Project Overview

- **Introduction**
  - Producer Express (PX), and PX automates the interaction between producers, distributors, and carriers.
  - AutoPilot is one of the central concepts in PX that is a completely configurable system.
  - AutoPilot is driven by the open-source project OSWorkflow and its task is to configure the XML files to define the steps in a workflow.
  - A graphical desktop application will be useful to assist users to minimize the amount of time and work to configure workflows: **AutoPilot Workflow Editor**

- **Purpose**
  - To design a graphical desktop application in order to configure AutoPilot workflows and create/view/edit the corresponding XML files.

- **Scope**
  - Useful tool to create/view/edit the corresponding XML files graphically using basic flowchart shapes.
  - Will decrease the required time spent by users for dealing with XML configuration files which results in an increase productivity.
  - Sircon's PSO analysts will benefit from the application as well as Sircon's development and professional support teams.
System Components

• Software Platforms / Technologies
  – Java
  – NetBeans IDE
  – SWING UI
  – XMLBeans Parser Library
Functional Overview

• Current Workflow Editing Process

- Manually Code XML Files
- Interviews With New Sircon Customer (Wishes to sell insurance)
- Becomes Producer Express (New Client Interview Web Application)
- Becomes Registered Customer

• New Workflow Editing Process

- Auto-Generates XML Files
- Interacts With PSO Analyst
- Becomes Autopilot Workflow Editor (AWE)
- Provides backend function configuration for Registered Customer
- New Sircon Customer (Wishes to sell insurance) Interviews With
Project Schedule

Overall

1. Framework
   a) Framework for Project Complete
   b) Date: Feb. 5

2. Alpha Demonstration
   a) Internal Representation and Output to Screen
   b) Date: Feb. 15

3. Beta Demonstration
   a) Everything except for non-crucial features
   b) Date: Mar. 14

4. Project Video
   a) Begin immediately following Beta Demonstration
   b) Date: Mar. 28
Project Schedule

XML Team

1. XML Parser
   a) XML Parser + XML Beans Data Structures set up
   b) Date: Jan. 30
2. Domain Model
   a) Class Structure and Implementation
   b) Date: Feb. 5
3. XML Auto-correction
   a) Auto-correct nesting of XML
   b) Date: Feb. 12
4. XSD Format
   a) XSD File Format for Functions + Decision Points
   b) Date: Feb. 15
5. Branch Structure
   a) Internal Rep for GUI Branch Component
   b) Date: Feb. 29
GUI Team

1. GUI Components
   a) Display and Place GUI components to screen
   b) Date: Feb. 1
2. Drawing Workflow
   a) Drawing Domain Model Objects in Workflow
   b) Date: Feb. 7
3. XML Auto-correction
   a) Auto-correct nesting of XML
   b) Date: Feb. 12
4. XSD Format
   a) XSD File Format for Functions + Decision Points
   b) Date: Feb. 15
**XML Tag Definitions (backend)**

- Read and writing XML files will be based upon the OSWorkflow XML Schema defined on the website.
- Example Tag Definition from Schema:

  **Result** - a conditional result of an action

<table>
<thead>
<tr>
<th></th>
<th>XML Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Result</strong></td>
<td>&lt;xs:element name=&quot;result&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td><a href="">xs:complexType</a></td>
</tr>
<tr>
<td></td>
<td><a href="">xs:sequence</a></td>
</tr>
<tr>
<td></td>
<td>&lt;xs:element ref=&quot;conditions&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:element ref=&quot;validators&quot; minOccurs=&quot;0&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:element ref=&quot;pre-functions&quot; minOccurs=&quot;0&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:element ref=&quot;post-functions&quot; minOccurs=&quot;0&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/xs:sequence&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;id&quot; type=&quot;xs:string&quot; use=&quot;optional&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;owner&quot; type=&quot;xs:string&quot; use=&quot;optional&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;join&quot; type=&quot;xs:string&quot; use=&quot;optional&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;display-name&quot; type=&quot;xs:string&quot; use=&quot;optional&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;step&quot; type=&quot;xs:string&quot; use=&quot;optional&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;old-status&quot; type=&quot;xs:string&quot; use=&quot;required&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;status&quot; type=&quot;xs:string&quot; use=&quot;optional&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;split&quot; type=&quot;xs:string&quot; use=&quot;optional&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;xs:attribute name=&quot;due-date&quot; type=&quot;xs:string&quot; use=&quot;optional&quot; /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/xs:complexType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/xs:element&gt;</td>
</tr>
</tbody>
</table>
XML Definitions

- **Functions** - A function that is executed automatically by OSWorkflow or by in-house code
  - An essential element within the XML and GUI
  - Various types of Functions, such as ‘class‘ or ‘spring‘ type
  - *Class Functions* (Java-based): the ClassLoader must know the class name of your function. This can be accomplished with the argument class.name

<table>
<thead>
<tr>
<th>Function (type=”class”)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
</tr>
<tr>
<td><code>&lt;function type=&quot;class&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;arg name=&quot;class.name&quot;&gt;com.acme.FooFunction&lt;/arg&gt;</code></td>
</tr>
<tr>
<td><code>&lt;arg name=&quot;message&quot;&gt;The message is ${message}&lt;/arg&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/function&gt;</code></td>
</tr>
</tbody>
</table>
The OSWorkflow contains a large number of possible tags that must be accounted for within our application:
- Action, Arg, Common-Action, Condition, Function, Global-Actions, Global-Conditions, Initial-Actions, Join, Meta, Permission, Post-Functions, Pre-Functions, Register, Restrict-to, Results, Split, Step, Trigger Functions, Unconditional Result, Validator, Workflow

Reading data and writing data could be easily accomplished with a data structure hierarchy based upon tags.

However our GUI will only represent a simplified viewpoint.

Our Domain Module (class structure for data in the front-end) will only view the workflow as functions (normal, pre, post) and decision points.

Therefore a bidirectional translator between two data hierarchies will be necessary.
Class Diagrams

• Autopilot Workflow Editor: Class Diagram Part 1
Class Diagram - Cont.

- Autopilot Workflow Editor: Class Diagram Part 2
Prototype

• Autopilot Workflow Editor: Prototype Part 1
Prototype – Cont.

- Autopilot Workflow Editor: Prototype Part 2
Prototype – Cont.

• Title: Autopilot Workflow Editor: Prototype Part 3