Final Report
Version <2.0>

Instructor
Dr. Kwok-Bun Yue

Mentor
Mr. Scott Hetherington

Team#1 Members
Amit Taneja
Dhruv Shah
Kartheek Koganti
Vishal Dhalwani

Spring 2008
25th April 2008
Acknowledgement:

We are thankful to our instructor Dr. Kwok Bun Yue and mentor Mr. Scott Hetherington for their valuable guidance and encouragement in bringing out this project successfully.

We are thankful to Ms. Abeysekera Krishani for providing us with the resources required for our project.

We would also like to thank all the people who helped us in this project and whom we might not have mentioned here.
Table of Contents

1. Abstract ..............................................................................................................1
2. Introduction ........................................................................................................1
3. Functional Requirements: ..............................................................................2
4. Usecase Diagram .............................................................................................3

5. Design and Implementation of the Solution
   5.1 System Architecture ..................................................................................4
   5.2 Flowchart ...................................................................................................6

6. Required Technologies .....................................................................................8

7. Implementation Issues .....................................................................................10

8. Solution of the Problem
   8.1 Tasks Completed .......................................................................................10
   8.2 Tasks to be done ........................................................................................10

9. Lessons Learned .............................................................................................11

10. Future Enhancements ....................................................................................12

11. Conclusions ....................................................................................................12

12. References .......................................................................................................13

Appendix A
   A.1 Time line of our project ............................................................................14
   A.2 Task management ......................................................................................15
   A.3 Team information ......................................................................................16
   A.4 Screenshots ................................................................................................17

Appendix B
   B.1 Classes Developed ....................................................................................23
   B.2 Detailed Algorithm ....................................................................................25
   B.3 Other Libraries ..........................................................................................28

Appendix C
   C.1 Downloads ...............................................................................................29
   C.2 Project Code ..............................................................................................29
   C.3 Project Configuration ................................................................................29
   C.4 Project Deployment ....................................................................................30
1. Abstract:

TieFlow workflow is a workflow design and analysis software developed by Tietronix Software, Inc. Workflow means the automation of a business process, in whole or part, during which the documents, information or tasks are passed from one participant/resource to another for action, according to a set of procedural rules. The existing TieFlow workflow software has two main parts, the editor, which is a standalone java application with a simulation capability and the process automation engine. A user can graphically design the workflow process using the editor. The editor then produces an XML representation (according to a DTD) of the process drawn by the user in the editor. The current simulation tool uses the editor classes to model the workflow and provides simulation functionality.

The aim of our project is to implement a web-based, graphical simulation of the process definition using Flash technology. The manual simulation process will allow the user to choose the disposition of each activity, and view the process execution graphically. Once the simulation is completed the simulation module will generate a report according to the specifications. The graphical representation of the processes, activities and the transitions will be as per the requirements. We have decided to develop the module using Flex.

2. Introduction

The TieFlow Workflow Toolkit is composed of several integrated tools that are used in the development and deployment of automated workflow systems. Process Designer, a primary tool, which is a Java-based application, allows user to graphically define the process definition. It converts the graphical representation of the process defined by users and its underlying data into an XML process definition document according to the
workflow toolkit DTD. Then the XML process definition is imported into the workflow engine for deployment and execution.

The TieFlow Workflow Toolkit is a web-based workflow system used to automate manual, form-based processes. It gets an input from the XML process definition generated by the Process Designer. These process definitions which are specified in XML are parsed and converted to a different XML format according to a DTD created for the workflow engine. The first method will allow the user to manually step through the process, choosing the disposition of each activity, and viewing the process execution graphically.

For all simulation options, a summary report will be generated to show statistical information about the simulation performed. This project is to implement the Flash-based view of the process definition, the first simulation mode, and generate a summary report. Upon selecting a particular execution path, the summary report will provide the user with the information to identify bottlenecks and inefficiencies in the process prior to deploying for production use.

3. **Functional Requirements:** Flash Simulator displays the process in a web-browser graphically using a flash player. The Simulator uses the layout information provided in the DTD of the XML process definition. It allows the user to manually advance the process by selecting one of the available transition options for one of the active activities.

