout>
                <Method name="setLayout"/>
            </FlowLayout>
            <TextBox type="text">
                <Method name="getContentPane().add"/>
            </TextBox>
            <BoxLayout>
                <Method name="setLayout"/>
            </BoxLayout>
            <MenuBar>
                <Method name="setJMenuBar"/>
            </MenuBar>
            <ToolBar>
                <Method name="getContentPane().add"/>
            </ToolBar>
            <TabbedPane>
                <Method name="getContentPane().add"/>
            </TabbedPane>
        </composition>
    </Window>
</rule>
<rule>
    <LineBorder>
        <widget>LineBorder
            <Constructor cast="explicit" method="BorderFactory.createLineBorder">
                <Parameter type="String" method="Color">$color</Parameter>
            </Constructor>
        </widget>
        <attribute>
            <Method name="color" type="String" variable="yes"/>
        </attribute>
    </LineBorder>
</rule>
<rule>
    <Group>
        <widget>ButtonGroup</widget>
        <composition>
            <RadioButton>
                <Method name="add"/>
            </RadioButton>
            <CheckBox>
                <Method name="add"/>
            </CheckBox>
        </composition>
    </Group>

```

```

</rule>
<rule>
  <RadioButton>
    <widget>JRadioButton</widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
  </RadioButton>
</rule>
<rule>
  <CheckBox>
    <widget>JCheckBox</widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
  </CheckBox>
</rule>
<rule>
  <Separator>
    <widget>JSeparator</widget>
  </Separator>
</rule>
<rule>
  <GridLayout>
    <!-- define the control mapping -->
    <widget>GridLayout </widget>
    <!-- define the attribute mapping -->
    <attribute>
      <Method name="rows" type="int">setRows</Method>
      <Method name="columns" type="int">setColumns</Method>
    </attribute>
  </GridLayout>
</rule>
<rule>
  <FlowLayout>
    <!-- define the control mapping -->
    <widget>FlowLayout </widget>
  </FlowLayout>
</rule>
<rule>
  <BoxLayout>
    <!-- define the control mapping -->
    <widget>BoxLayout
      <Constructor>
        <Parameter type="text">$Parent</Parameter>
        <Parameter type="int">$axis</Parameter>
      </Constructor>
    </widget>
    <!-- define the attribute mapping -->
    <attribute>
      <Method name="axis" type="int" variable="yes"/>
    </attribute>
  </BoxLayout>
</rule>
<rule>
  <Label>
    <!-- define the control mapping -->
    <!-- this will always define a data type i.e JLabel jLabel = new JLabel(); -->
    <widget>JLabel</widget>
    <!-- define the attribute mapping -->
    <attribute>
      <Method name="title" type="String">setTitle</Method>
      <Method name="text" type="String">setText</Method>
      <Method name="icon" type="text">setIcon</Method>
    </attribute>
    <!-- define the composition mapping -->
  </Label>
</rule>
<rule>
  <TextBox type="text">

```

```

        <!-- define the control mapping -->
        <widget>JTextField</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="background" type="Color">setBackground</Method>
            <Method name="columns" type="int">setColumns</Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition/>
    </TextBox>
</rule>
<rule>
    <ToolBar>
        <!-- define the control mapping -->
        <widget>JToolBar</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="background" type="Color">setBackground</Method>
            <Method name="columns" type="int">setColumns</Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition>
            <ToolBarItem>
                <Method name="add">
                    </Method>
            </ToolBarItem>
        </composition>
    </ToolBar>
</rule>
<rule>
    <TabbedPane>
        <!-- define the control mapping -->
        <widget>JTabbedPane</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="title" type="String">setTitle</Method>
            <Method name="name" type="String">setName</Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition>
            <Tabs>
                <Method name="addTab">
                    <Parameter rule="Tabs" type="String">$text</Parameter>
                    <Parameter>$Tabs</Parameter>
                </Method>
            </Tabs>
        </composition>
    </TabbedPane>
</rule>
<rule>
    <Tabs>
        <!-- define the control mapping -->
        <widget>JPanel</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="title" type="String">setTitle</Method>
            <Method name="name" type="String">setName</Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition>
            <GridLayout>
                <Method name="setLayout"/>
            </GridLayout>
            <Button>
                <Method name="add"/>
            </Button>
            <TextBox type="text">
                <Method name="add"/>
            </TextBox>
        </Panel>
    </Tabs>
</rule>

```

```

        <Method name="add"/>
    </Panel>
    <Slider>
        <Method name="add"/>
    </Slider>
    <Separator>
        <Method name="add"/>
    </Separator>
    <ProgressBar>
        <Method name="add"/>
    </ProgressBar>
    <BoxLayout>
        <Method name="setLayout"/>
    </BoxLayout>
    <ToolBar>
        <Method name="add"/>
    </ToolBar>
    <Label>
        <Method name="add"/>
    </Label>
    <CheckBox>
        <Method name="add"/>
    </CheckBox>
    <RadioButton>
        <Method name="add"/>
    </RadioButton>
    <ComboBox>
        <Method name="add"/>
    </ComboBox>
    <List>
        <Method name="add">
            <Parameter rule="List">$myList</Parameter>
        </Method>
    </List>
    <LineBorder>
        <Method name="setBorder"/>
    </LineBorder>
    <Image>
        <Method name="add"/>
    </Image>
    <Tree>
        <Method name="add">
            <Parameter rule="Tree">$myTree</Parameter>
        </Method>
    </Tree>
</composition>
</Tabs>
</rule>
<rule>
    <Button>
        <!-- define the control mapping -->
        <widget>JButton</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="title" type="String">setTitle</Method>
            <Method name="text" type="String">setText</Method>
        </attribute>
        <!-- define the composition mapping -->
    </Button>
</rule>
<rule>
    <Panel>
        <!-- define the control mapping -->
        <widget>JPanel</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="title" type="String">setTitle</Method>
            <Method name="name" type="String">setName</Method>
        </attribute>
        <!-- define the composition mapping -->
    </Panel>
</rule>

```

```

<composition>
  <GridLayout>
    <Method name="setLayout"/>
  </GridLayout>
  <Button>
    <Method name="add"/>
  </Button>
  <TextBox type="text">
    <Method name="add"/>
  </TextBox>
  <Panel>
    <Method name="add"/>
  </Panel>
  <Slider>
    <Method name="add"/>
  </Slider>
  <Separator>
    <Method name="add"/>
  </Separator>
  <ProgressBar>
    <Method name="add"/>
  </ProgressBar>
  <BoxLayout>
    <Method name="setLayout"/>
  </BoxLayout>
  <ToolBar>
    <Method name="add"/>
  </ToolBar>
  <Label>
    <Method name="add"/>
  </Label>
  <CheckBox>
    <Method name="add"/>
  </CheckBox>
  <RadioButton>
    <Method name="add"/>
  </RadioButton>
  <ComboBox>
    <Method name="add"/>
  </ComboBox>
  <List>
    <Method name="add">
      <Parameter rule="List">$myList</Parameter>
    </Method>
  </List>
  <LineBorder>
    <Method name="setBorder"/>
  </LineBorder>
  <Image>
    <Method name="add"/>
  </Image>
  <Tree>
    <Method name="add">
      <Parameter rule="Tree">$myTree</Parameter>
    </Method>
  </Tree>
</composition>
</Panel>
</rule>
<rule>
  <MenuBar>
    <!-- define the control mapping -->
    <widget>JMenuBar</widget>
    <!-- define the attribute mapping -->
    <attribute>
      <Method name="title" type="String">setTitle</Method>
      <Method name="name" type="String">setName</Method>
    </attribute>
    <!-- define the composition mapping -->
    <composition>

```