Flash Simulator provides a summary report at the completion of simulation. It gives us the information about the execution path of the process, time to execute and total work time for activities and roles in that particular process. It gives the count for the
number of activities performed, number of different roles involved, and number of activities performed by each role during the simulation.

4 Use case diagram:

Display main page: Once the user accesses the application using URL, simulator displays the main page of the application.

Read XML files list: After the main page is displayed, simulator reads the XML files list from server. The read data is displayed into a combo box by simulator.

Find file: Simulator finds the file selected by user on the server machine.

Parse file: Simulator after finding the selected file parses it.

Display process flow: Simulator displays the process flow using the parsed data from XML file.

Generate report: As user manually steps through the process, simulator generates the report of manual simulation.

Enter Application URL: User accesses the application using URL.

Select XML: User selects the XML once the applications main page is displayed.

Load Process: User after selecting the XML loads the process.

Steps through process: User steps through the process displayed by simulator.
Select transition: User selects transition one at a time while simulating the process manually.

View report: User can view the simulation report either in between the manual simulation or once the simulation is complete.

5. Design and implementation of the solution:

5.1 System Architecture:

Work Flow Editor: The work flow editor is used to design the process definition by using the tool box available. The toolbox has components for drawing start symbol, stop
symbol, activity symbol, Process Ground symbol, edge etc. The process definition is generated in an XML format.

**XML Pool:** All process definitions which are in XML format are saved in a XML pool.

**Accessible File List:** The files available in the XML pool will be listed in a file, which can be accessed by the client browser. This information is displayed in a dropdown list control on a client browser when the TieFlow application loads in the browser.

**Flash Simulator:** After loading the process definition the XML process definition is parsed and is recreated according to the format accepted by the visual library. It displays the workflow definition graphically. The simulator consists of a engine which comprises

Figure 5.1.1 High-level architecture diagram
of transition rules which are used to simulate the workflow process. Our simulator requires flash player on the client’s browser to run the generated rich internet application.

To summarize the whole process, the workflow editor generates a process definition which is an XML file stored on the server XML pool. Its availability information is stored in the XML file list stored on the server. This file is read by the application using http services. It populates the available information in a combo box when the application is launched on a client browser. Flash player is required to run the application in the client browser. The events are handled by the simulator on the client side and the object stores the activities performed in the simulation and generates a report, which is not stored anywhere.

**Note: Refer to Appendix C for Low level Architecture**

**5.2 Flowchart Diagram:**

Process definitions are generated by the workflow editor and saved on the server. These are displayed in the user interface and selected from the combo box which shows the available Process Definitions. The availability information is stored in an xml file and it is saved on the server. Once the process definition is selected, the next step is to load the process. By clicking the load process button the process gets loaded and displayed in a canvas. The report can be checked when ever the user desires to view the actions performed. Upon clicking the start the process gets activated and it displays the activities which are to be activated (displays the path which can be followed). By stepping through transitions upon activating all the activities the process gets ended once it reaches the End. The report can be viewed by clicking the view report button provided beneath the
canvas. The report displays the information about the actions performed in the simulation process.

5.2.1 Flowchart diagram
6. **Required Technologies:** We have used the Flex builder as our IDE, used Action Script 3.0 and MXML as the programming languages. Swf files generated by the flex application that run in Adobe Flash player 9 on the client side web browser.

**Flex Builder:** Adobe Flex Builder is an integrated development environment (IDE) for developing applications that use the Adobe Flex framework, MXML, Adobe Flash Player 9 and ActionScript 3.0. Adobe's Flex Builder IDE and Flex SDK generate SWF (runs in flash player) files from MXML. Developing user interface is very easy using flex builder. This IDE is mainly used to develop user interface, build, run, and debug our project code. Built-in classes/API’s like XML, XMLList, XMLDocument, etc makes the application parse the XML data very easily.

**MXML:** It is an XML-based user interface markup language which in combination with ActionScript is used to develop applications in Flex. Since flex doesn’t have the concept of a page, this project has different MXML tags like mx: States, mx: ViewStack, etc to change the view of the application. These interfaces provide a smooth transition between different views in a page. The most significant difference is that MXML-defined user interfaces are rendered by Flash Player, providing the users with a much more engaging experience.