```

        <Menu>
            <Method name="add"/>
        </Menu>
    </composition>
</MenuBar>
</rule>
<rule>
    <MenuItem>
        <!-- define the control mapping -->
        <widget>JMenuItem</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="text" type="String">setText</Method>
            <Method name="name" type="String">setName</Method>
            <Method name="mneonomic" type="char">setMnemonic</Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition/>
    </MenuItem>
</rule>
<rule>
    <MenuItem type="checkbox">
        <!-- define the control mapping -->
        <widget>JCheckBoxMenuItem</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="text" type="String">setText</Method>
            <Method name="name" type="String">setName</Method>
            <Method name="mneonomic" type="char">setMnemonic</Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition/>
    </MenuItem>
</rule>
<rule>
    <MenuItem type="radiobutton">
        <!-- define the control mapping -->
        <widget>JRadioButtonMenuItem</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="text" type="String">setText</Method>
            <Method name="name" type="String">setName</Method>
            <Method name="mneonomic" type="char">setMnemonic</Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition/>
    </MenuItem>
</rule>
<rule>
    <Menu>
        <!-- define the control mapping -->
        <widget>JMenu</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="text" type="String">setText</Method>
            <Method name="name" type="String">setName</Method>
            <Method name="title" type="String">setTitle </Method>
        </attribute>
        <!-- define the composition mapping -->
        <composition>
            <MenuItem>
                <method name="add"/>
            </MenuItem>
            <MenuItem type="checkbox">
                <method name="add"/>
            </MenuItem>
            <MenuItem type="radiobutton">
                <method name="add"/>
            </MenuItem>
        </composition>
    </Menu>

```

```

        <method name="add"/>
    </Menu>
</composition>
</Menu>
</rule>
<rule>
    <List>
        <!-- define the control mapping -->
        <widget component="JList">DefaultListModel</widget>
        <!-- define the attribute mapping -->
        <attribute>
            <Method name="text" type="String">setText</Method>
            <Method name="title" type="String">setTitle</Method>
            <!-- very important we need to know how to add individual element to this component -->
        </attribute>
        <!-- define the composition mapping -->
        <define>
            <Method name="myList" order="post" type="JList">
                <Parameter>$List</Parameter>
            </Method>
        </define>
        <composition>
            <ListItem>
                <Method name="add">
                    <Parameter>$counter</Parameter>
                    <Parameter>$default</Parameter>
                </Method>
            </ListItem>
        </composition>
    </List>
</rule>
<rule>
    <ListItem>
        <widget create="false">DefaultListModel</widget>
    </ListItem>
</rule>
<rule>
    <ComboBox>
        <widget>JComboBox</widget>
        <attribute>
            <Method name="title" type="String">setTitle</Method>
            <Method name="text" type="String">setText</Method>
        </attribute>
        <Composition>
            <Item>
                <Method name="addItem" type="String"/>
            </Item>
        </Composition>
    </ComboBox>
</rule>
<rule>
    <Item>
        <widget create="false">JComboBox</widget>
    </Item>
</rule>
<rule>
    <Slider>
        <widget>JSlider</widget>
        <attribute>
            <Method name="maximum" type="int">setMaximum</Method>
            <Method name="minimum" type="int">setMinimum</Method>
        </attribute>
    </Slider>
</rule>
<rule>
    <ProgressBar>
        <widget>JProgressBar</widget>
        <attribute>
            <Method name="maximum" type="int">setMaximum</Method>
            <Method name="minimum" type="int">setMinimum</Method>
        </attribute>
    </ProgressBar>
</rule>

```

```

        <Method name="value" type="int">setValue</Method>
    </attribute>
</ProgressBar>
</rule>
<rule>
    <Image>
        <widget>JLabel
            <Constructor>
                <Parameter type="text">$image</Parameter>
            </Constructor>
        </widget>
        <define>
            <Method name="image" type="ImageIcon">
                <Parameter type="String">$imagesrc</Parameter>
            </Method>
        </define>
        <attribute>
            <Method name="imagesrc" type="String" variable="yes">setImage</Method>
        </attribute>
    </Image>
</rule>
<rule>
    <Tree>
        <widget>DefaultMutableTreeNode</widget>
        <attribute>
            <Method name="text" type="String">setText</Method>
        </attribute>
        <define>
            <Method name="myTree" order="post" type="JTree">
                <Parameter>$Tree</Parameter>
            </Method>
        </define>
        <composition>
            <Node>
                <Method name="add"/>
            </Node>
        </composition>
    </Tree>
</rule>
<rule>
    <Node>
        <widget>DefaultMutableTreeNode
            <Constructor>
                <Parameter type="String">$text</Parameter>
            </Constructor>
        </widget>
        <composition>
            <Node>
                <Method name="add"/>
            </Node>
        </composition>
    </Node>
</rule>
<rule>
    <SplitPane>
        <widget>JSplitPane</widget>
        <Composition>
            <Panel name="left">
                <Method name="setLeftComponent"/>
            </Panel>
            <Panel name="right">
                <Method name="setRightComponent"/>
            </Panel>
        </Composition>
    </SplitPane>
</rule>
</Tag>
</rules>

```

Appendix B – SWT Rule Specfication

```
<?XML version="1.0" encoding="UTF-8"?>
<rules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Profile>
    <Import name="Import">
      <Statement>org.eclipse.swt.*</Statement>
      <Statement>org.eclipse.swt.widgets.*</Statement>
      <Statement>org.eclipse.swt.layout.*</Statement>
      <Statement>org.eclipse.swt.graphics.*</Statement>
      <Statement>org.eclipse.swt.custom.*</Statement>
    </Import>
    <DisplayMethod>open</DisplayMethod>
    <Toolkit>SWT</Toolkit>
    <Accessor>
      <Method name="getComponent" returnType="Object">
        <Parameter type="String">name</Parameter>
      </Method>
    </Accessor>
  </Profile>
  <Tag>
    <rule>
      <Screen>
        <!-- define the control mapping -->
        <widget create="false">Class</widget>
        <!-- define the attribute mapping -->
        <attribute>
          <Method name="text" type="String">setText</Method>
          <Method name="name" type="String">setName</Method>
        </attribute>
      </Screen>
    </rule>
    <rule>
      <Window>
        <!-- define the control mapping -->
        <Widget>Shell
          <Constructor>
            <Parameter type="Display">$display</Parameter>
          </Constructor>
        </Widget>
        <!-- define the attribute mapping -->
        <attribute>
          <Method name="name" type="String">setText</Method>
          <Method name="addLayout" set="true">setLayout
            <Parameter type="FillLayout">$Fill</Parameter>
          </Method>
        </attribute>
        <define>
          <Method name="Fill" type="FillLayout"/>
          <Method name="display" type="Display"/>
        </define>
        <!-- define the composition mapping -->
        <Composition>
          <Panel>
            <Constructor>
              <Parameter>$Window</Parameter>
              <Parameter>#SWT.FLAT</Parameter>
            </Constructor>
          </Panel>
          <MenuBar>
            <method name="setMenuBar"/>
          </MenuBar>
          <SplitPane>
            <Constructor>
              <Parameter>$Window</Parameter>
              <Parameter>#SWT.HORIZONTAL | SWT.BORDER </Parameter>
            </Constructor>
          </SplitPane>
        </Composition>
      </Window>
    </rule>
  </Tag>
</rules>
```

```

</Window>
</rule>
<rule>
  <Label>
    <!-- define the control mapping -->
    <!-- tell the ruleEngine to DO NOT DEFINE THIS COMPONENT -->
    <widget create="false">Label</widget>
    <!-- define the attribute mapping -->
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
  </Label>
</rule>
<rule>
  <BoxLayout>
    <!-- define the control mapping -->
    <widget>RowLayout
      <Constructor>
        <Parameter>#SWT.VERTICAL</Parameter>
      </Constructor>
    </widget>
    <!-- define the attribute mapping -->
  </BoxLayout>
</rule>
<rule>
  <FlowLayout>
    <!-- define the control mapping -->
    <widget>FillLayout</widget>
    <!-- define the attribute mapping -->
  </FlowLayout>
</rule>
<rule>
  <GridLayout>
    <widget>GridLayout </widget>
    <!-- define the attribute mapping -->
    <attribute>
      <Method name="icolumns" type="int" assignment="true">numColumns</Method>
      <Method name="horizontalSpacing" type="int" assignment="true">horizontalSpacing</Method>
    </attribute>
  </GridLayout>
</rule>
<rule>
  <Panel>
    <widget create="false">Composite</widget>
    <composition>
      <Label>
        <Constructor>
          <Parameter>$Panel</Parameter>
          <Parameter>#SWT.FLAT</Parameter>
        </Constructor>
      </Label>
      <Panel>
        <Constructor>
          <Parameter>$Parent</Parameter>
          <Parameter>#SWT.FLAT</Parameter>
        </Constructor>
      </Panel>
      <GridLayout>
        <method name="setLayout"/>
      </GridLayout>
      <FlowLayout>
        <method name="setLayout"/>
      </FlowLayout>
      <BoxLayout>
        <method name="setLayout"/>
      </BoxLayout>
      <TextBox type="text">
        <Constructor>
          <Parameter>$Panel</Parameter>
          <Parameter>#SWT.BORDER </Parameter>
        </Constructor>
      </TextBox>
    </composition>
  </Panel>
</rule>

```

```
        </Constructor>
    </TextBox>
    <Button>
        <Constructor>
            <Parameter>$Panel</Parameter>
            <Parameter>#SWT.PUSH</Parameter>
        </Constructor>
    </Button>
    <Group>
        <Constructor>
            <Parameter>$Panel</Parameter>
            <Parameter>#SWT.BORDER</Parameter>
        </Constructor>
    </Group>
    <CheckBox>
        <Constructor>
            <Parameter>$Panel</Parameter>
            <Parameter>#SWT.CHECK</Parameter>
        </Constructor>
    </CheckBox>
    <List>
        <Constructor>
            <Parameter>$Panel</Parameter>
            <Parameter>#SWT.MULTI | SWT.V_SCROLL | SWT.BORDER</Parameter>
        </Constructor>
    </List>
    <ComboBox>
        <Constructor>
            <parameter>$Parent</parameter>
            <parameter>#SWT.READ_ONLY</parameter>
        </Constructor>
    </ComboBox>
    <Slider>
        <Constructor>
            <parameter>$Panel</parameter>
            <parameter>#SWT.HORIZONTAL</parameter>
        </Constructor>
    </Slider>
    <Scale>
        <Constructor>
            <parameter>$Panel</parameter>
            <parameter>#SWT.HORIZONTAL</parameter>
        </Constructor>
    </Scale>
    <ProgressBar>
        <Constructor>
            <parameter>$Panel</parameter>
            <parameter>#SWT.HORIZONTAL</parameter>
        </Constructor>
    </ProgressBar>
    <Image>
        <Constructor>
            <parameter>$Panel</parameter>
            <parameter>#SWT.FLAT</parameter>
        </Constructor>
    </Image>
    <Separator>
        <Constructor>
            <parameter>$Panel</parameter>
            <parameter>#SWT.SEPARATOR | SWT.HORIZONTAL</parameter>
        </Constructor>
    </Separator>
    <Tree>
        <Constructor>
            <parameter>$Panel</parameter>
            <parameter>#SWT.MULTI | SWT.BORDER</parameter>
        </Constructor>
    </Tree>
</composition>
</Panel>
```

```

</rule>
<rule>
  <MenuBar>
    <widget>Menu
      <Constructor>
        <Parameter>$Parent</Parameter>
        <Parameter>#SWT.BAR</Parameter>
      </Constructor>
    </widget>
    <composition>
      <Menu>
        <Constructor>
          <Parameter>$MenuBar</Parameter>
          <Parameter>#SWT.CASCADE</Parameter>
        </Constructor>
      </Menu>
    </composition>
  </MenuBar>
</rule>
<rule>
  <Menu>
    <widget create="false">MenuItem</widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
      <Method name="addMenu" set="true">setMenu
        <Parameter type="Menu">$MenuObject</Parameter>
      </Method>
    </attribute>
    <define>
      <Method name="MenuObject" order="post" type="Menu">
        <Parameter>$TopContainer</Parameter>
        <Parameter>#SWT.DROP_DOWN</Parameter>
      </Method>
    </define>
    <composition>
      <MenuItem>
        <Constructor>
          <Parameter>$Parent</Parameter>
          <Parameter>#SWT.PUSH</Parameter>
        </Constructor>
      </MenuItem>
      <Menu>
        <Constructor>
          <Parameter>$Parent</Parameter>
          <Parameter>#SWT.CASCADE</Parameter>
        </Constructor>
      </Menu>
    </composition>
  </Menu>
</rule>
<rule>
  <MenuItem>
    <widget create="false"> MenuItem </widget>
    <!-- define the attribute mapping -->
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
    <!-- define the composition mapping -->
  </MenuItem>
</rule>
<rule>
  <TextBox type="text">
    <widget create="false">Text</widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
      <Method name="bounds">setBounds
        <Parameter type="int">$left</Parameter>
        <Parameter type="int">$lefty</Parameter>
        <Parameter type="int">$rightx</Parameter>
        <Parameter type="int">$righty</Parameter>
      </Method>
    </attribute>
  </TextBox>
</rule>

```

```

        </Method>
      </attribute>
    </TextBox>
  </rule>
</rule>
<rule>
  <Button>
    <widget create="false">Button</widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
  </Button>
</rule>
<rule>
  <Group>
    <widget create="false">Group</widget>
    <attribute>
      <Method name="addLayout" set="true">setLayout
        <Parameter type="RowLayout">$RowGroupLayout</Parameter>
      </Method>
      <Method name="text" type="String">setText</Method>
    </attribute>
    <define>
      <Method name="RowGroupLayout" type="RowLayout"/>
    </define>
    <composition>
      <checkbox>
        <Constructor>
          <parameter>$Group</parameter>
          <parameter>#SWT.CHECK</parameter>
        </Constructor>
      </checkbox>
      <radiobutton>
        <Constructor>
          <parameter>$Group</parameter>
          <parameter>#SWT.RADIO</parameter>
        </Constructor>
      </radiobutton>
    </composition>
  </Group>
</rule>
<rule>
  <CheckBox>
    <widget create="false">Button</widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
  </CheckBox>
</rule>
<rule>
  <RadioButton>
    <widget create="false">Button</widget>
    <attribute>
      <Method name="text" type="String">setText</Method>
    </attribute>
  </RadioButton>
</rule>
<rule>
  <List>
    <widget create="false">List</widget>
    <composition>
      <ListItem>
        <method name="add" type="String"/>
      </ListItem>
    </composition>
  </List>
</rule>
<rule>
  <ListItem>
    <widget create="false">List</widget>
  </ListItem>

```

```

</rule>
<rule>
  <ComboBox>
    <widget create="false">Combo</widget>
    <composition>
      <item>
        <Method name="add" type="String"/>
      </item>
    </composition>
  </ComboBox>
</rule>
<rule>
  <Item>
    <widget create="false">Combo</widget>
  </Item>
</rule>
<rule>
  <TabbedPane>
    <widget>TabFolder
      <Constructor>
        <Parameter>$TopContainer</Parameter>
        <Parameter>#SWT.NONE</Parameter>
      </Constructor>
    </widget>
    <composition>
      <Tabs>
        <Constructor>
          <Parameter>$TabbedPane</Parameter>
          <Parameter>#SWT.NONE</Parameter>
        </Constructor>
      </Tabs>
    </composition>
  </TabbedPane>
</rule>
<rule>
  <Tabs>
    <widget create="false">Composite</widget>
    <define>
      <Method name="TabItemName" type="TabItem">
        <Parameter rule="TabbedPane">$Parent</Parameter>
        <Parameter>#SWT.NONE</Parameter>
        <attribute>
          <Method name="ContainerText">setControl
            <Parameter type="Composite">$Tabs</Parameter>
          </Method>
          <Method name="text" type="String">setText</Method>
        </attribute>
      </Method>
    </define>
    <composition>
      <Label>
        <Constructor>
          <Parameter>$Tabs</Parameter>
          <Parameter>#SWT.FLAT</Parameter>
        </Constructor>
      </Label>
      <Panel>
        <Constructor>
          <Parameter>$TabbedPane</Parameter>
          <Parameter>#SWT.FLAT</Parameter>
        </Constructor>
      </Panel>
      <GridLayout>
        <method name="setLayout"/>
      </GridLayout>
      <FlowLayout>
        <method name="setLayout"/>
      </FlowLayout>
      <BoxLayout>
        <method name="setLayout"/>
      </BoxLayout>
    </composition>
  </Tabs>
</rule>

```