**Action Script 3.0:** ActionScript is used primarily for the development of websites and software using the Adobe Flash Player platform (in the form of SWF files embedded into Web pages). The language enables developers to write complex programs that perform efficiently and responsively. Since its inception Action Script supports object-oriented behavior which can be used to represent real-time objects. Our project uses Action Script
3.0 to represent processes, activities and edges as objects. All the event handling on these objects has also been handled using this.

**Flex Visual Graph Library:** This library has a MIT license. The Flex Visual Graph Library is a community project to advance the design and development of an open source data visualization library and component suite for Adobe Flex. The library is very robust by extending and providing separation of base, interface, and layout code. Additional layout algorithms can be readily integrated as an extended class containing only the mathematical calculations and controls needed specifically for the layout. The Visual Graph is a Flex Component for visualizing data.

The library basically uses an XML representation of the nodes and edges. This representation is then displayed visually in the flash player. Our application parses and converts the process definition XML into the XML format required by the library. The library is extended to display different shapes according to the type of the nodeClass.

**Features of Flex Visual Graph Library**

- Fully customizable – It is an open source library.
- Comparatively easy to use with basic Mxml and action script scripting;
- Self-contained – all mathematical processing is performed within the library;
- This library can be used both for commercial and non commercial purposes without any purchasing.

**Cons:** The performance of the library is contingent on the host computer processing capacity which may limit the efficiency of graphing high volume data sets.
7. **Implementation Issues**: We can not access the file system with Flex due to the Flash security with the exception of uploading files which give you tightly regulated access. This can be resolved by using a java applet which does the file operations, but in our project, the administrator needs to maintain a list of available workflow process definitions on the server machine in an XML format file which is accessible and read by our application.

8. **Solution of the problem**

   Our task is to process an input XML generated from the workflow process editor. And parse it in the format required by the Visual Graph library to draw the process graphically. Once the process is displayed graphically, next task is to manually step through the process using the transition rules provided. The report is generated automatically as the user steps through the process and this report can be viewed any time during the execution. User gets the current status of the activities completed in the process.

**Tasks completed:**

1. Developed a web-based application.
2. Displayed the work flow process graphically.
3. Simulation of manual mode.
4. Generated a summary report for the activities performed in the simulation.

**Tasks to be done:**

1. Graphical display of transition between Parent process to sub process and activities involved subprocess has to be displayed.
2. Handle invalid process leading to deadlock and unsynchronized state.
3. Graphical display of disposition and transitions from activities to activities and expand the process directly and fit into the canvas.

9 Lessons Learned:

9.1 Importance of Research

As we are new to workflow processes and also to the technologies that are required to develop an application, getting to know them required a lot of research. We had done lot of research on action script 3.0, flex technology and various visual graph libraries to display the workflow process graphically. The time that we all spent on the research really helped to implement the required project.

9.2 Team Work

Working in a team with different people is very challenging and reaching a consensus might be tough in certain situations.

9.3 Time Management

Time management is another important lesson that needs to be considered when meeting various deadlines. It was challenging and gave us a valuable experience to complete the work on time.

9.4 Gained valuable experience

Gained valuable knowledge and experience working on this project. Acquired knowledge about technologies like Action Script, MXML, and flex by working in this project which is totally new to us.
10. Future Enhancements

- Error handling and generating warnings when undesired actions are performed.
- Making the visual look of simulator more interesting and development of directed connectors with arrow heads
- Uploading XML files to simulator and reading XML process definition pool from a file system.
- The second simulation mode will execute the process definition multiple times; each time with a different path through the process until all unique paths are complete or until specified maximum executions is reached.
- The third simulation mode will execute the process definition a specified number of times, with the path through the process determined by probabilities entered by the user.
- The fourth simulation mode will execute the process definition that uses the concepts of time of day and resource availability to provide more realistic simulation results and provide identification of bottlenecks in the process execution.

11. Conclusion:

We have developed a professional web application for simulation of Tieflow workflow process definitions. Our current work enables user to manually step through the process and provides various parameters like time to complete a particular execution path, time to complete each activity, etc and thus assures the better process control and business process improvement. Besides developing the project we learnt to work professionally as a team to meet the deadlines set by ourselves.
12. References:

The following references have been used in order to understand the underlying technologies used in developing this prototype.

[1] Team website http://dcm.uhcl.edu/cap08spgp1/


[4] SpringGraph Library – Adobe License
http://mark-shepherd.com/blog/2006/11/17/a-flex-component-for-graph-visualization/


[6] Flex Resources -

[7] Dr. Yue’s Website - http://nas.cl.uh.edu/yue

[8] Flex visual graph library documentation http://dcm.uhcl.edu/cap08spgp1/fvg-doc/
Appendix A:

A.1 Project Management:

A.1.1 Time Line of Our Project:

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Team Setup</td>
<td>-</td>
</tr>
<tr>
<td>Website</td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
</tr>
<tr>
<td>Redefine requirements</td>
<td></td>
</tr>
<tr>
<td>Research and development</td>
<td></td>
</tr>
<tr>
<td>Study the Visual Graph Library</td>
<td></td>
</tr>
<tr>
<td>Adopt the library to the need</td>
<td></td>
</tr>
<tr>
<td>Design and Code Simulation program</td>
<td></td>
</tr>
<tr>
<td>Converting the XML to the library needs.</td>
<td></td>
</tr>
<tr>
<td>Design and coding – Summary Report</td>
<td></td>
</tr>
<tr>
<td>Testing and Enhancement</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
</tr>
</tbody>
</table>

Table A.1.1.1 Timeline
## A.1.2 Time Management

### Tasks

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Amit Taneja</th>
<th>Dhruv Shah</th>
<th>Kartheek Koganti</th>
<th>Vishal Dhalwani</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Study and Research</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Task Management</td>
<td>15</td>
<td>5</td>
<td>10</td>
<td>70</td>
</tr>
<tr>
<td>Web Site Maintenance</td>
<td>25</td>
<td>65</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Presentations &amp; Meeting Participation</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Simulation Algorithm</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>Use-Cases for Manual Simulation mode</td>
<td>15</td>
<td>10</td>
<td>65</td>
<td>10</td>
</tr>
<tr>
<td>Design of User Interface</td>
<td>30</td>
<td>55</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Loading the application using URL and reading the files list.</td>
<td>30</td>
<td>35</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>Parsing the XML</td>
<td>70</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Displaying the Process Definition using the parsed data</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>Summary Report</td>
<td>20</td>
<td>15</td>
<td>25</td>
<td>40</td>
</tr>
<tr>
<td>Refinement in Summary Report</td>
<td>20</td>
<td>25</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>Testing &amp; Error Debugging</td>
<td>15</td>
<td>15</td>
<td>65</td>
<td>5</td>
</tr>
<tr>
<td>Documentation</td>
<td>25</td>
<td>30</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

### A.1.2.1 Task Management
A.1.2 Team Information:

**Amit Taneja** – Involved in Research & design, technical writing, development and Documentation.

**Dhruv Shah** – Involved in Research & design, website maintenance, development & Documentation

**Kartheek Koganti** – Involved in Research & design, development, Testing and Documentation

**Vishal Dhalwani (Team Leader)** – Coordinated team meetings and assigned asks. Involved in Research & design, development and documentation.
A.1.3. Screen shots

**Step 1:** This is the homepage of the TieFlow Workflow Toolkit, which displays a greeting about TieFlow Workflow Toolkit.

![Figure A.1.3.1](image-url)
**Step 2:** This is the second page where the user has the option to select the process definition which is displayed in the dropdown list. Here the user is also given an option to load the process or to go the home page.

![Tieflow Workflow Toolkit](image)

**Figure A.1.3.2**
Step 3: This page shows us the workflow process once it loads on the canvas. It has information regarding different components like Start, End, Activities, Roles, Transitions, And and Or’s of the process.

Figure A.1.3.3
**Step 4:** This page displays the activities which are executed, which are active and which activities are to be executed in the simulation.

![Tieflow Workflow Toolkit](image)

Figure A.1.3.4
Step 5: This page displays shows that all the activities are executed and the process is completed.

Figure A.1.3.5
**Step 6:** This page gives information about the summary report about the execution path, the number of activities and roles involved, the total work time and the total time to execute the whole process etc.