```

</BoxLayout>
<TextBox type="text">
  <Constructor>
    <Parameter>$Tabs</Parameter>
    <Parameter>#SWT.BORDER </Parameter>
  </Constructor>
</TextBox>
<Button>
  <Constructor>
    <Parameter>$Tabs</Parameter>
    <Parameter>#SWT.PUSH</Parameter>
  </Constructor>
</Button>
<Group>
  <Constructor>
    <Parameter>$Tabs</Parameter>
    <Parameter>#SWT.BORDER</Parameter>
  </Constructor>
</Group>
<CheckBox>
  <Constructor>
    <Parameter>$Tabs</Parameter>
    <Parameter>#SWT.CHECK</Parameter>
  </Constructor>
</CheckBox>
<List>
  <Constructor>
    <Parameter>$Tabs</Parameter>
    <Parameter>#SWT.MULTI | SWT.V_SCROLL | SWT.BORDER</Parameter>
  </Constructor>
</List>
<ComboBox>
  <Constructor>
    <parameter>$Tabs</parameter>
    <parameter>#SWT.READ_ONLY</parameter>
  </Constructor>
</ComboBox>
<Slider>
  <Constructor>
    <parameter>$Tabs</parameter>
    <parameter>#SWT.HORIZONTAL</parameter>
  </Constructor>
</Slider>
<Scale>
  <Constructor>
    <parameter>$Tabs</parameter>
    <parameter>#SWT.HORIZONTAL</parameter>
  </Constructor>
</Scale>
<ProgressBar>
  <Constructor>
    <parameter>$Tabs</parameter>
    <parameter>#SWT.HORIZONTAL</parameter>
  </Constructor>
</ProgressBar>
</composition>
</Tabs>
</rule>
<rule>
  <Slider>
    <widget create="false">Scale</widget>
    <attribute>
      <Method name="maximum" type="int">setMaximum</Method>
      <Method name="minimum" type="int">setMinimum</Method>
      <Method name="value" type="int">setSelection</Method>
    </attribute>
  </Slider>
</rule>
<rule>
  <Scale>

```

```

        <widget create="false">Slider</widget>
        <attribute>
            <Method name="maximum" type="int">setMaximum</Method>
            <Method name="minimum" type="int">setMinimum</Method>
            <Method name="value" type="int">setSelection</Method>
        </attribute>
    </Scale>
</rule>
<rule>
    <ProgressBar>
        <widget create="false">ProgressBar</widget>
        <attribute>
            <Method name="maximum" type="int">setMaximum</Method>
            <Method name="minimum" type="int">setMinimum</Method>
            <Method name="value" type="int">setSelection</Method>
        </attribute>
    </ProgressBar>
</rule>
<rule>
    <Image>
        <widget create="false">Button</widget>
        <define>
            <Method name="imageIcon" type="Image">
                <Parameter rule="Window">$display</Parameter>
                <Parameter type="String">$imageSrc</Parameter>
            </Method>
        </define>
        <attribute>
            <Method name="myImage" type="Button" set="true">setImage
                <Parameter>$imageIcon</Parameter>
            </Method>
        </attribute>
    </Image>
</rule>
<rule>
    <Separator>
        <widget create="false">Label</widget>
    </Separator>
</rule>
<rule>
    <Tree>
        <widget create="false">Tree</widget>
        <attribute>
            <Method name="text" type="String">setText</Method>
        </attribute>
        <composition>
            <Node>
                <Constructor>
                    <Parameter>$Parent</Parameter>
                    <Parameter>#SWT.NONE</Parameter>
                </Constructor>
            </Node>
        </composition>
    </Tree>
</rule>
<rule>
    <Node>
        <widget create="false">TreeItem</widget>
        <attribute>
            <Method name="text" type="String">setText</Method>
        </attribute>
        <composition>
            <Node>
                <Constructor>
                    <Parameter>$Node</Parameter>
                    <Parameter>#SWT.NONE</Parameter>
                </Constructor>
            </Node>
        </composition>
    </Node>

```

```
</rule>
<rule>
  <SplitPane>
    <widget create="false">SashForm</widget>
    <define>
      <Method name="fillPanelLayout" type="FillLayout"/>
    </define>
    <attribute>
      <Method name="addLayout" set="true">setLayout
        <Parameter type="FillLayout">$fillPanelLayout</Parameter>
      </Method>
    </attribute>
    <Composition>
      <Panel name="left">
        <Constructor>
          <Parameter>$Parent</Parameter>
          <Parameter>#SWT.FLAT</Parameter>
        </Constructor>
      </Panel>
      <Panel name="right">
        <Constructor>
          <Parameter>$Parent</Parameter>
          <Parameter>#SWT.FLAT</Parameter>
        </Constructor>
      </Panel>
    </Composition>
  </SplitPane>
</rule>
</Tag>
</rules>
```

Appendix C – Property File

MDML Filename and Location

mdml = j:/eclipse/workspace/zoom/test/ui/CalculatorGUI.XML

Rule XML Table

ruleXML = rule.XML

Project Home Directory

project_directory = j:/eclipse/workspace/zoom/

Code generator

project_codegenerator = j:/eclipse/workspace/zoom/lib/codegen.jar

Project Resources

project_resources_dir = j:/eclipse/workspace/zoom/src/resources/ui/

Project Output Directory

project_output_dir = j:/eclipse/workspace/zoom/test/ui/

#end

Appendix D – Multiple Device Markup Language

GUI Component	Description & XML Format
---------------	--------------------------

MDML Components	Common Properties For Multiple Device Markup Language																																																
Label	<p>Label displays a short text string and an image. Its content is may be centered both vertically and horizontally</p> <p>Example: <code><label text="Label" icon="image.gif" /></code></p> <p>Attributes:</p> <table border="1" data-bbox="565 478 1572 674"> <tr><td>vertical-text-position</td><td>top center bottom</td></tr> <tr><td>vertical-alignment</td><td>top center bottom</td></tr> <tr><td>text</td><td>The display text</td></tr> <tr><td>icon</td><td>image displayed</td></tr> <tr><td>horizontal-alignment</td><td>left center right leading trailing</td></tr> <tr><td>displayed-mnemonic</td><td>mnemonic</td></tr> <tr><td>horizontal-text-position</td><td>left center right leading trailing</td></tr> </table> <p>Component Attributes:</p> <table border="1" data-bbox="565 722 1572 1209"> <tr><td>id</td><td><<generated>></td></tr> <tr><td>name</td><td>name of the component</td></tr> <tr><td>background</td><td>Background of the component</td></tr> <tr><td>foreground</td><td>foreground of the component</td></tr> <tr><td>enabled</td><td>true false</td></tr> <tr><td>visible</td><td>true false</td></tr> <tr><td>font</td><td>font</td></tr> <tr><td>font-name</td><td>font name</td></tr> <tr><td>font-style</td><td>font style</td></tr> <tr><td>tool-tip-text</td><td>tool tip</td></tr> <tr><td>preferred-size</td><td>preferred size of component</td></tr> <tr><td>minimum-size</td><td>minimum size of component</td></tr> <tr><td>maximum-size</td><td>maximum size of component</td></tr> <tr><td>Opaque</td><td>true false</td></tr> <tr><td>alignment-x</td><td>left right center</td></tr> <tr><td>alignment-y</td><td>center top bottom</td></tr> <tr><td>Border</td><td>LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder</td></tr> </table>	vertical-text-position	top center bottom	vertical-alignment	top center bottom	text	The display text	icon	image displayed	horizontal-alignment	left center right leading trailing	displayed-mnemonic	mnemonic	horizontal-text-position	left center right leading trailing	id	<<generated>>	name	name of the component	background	Background of the component	foreground	foreground of the component	enabled	true false	visible	true false	font	font	font-name	font name	font-style	font style	tool-tip-text	tool tip	preferred-size	preferred size of component	minimum-size	minimum size of component	maximum-size	maximum size of component	Opaque	true false	alignment-x	left right center	alignment-y	center top bottom	Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder
vertical-text-position	top center bottom																																																
vertical-alignment	top center bottom																																																
text	The display text																																																
icon	image displayed																																																
horizontal-alignment	left center right leading trailing																																																
displayed-mnemonic	mnemonic																																																
horizontal-text-position	left center right leading trailing																																																
id	<<generated>>																																																
name	name of the component																																																
background	Background of the component																																																
foreground	foreground of the component																																																
enabled	true false																																																
visible	true false																																																
font	font																																																
font-name	font name																																																
font-style	font style																																																
tool-tip-text	tool tip																																																
preferred-size	preferred size of component																																																
minimum-size	minimum size of component																																																
maximum-size	maximum size of component																																																
Opaque	true false																																																
alignment-x	left right center																																																
alignment-y	center top bottom																																																
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder																																																

Button

button displays a short text string and an image. Its content is on the left and centered vertically

```
<button type="button"
        text="Button"
        icon="image.gif" horizontal-alignment = "left" tool-tip-text = "ToolTip" />
```

A two-state button

```
<button type="toggle" text="ToggleButton" icon="image.gif" selected="true"/>
```

Attributes:

type	button toggle
event	

Button Attributes:

text	The display text on the button
action-command	
disabled-icon	
icon	image displayed
disabled-selected-icon	
pressed-icon	
rollover-icon	left center right leading trailing
rollover-selected-icon	
selected-icon	
mnemonic	
rollover-enabled	true false
selected	true false
horizontal-alignment	left center right leading trailing
horizontal-text-position	left center right leading trailing
vertical-alignment	center top bottom
vertical-text-position	center top bottom
focus-painted	

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

CheckBox

Can be a checkbox or a grouped radio buttons

```
<checkbox text="CheckBox" icon="image.gif" selected="true"/>
```

```
<checkbox text="Option 1" group=" CheckButtonGroup "/>
```

Attributes:

CheckBox Attributes:

text	The display text on the button
action-command	
disabled-icon	
icon	image displayed
disabled-selected-icon	
pressed-icon	
rollover-icon	left center right leading trailing
rollover-selected-icon	
selected-icon	
mnemonic	
rollover-enabled	true false
selected	true false
horizontal-alignment	left center right leading trailing
horizontal-text-position	left center right leading trailing
vertical-alignment	center top bottom
vertical-text-position	center top bottom
focus-painted	

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Group

Specify a mutually exclusive group of CheckBox or RadioButton items.

```
<group type="CheckBox" name="CheckButtonGroup"/>
```

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

ComboBox

A choice of a combobox or radiobutton.

```
<choice type="combobox">  
  <item element="Choice" icon="image.gif" />  
  <item element="Disabled" enabled="false" />  
</choice>
```

Choice Attributes:

name	
------	--

Item Attributes:

element	
---------	--

Button Attributes:

text	The display text on the button
action-command	
disabled-icon	
icon	image displayed
disabled-selected-icon	
pressed-icon	
rollover-icon	left center right leading trailing
rollover-selected-icon	
selected-icon	
mnemonic	
rollover-enabled	true false
selected	true false
horizontal-alignment	left center right leading trailing
horizontal-text-position	left center right leading trailing
vertical-alignment	center top bottom
vertical-text-position	center top bottom
focus-painted	

ComboBox Attributes:

selected-item	
selected-index	
renderer	
maximum-row-count	
editor	true false
editable	true false
action-command	

TextBox

An editable single/multi/password text

```
<textbox type="text" text="TextField" />
```

A hidden text field characters

```
<textbox type="password" text="secret value" />
```

Multi-line text with scroll handling

```
<textbox type="multi" text="TextArea" />
```

TextBox Attributes:

Type	text password multi
horizontal-alignment	left center right
scroll-offset	
editable	true false
text	

Multi Attributes:

wrap-style-word	true false
tab-size	
rows	
line-wrap	true false
columns	

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

TabbedPane

Combined collection of components tabbed together.

```
<TabbedPane>
  <Tabs name="Label/TextBox">
    <GridLayout rows="3" columns="2" icolumns="6"/>
    <Label text="First Name"/>
    <TextBox type="text" id="txt1" text="firstName"/>
    <Label text="Last Name"/>
    <TextBox type="text" id="txt1" text="lastName"/>
    <Label text="Address "/>
    <TextBox type="text" id="txt1" text="addr1"/>
    <Label text="City"/>
    <TextBox type="text" id="txt1" text="city"/>
    <Label text="State"/>
    <TextBox type="text" id="txt1" text="state"/>
    <Label text="Zip"/>
    <TextBox type="text" id="txt1" text="zipcode"/>
  </Tabs>
  <Tabs name="ComboBox">
    <Label text="Combo"/>
    <Choice type="combobox">
      <Item element="24"/>
      <Item element="36"/>
    </Choice>
  </Tabs>
</TabbedPane>
```

TabbedPane Attributes:

tab-placement	top bottom left right
---------------	-----------------------

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Tabs Attributes:

mnemonic	
tool-tip	
title	

Tabs Component Attributes:

text	The display text on the button
action-command	
disabled-icon	
icon	image displayed
disabled-selected-icon	
pressed-icon	
rollover-icon	left center right leading trailing
rollover-selected-icon	
selected-icon	
mnemonic	
rollover-enabled	true false
selected	true false
horizontal-alignment	left center right leading trailing
horizontal-text-position	left center right leading trailing
vertical-alignment	center top bottom
vertical-text-position	center top bottom

Panel

Another container that holds components.

```
<Panel>
  <GridLayout rows="3" columns="2" icolumns="4"/>
  <Label text="First Name"/>
  <TextBox type="text" text="firstName"/>
  <Label text="Last Name"/>
  <TextBox type="text" text="lastName"/>
  <Label text="Address "/>
  <TextBox type="text" text="addr1"/>
  <Label text="City"/>
  <TextBox type="text" text="city"/>
  <Label text="State"/>
  <TextBox type="text" text="state"/>
  <Label text="Zip"/>
  <TextBox type="text" text="zipcode"/>
</Panel>
```

Panel Attributes:

scrollable	
hsb-policy	as-needed never always
vsb-policy	as-needed never always

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Start

A root pane that can contain other containers.

```
<Start name=" SampleApplication" XMLNs:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="C:\eclipse\workspace\Zoom\src\resources\ui\UISchema.xs
d">
</Start>
```

Window

A Container that contains other containers, components, etc

```
<Window title="Sample Application" visible="true" show="true">
  <Panel>
    <GridLayout rows="3" columns="2" icolumns="4"/>
    <Label text="First Name"/>
    <TextBox type="text" text="firstName"/>
    <Label text="Last Name"/>
    <TextBox type="text" text="lastName"/>
    <Label text="Address "/>
    <TextBox type="text" text="addr1"/>
    <Label text="City"/>
    <TextBox type="text" text="city"/>
    <Label text="State"/>
    <TextBox type="text" text="state"/>
    <Label text="Zip"/>
    <TextBox type="text" text="zipcode"/>
  </Panel>
</Window>
```

Window Attributes:

title	
location	
bounds	
dimension	
resizeable	true false
show	true false

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Dialog

A Dialog Container which holds containers, components, etc.