![Workflow Toolkit](image)

**Figure A.1.3.6**
Appendix B:

B.1 Classes developed:

Classes used in parsing of input XML i.e. workflow process definition:

**Parse.as:** Parses the input XML file i.e. workflow process definition selected by the user.

**Package.as:** This class has certain elements and/or attributes contained in Package element of input XML like Package id, Package name.

**Parameter.as:** This class has certain elements and/or attributes contained in Parameter element of input XML like Parameter id, Parameter name.

**Role.as:** This class has certain elements and/or attributes contained in Role element of input XML like Role id, Role name.

**Process.as:** This class has certain elements and/or attributes contained in Process element of input XML like Process id, Process name, Process version, Process creator.

**ProcessStart.as:** This class has certain elements and/or attributes contained in ProcessStart element of input XML like ProcessStart id, ProcessStart EntryPoint, ProcessStart ExitPoint.

**Activity.as:** This class has certain elements and/or attributes contained in Activity element of input XML like id, name, AssigneeRole, EntryPoint, ExitPoint, TimeToComplete, TotalTimeToComplete.

**Edge.as:** This class has certain elements and/or attributes contained in Edge element of input XML like id, name, origin, destination, entrypoint, exitpoint.

**ProcessGround.as:** This class has certain elements and/or attributes contained in ProcessGround element of input XML like ProcessGround id.
**ProcessAnd.as**: This class has certain elements and/or attributes contained in the ProcessAnd element of input XML like ProcessAnd id.

**ProcessEnd.as**: This class certain elements and/or attributes contained in the for the ProcessEnd element of input XML like ProcessEnd id.

**Classes to implement transition rules:**

**wfgEngine.as** – It handles all the events occurred on display components. It implements the transition rules and records the information activities performed. This information is used to generate a summary report.

**Classes used for displaying**

**Graph.as** – Graph implements a graph data structure G (V, E) with vertices V and edges E, except that we call the vertices nodes, which is here more in line with similar implementations. A graph may be associated with a VisualGraph object, which can visualize graph components in Flash.

**VisualGraph** - This component can visualize and layout a graph data structure in a Flex application. It is derived from canvas and thus behaves much like that in general. Currently the graphs are required to be connected. And for most layouts a root node is required (as they are tree based). A graph object needs to be specified as well as a layouter object that implements the ILayoutAlgorithm interface. Force directed layouter(vgraph)

**DirectedEdgeRenderer.as** - This is a directed edge renderer, which draws the edges with slim balloon like curves that indicate a source. Please note that for undirected graphs, the actual direction of the edge might be arbitrary.
**IViewFactory.as** – This is an addition to the library. It is taken from the spring graph library. This is an interface that knows how to create views.

**wfgSNode.as:** This is an addition to the library which implements IViewFactory. It gets UICOMPONENT (shape) depending on the node class.

**wfgSNodeRenderer.mxml** – This is an addition to the library. It is used to render the activity component.

**wfgENodeRenderer.mxml** – This is an addition to the library. This is used to render the edge component.

**wfgANodeRenderer.mxml** – This is an addition to the library. This is used to render the process And component.

**Classes used to display shapes:**

**Roundrect.as** – This class has the drawing API’s for drawing a round rectangle

**Circle.as** – This class has the drawing API’s for drawing a circle.

**Rectangle.as** – This class has the drawing API’s for drawing a rectangle.

**B. 2. Detailed Algorithm**

**Algorithm for parsing of input XML i.e. workflow process definition:**

1. Load the selected XML file using HTTP Services

2. Read the XML file from step 1

3. Parse the input XML in **Parse.as** as follows:

   A. For each **Package element** node in XML file: Create an object of Package.as class and store the information like id, name, version, etc of the Package element.
B. For each **Parameter** element node in XML file: Create an object of Parameter.as class and store the information like Parameter id, Parameter name, etc of the Parameter element.

C. For each **Role** element node in XML file: Create an object of Role.as class and store the information like Role id, Role name, all activities related with the particular Role etc of the Role element.

D. For each **Process** element node in XML file: Create an object of Process.as class and store the information like Process id, Process name, Process version, Process creator, and Process state etc of the Process element.