```
<Dialog name="DialogGUI" show="true">
  <MenuBar>
    <Menu text="File">
      <MenuItem text="Open"/>
    </Menu>
  </MenuBar>
  <Panel>
    <BoxLayout axis="1"/>
    <TextBox type="text" editable="false" columns="25"/>
  </Panel>
</Dialog>
```

Window Attributes:

title	
location	
bounds	
dimension	
resizeable	true false
show	true false
modal	true false

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

ProgressBar

Graphically displays an integer number with Min, Max and interval values

```
<progressbar minimum="25" maximum="75" value="50" orientation="vertical" />
```

ProgressBar Attributes:

value	
string-painted	true false
string	
maximum	
minimum	
orientation	horizontal vertical
change-listener	

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Slider

Enables the user to select an integer value within a range and interval
<slider minimum="25" maximum="75" value="50" orientation="vertical" />

Slider Attributes:

orientation	horizontal vertical
minor-tick-spacing	
major-tick-spacing	
paint-labels	true false
paint-ticks	true false
paint-track	true false
snap-to-ticks	true false
minimum	
maximum	
value	
standard-labels	

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

List

Enables the user to select one or more items from the list with internal scroll handling

```
<list selection="multiple">
  <item element="List" selected="true" />
  <item element="Item"/>
  <item element="Disabled" enabled="false" />
</list>
```

List Attributes:

model	
visible-row-count	
selection-model	
selection-mode	
selection-foreground	
selection-background	
fixed-cell-width	
fixed-cell-height	
cell-renderer	

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Item Attributes:

element	
---------	--

Button Attributes:

text	The display text on the button
action-command	
disabled-icon	
icon	image displayed
disabled-selected-icon	
pressed-icon	
rollover-icon	left center right leading trailing
rollover-selected-icon	
selected-icon	
mnemonic	
rollover-enabled	true false
selected	true false
horizontal-alignment	left center right leading trailing
horizontal-text-position	left center right leading trailing
vertical-alignment	center top bottom
vertical-text-position	center top bottom
focus-painted	

<p>Separator</p>	<p>A horizontal or vertical separator bar <code><separator /></code></p> <p>Separator Attributes:</p> <table border="1"> <tr> <td>orientation</td> <td>horizontal vertical</td> </tr> </table> <p>Component Attributes:</p> <table border="1"> <tr><td>id</td><td><<generated>></td></tr> <tr><td>name</td><td>name of the component</td></tr> <tr><td>background</td><td>Background of the component</td></tr> <tr><td>foreground</td><td>foreground of the component</td></tr> <tr><td>enabled</td><td>true false</td></tr> <tr><td>visible</td><td>true false</td></tr> <tr><td>font</td><td>font</td></tr> <tr><td>font-name</td><td>font name</td></tr> <tr><td>font-style</td><td>font style</td></tr> <tr><td>tool-tip-text</td><td>tool tip</td></tr> <tr><td>preferred-size</td><td>preferred size of component</td></tr> <tr><td>minimum-size</td><td>minimum size of component</td></tr> <tr><td>maximum-size</td><td>maximum size of component</td></tr> <tr><td>Opaque</td><td>true false</td></tr> <tr><td>alignment-x</td><td>left right center</td></tr> <tr><td>alignment-y</td><td>center top bottom</td></tr> <tr><td>Border</td><td>LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder</td></tr> </table>	orientation	horizontal vertical	id	<<generated>>	name	name of the component	background	Background of the component	foreground	foreground of the component	enabled	true false	visible	true false	font	font	font-name	font name	font-style	font style	tool-tip-text	tool tip	preferred-size	preferred size of component	minimum-size	minimum size of component	maximum-size	maximum size of component	Opaque	true false	alignment-x	left right center	alignment-y	center top bottom	Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder
orientation	horizontal vertical																																				
id	<<generated>>																																				
name	name of the component																																				
background	Background of the component																																				
foreground	foreground of the component																																				
enabled	true false																																				
visible	true false																																				
font	font																																				
font-name	font name																																				
font-style	font style																																				
tool-tip-text	tool tip																																				
preferred-size	preferred size of component																																				
minimum-size	minimum size of component																																				
maximum-size	maximum size of component																																				
Opaque	true false																																				
alignment-x	left right center																																				
alignment-y	center top bottom																																				
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder																																				
<p>FlowLayout</p>	<p>A layout which places components from left to right filling the area.</p> <p><code><flowlayout></code></p> <p>FlowLayout Attributes: <<none>></p>																																				
<p>BoxLayout</p>	<p><code><boxlayout axis = "1"/></code></p> <p>BoxLayout Attributes:</p> <table border="1"> <tr> <td>Axis</td> <td>The orientation: 1 = vertical, 2=horizontal</td> </tr> </table>	Axis	The orientation: 1 = vertical, 2=horizontal																																		
Axis	The orientation: 1 = vertical, 2=horizontal																																				
<p>GridLayout</p>	<p>A Grid Layout with places components inside a grid</p> <p><code><gridlayout rows="3" columns = "2" icolumns = "4"/></code></p> <p>GridLayout Attributes:</p> <table border="1"> <tr> <td>rows</td> <td>the numbers of rows in the grid</td> </tr> <tr> <td>columns</td> <td>the number of columns in the grid</td> </tr> <tr> <td>iColumns</td> <td>the number of columns with one widget per cell (Required for SWT)</td> </tr> </table>	rows	the numbers of rows in the grid	columns	the number of columns in the grid	iColumns	the number of columns with one widget per cell (Required for SWT)																														
rows	the numbers of rows in the grid																																				
columns	the number of columns in the grid																																				
iColumns	the number of columns with one widget per cell (Required for SWT)																																				

MenuBar

Just like a main menu in windows application with ability to have different components as a menu items like checkboxes or radio buttons

```
<menubar name="menubar1">
  <menu text="Menu">
    <menuItem text="MenuItem" icon="image.gif" />
    <menu text="Menu">
      <menuItem type="checkbox" text="CheckBox" selected="true" />
      <separator />
      <menuItem type="radiobutton" text="RadioButton"/>
    </menu>
  </menu>
</menubar>
```

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Menu

Menu have different heading; usually appear at the top of a Windows Application. Menu can also be used as a submenu too.

```
<menu text="Menu">
  <menuItem text="MenuItem" icon="image.gif" />
  <menu text="Menu">
    <menuItem type="checkbox" text="CheckBox" selected="true" />
    <separator />
    <menuItem type="radiobutton" text="RadioButton"/>
  </menu>
</menu>
```

Menu Attributes:

accelerator	
-------------	--

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Button Attributes:

text	The display text on the button
action-command	
disabled-icon	
icon	image displayed
disabled-selected-icon	
pressed-icon	
rollover-icon	left center right leading trailing
rollover-selected-icon	
selected-icon	
mnemonic	
rollover-enabled	true false
selected	true false
horizontal-alignment	left center right leading trailing
horizontal-text-position	left center right leading trailing
vertical-alignment	center top bottom
vertical-text-position	center top bottom
focus-painted	

MenuItem

Menu Items like checkboxes or radio buttons or normal .

```
<menuItem text="MenuItem" icon="image.gif" />
```

```
<menuItem type="checkbox" text="CheckBox" selected="true" />
```

MenuItem Attributes:

accelerator	
type	checkbox radiobutton normal
state	
event	
separator	true false

Component Attributes:

id	<<generated>>
name	name of the component
background	Background of the component
foreground	foreground of the component
enabled	true false
visible	true false
font	font
font-name	font name
font-style	font style
tool-tip-text	tool tip
preferred-size	preferred size of component
minimum-size	minimum size of component
maximum-size	maximum size of component
Opaque	true false
alignment-x	left right center
alignment-y	center top bottom
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder

Button Attributes:

text	The display text on the button
action-command	
disabled-icon	
icon	image displayed
disabled-selected-icon	
pressed-icon	
rollover-icon	left center right leading trailing
rollover-selected-icon	
selected-icon	
mnemonic	
rollover-enabled	true false
selected	true false
horizontal-alignment	left center right leading trailing
horizontal-text-position	left center right leading trailing
vertical-alignment	center top bottom
vertical-text-position	center top bottom
focus-painted	

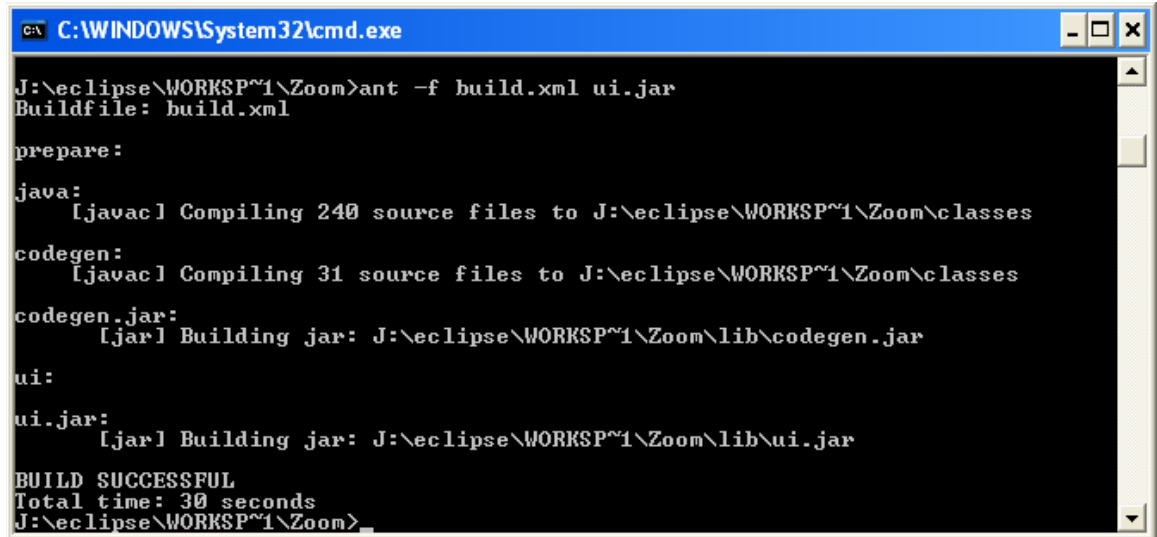
ToolBar	<p>ToolBar provides quick access to menu items</p> <pre><ToolBar name="tool bar"> <Button name="New"/> <Button name="Open"/> <Button name="Simple"/> <Button name="Complex"/> <Button name="Help"/> </ToolBar></pre> <p>ToolBar Attributes:</p> <table border="1"> <tr> <td>floatable</td> <td>true false</td> </tr> <tr> <td>orientation</td> <td>horizontal vertical</td> </tr> </table> <p>Component Attributes:</p> <table border="1"> <tr> <td>id</td> <td>< <generated> ></td> </tr> <tr> <td>name</td> <td>name of the component</td> </tr> <tr> <td>background</td> <td>Background of the component</td> </tr> <tr> <td>foreground</td> <td>foreground of the component</td> </tr> <tr> <td>enabled</td> <td>true false</td> </tr> <tr> <td>visible</td> <td>true false</td> </tr> <tr> <td>font</td> <td>font</td> </tr> <tr> <td>font-name</td> <td>font name</td> </tr> <tr> <td>font-style</td> <td>font style</td> </tr> <tr> <td>tool-tip-text</td> <td>tool tip</td> </tr> <tr> <td>preferred-size</td> <td>preferred size of component</td> </tr> <tr> <td>minimum-size</td> <td>minimum size of component</td> </tr> <tr> <td>maximum-size</td> <td>maximum size of component</td> </tr> <tr> <td>Opaque</td> <td>true false</td> </tr> <tr> <td>alignment-x</td> <td>left right center</td> </tr> <tr> <td>alignment-y</td> <td>center top bottom</td> </tr> <tr> <td>Border</td> <td>LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder</td> </tr> </table>	floatable	true false	orientation	horizontal vertical	id	< <generated> >	name	name of the component	background	Background of the component	foreground	foreground of the component	enabled	true false	visible	true false	font	font	font-name	font name	font-style	font style	tool-tip-text	tool tip	preferred-size	preferred size of component	minimum-size	minimum size of component	maximum-size	maximum size of component	Opaque	true false	alignment-x	left right center	alignment-y	center top bottom	Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder
floatable	true false																																						
orientation	horizontal vertical																																						
id	< <generated> >																																						
name	name of the component																																						
background	Background of the component																																						
foreground	foreground of the component																																						
enabled	true false																																						
visible	true false																																						
font	font																																						
font-name	font name																																						
font-style	font style																																						
tool-tip-text	tool tip																																						
preferred-size	preferred size of component																																						
minimum-size	minimum size of component																																						
maximum-size	maximum size of component																																						
Opaque	true false																																						
alignment-x	left right center																																						
alignment-y	center top bottom																																						
Border	LineBorder BevelBorder EtchedBorder EmptyBorder MatteBorder TitledBorder CompoundBorder																																						
Tree	<p>Tree like representation of the data with internal scroll handling</p> <pre><tree selection="multiple"> <node text="Node A" icon="image.gif"> <node text="Node B" icon="image.gif" selected="true" /> <node text="Node C" icon="image.gif" /> </node> <node text="Node D" expanded="false"> <node text="Node E" icon="image.gif" /> </node> </tree></pre>																																						
SplitPane	<p>To divide two components</p> <pre><splitpane orientation="vertical" divider="24"right="right" left="left"> <panel name="left"> <textbox text="Top"/> <textbox text="Bottom" /> </panel> <panel name="right"> <textbox text="Top"/> <textbox text="Bottom" /> </panel> </splitpane></pre>																																						

Table	Two dimensional table with internal scroll handling and ability to select rows an headers. <pre><table selection="multiple"> <column text="A" width="24" /><column text="B" /> <row selected="true"> <cell text="a1" /><cell text="b1" enabled="false" /> </row> <row> <cell text="a2" /><cell text="b2" /> </row> </table></pre>
--------------	--

Appendix D – Walkthrough

1. This project is part of the ZOOM project in CVS. Download The ZOOM Project either through CVS or eclipse.
2. Build the UI Project in the \$ZOOM_HOME directory

```
ant -f build.XML ui.jar
```



```
C:\WINDOWS\System32\cmd.exe

J:\eclipse\WORKSP~1\Zoom>ant -f build.xml ui.jar
Buildfile: build.xml

prepare:

java:
[javac] Compiling 240 source files to J:\eclipse\WORKSP~1\Zoom\classes

codegen:
[javac] Compiling 31 source files to J:\eclipse\WORKSP~1\Zoom\classes

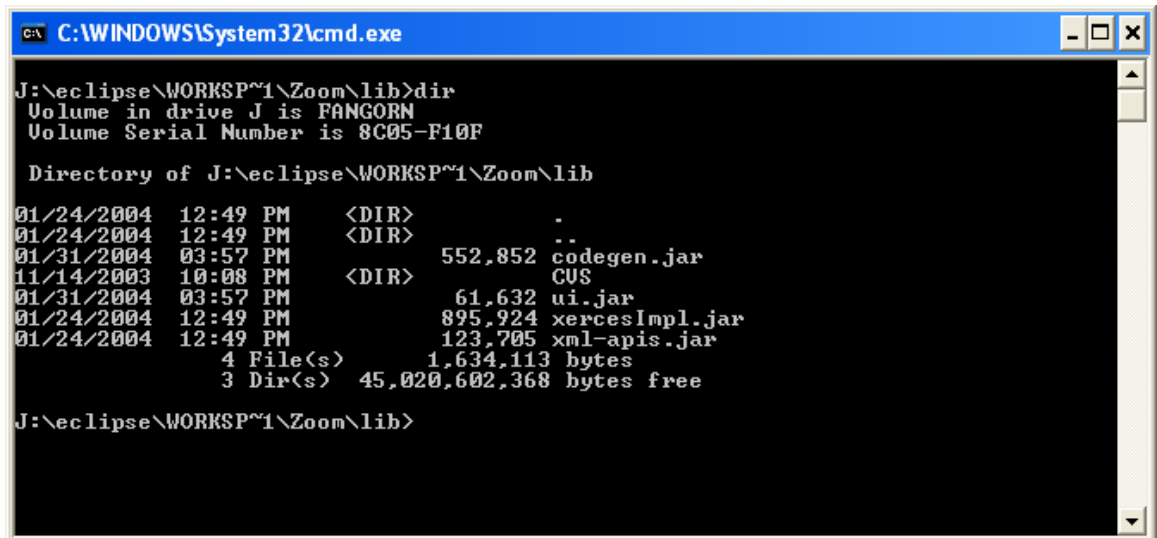
codegen.jar:
[jar] Building jar: J:\eclipse\WORKSP~1\Zoom\lib\codegen.jar

ui:

ui.jar:
[jar] Building jar: J:\eclipse\WORKSP~1\Zoom\lib\ui.jar

BUILD SUCCESSFUL
Total time: 30 seconds
J:\eclipse\WORKSP~1\Zoom>
```

Check the \$ZOOM_HOME/lib directory for these files.