E. For each **ProcessStart** element node in XML file: Create an object of ProcessStart.as class and store the information like ProcessStart id, ProcessStart EntryPoint, ProcessStart ExitPoint, etc of the ProcessStart element. Add the ProcessStart id to the Node id attribute, “start” to the Node class attribute in the Node element. Update this XML node which is used in visual graph library.

F. For each **Activity** element node in XML file: Create an object of Activity.as class and store the information like Activity id, Activity name, Activity EntryPoint, Activity ExitPoint, Activity TimeToComplete, Activity TotalTimeToComplete, all assignee roles related to the activity etc of the Activity element. Add the activity id to the Node id attribute, “activity” to the Node class attribute in the Node element. Update this XML node which is used in visual graph library.

G. For each **Edge** element node in XML file: Create an object of Edge.as class and store the information like Edge id, Edge origin, Edge destination, Edge EntryPoint, Edge ExitPoint, etc of the Edge element. Add the edge id to the Node id attribute,
“edge” to the Node class attribute in the Node element. Update this XML node which is used in visual graph library.

H. For each ProcessAnd element node in XML file: Create an object of ProcessAnd.as class and store the information like ProcessAnd id, etc of the ProcessAnd element. Add the ProcessAnd id to the Node id attribute, “pand” to the Node class attribute in the Node element. Update this XML node which is used in visual graph library.

I. For each ProcessGround element node in XML file: Create an object of ProcessGround.as class and store the information like ProcessGround id, etc of the ProcessGround element. Add the ProcessGround id to the Node id attribute, “pgrnd” to the Node class attribute in the Node element. Update this XML node which is used in visual graph library.

J. For each ProcessEnd element node in XML file: Create an object of ProcessEnd.as class and store the information like ProcessEnd id, etc of the ProcessEnd element. Add the ProcessEnd id to the Node id attribute, “end” to the Node class attribute in the Node element. Update this XML node which is used in visual graph library.

B.3 Other Libraries visited – spring graph library: The Spring Graph component displays a set of objects, using a force-directed layout algorithm to position the objects. Behind the objects, the component draws lines connecting items that are linked. This library is an adobe library. This is very simple to implement. But we need to purchase this license if we have to use for commercial purposes.
The user interface of flash simulator is developed in flex builder IDE. This user interface uses mxml tags for the GUI. E.g. button controls, combobox control, drawing canvas, etc. The in-built classes/API is used in the implementation of different action script classes like Parser, visual graph and workflow engine.

When the user interface is loaded, an event is generated to read data of the accessible list which is available on server. The URLLoader class of HTTPService API is used to access this list from the server. The data in the list (list of available process definitions) is displayed in the combo box at user interface.

Figure B.1.1 Low-level Architecture diagram
Once the user selects the process definition from combo box and clicks the button control (e.g. Load Process button) an event is generated and Parser action script class is called. This class parses the selected XML file in the combo box and gives the data to visual graph class. The visual graph class accepts the input in certain XML format and draws the process graph on the canvas on user interface. Once the process graph is drawn, user manually steps through the process with the use of mouse clicks. The handling of events and transition rules are incorporated in workflow engine class.

Appendix D:

D.1. Downloads:

- Flex Builder from http://www.adobe.com/products/flex/
- Visual graph library from http://code.google.com/p/flexvizgraphlib/

D.2. Project Code:

We have provided the project code on our project team website:

http://dcm.uhcl.edu/cap08spgp1/deliverables.html

D.3 Project Configuration (Read Me)

- Folders required : img(images), lib(.swc resides), org (contains packages used), tfl.mxml
- Install flex builder.
- Open the flex builder IDE.
- Create a new flex project.
- Import the main folder which consists of all the folders mentioned above.
- Add the .swc contained in the library path.
- Add the org folder in the source path.
D.4. Read Me (Deployment):

Step 1> Upload the following files generated from the flex builder.

a. Main.css
b. tfl.html
c. AC_OCTags.js
d. history.js
e. history.swf
f. tfl.swf
g. XMLInput.xml

Note: All the process definition files that have entries in the XMLInput.xml should uploaded/present to the server.