```
C:\WINDOWS\System32\cmd.exe

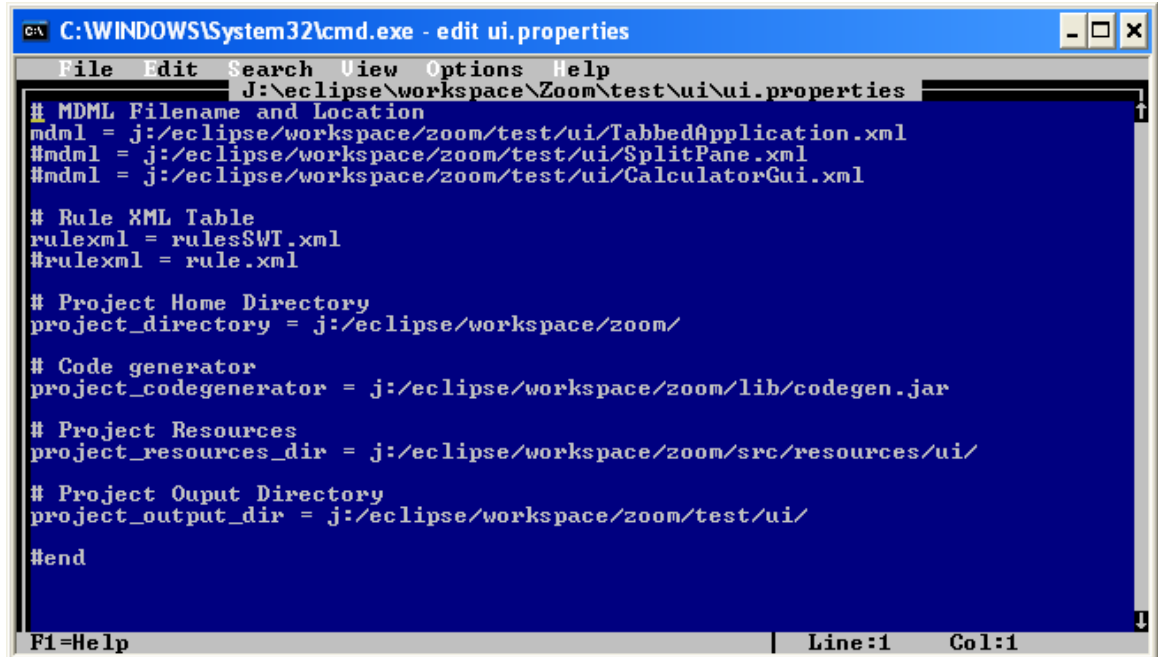
J:\eclipse\WORKSP~1\Zoom\lib>dir
Volume in drive J is FANGORN
Volume Serial Number is 8C05-F10F

Directory of J:\eclipse\WORKSP~1\Zoom\lib

01/24/2004  12:49 PM    <DIR>          .
01/24/2004  12:49 PM    <DIR>          ..
01/31/2004  03:57 PM             552,852  codegen.jar
11/14/2003  10:08 PM    <DIR>          CUS
01/31/2004  03:57 PM             61,632  ui.jar
01/24/2004  12:49 PM             895,924  xercesImpl.jar
01/24/2004  12:49 PM             123,705  xml-apis.jar
           4 File(s)              1,634,113 bytes
           3 Dir(s)          45,020,602,368 bytes free

J:\eclipse\WORKSP~1\Zoom\lib>
```

3. Goto the \$ZOOM_HOME/test/ui directory or create your own test directory to create the ui.properties file. For Windows execute notepad ui.properties or edit ui.properties, for Unix vi ui.properties.



```
C:\WINDOWS\System32\cmd.exe - edit ui.properties
File Edit Search View Options Help
J:\eclipse\workspace\Zoom\test\ui\ui.properties
# MDML Filename and Location
mdml = j:/eclipse/workspace/zoom/test/ui/TabbedApplication.xml
#mdml = j:/eclipse/workspace/zoom/test/ui/SplitPane.xml
#mdml = j:/eclipse/workspace/zoom/test/ui/CalculatorGui.xml

# Rule XML Table
rulexml = rulesSWT.xml
#rulexml = rule.xml

# Project Home Directory
project_directory = j:/eclipse/workspace/zoom/

# Code generator
project_codegenator = j:/eclipse/workspace/zoom/lib/codegen.jar

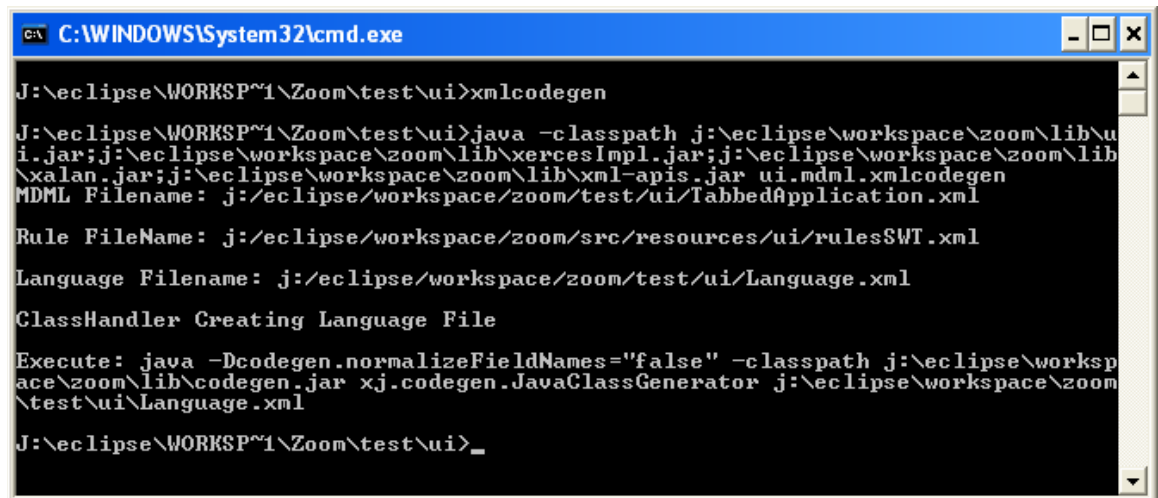
# Project Resources
project_resources_dir = j:/eclipse/workspace/zoom/src/resources/ui/

# Project Output Directory
project_output_dir = j:/eclipse/workspace/zoom/test/ui/

#end

F1=Help | Line:1 Col:1
```

4. After the ui.properties is created, create your own MDML file. Be sure to specify in the MDML file where the UISchema.xsd is located. This file is located in the \$ZOOM_HOME/resources/ui directory.
5. After the mdml files is created execute the XMLcodegen.bat if your are in the \$ZOOM_HOME/test/ui directory. If not copy the XMLcodegen.bat file to your test directory.



```
C:\WINDOWS\System32\cmd.exe
J:\eclipse\WORKSP~1\Zoom\test\ui>xmlcodegen
J:\eclipse\WORKSP~1\Zoom\test\ui>java -classpath j:\eclipse\workspace\zoom\lib\ui.jar;j:\eclipse\workspace\zoom\lib\xercesImpl.jar;j:\eclipse\workspace\zoom\lib\xalan.jar;j:\eclipse\workspace\zoom\lib\xml-apis.jar ui.mdml.xmlcodegen
MDML Filename: j:/eclipse/workspace/zoom/test/ui/TabbedApplication.xml
Rule FileName: j:/eclipse/workspace/zoom/src/resources/ui/rulesSWT.xml
Language FileName: j:/eclipse/workspace/zoom/test/ui/Language.xml
ClassHandler Creating Language File
Execute: java -Dcodegen.normalizeFieldNames="false" -classpath j:\eclipse\workspace\zoom\lib\codegen.jar xj.codegen.JavaClassGenerator j:\eclipse\workspace\zoom\test\ui\Language.xml
J:\eclipse\WORKSP~1\Zoom\test\ui>_
```