Project Overview

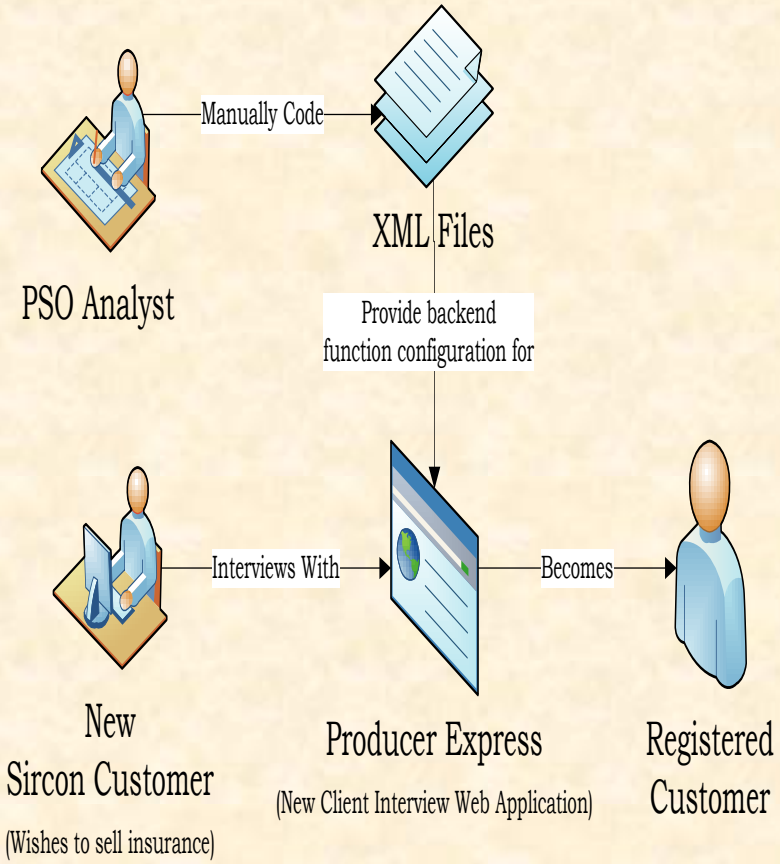
- **Introduction**
  - Producer Express (PX), and PX automates the interaction between producers, distributors, and carriers .
  - AutoPilot is one of the central concepts in PX that is a completely configurable system
  - AutoPilot is driven by the open-source project OSWorkflow and its task is to configure 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

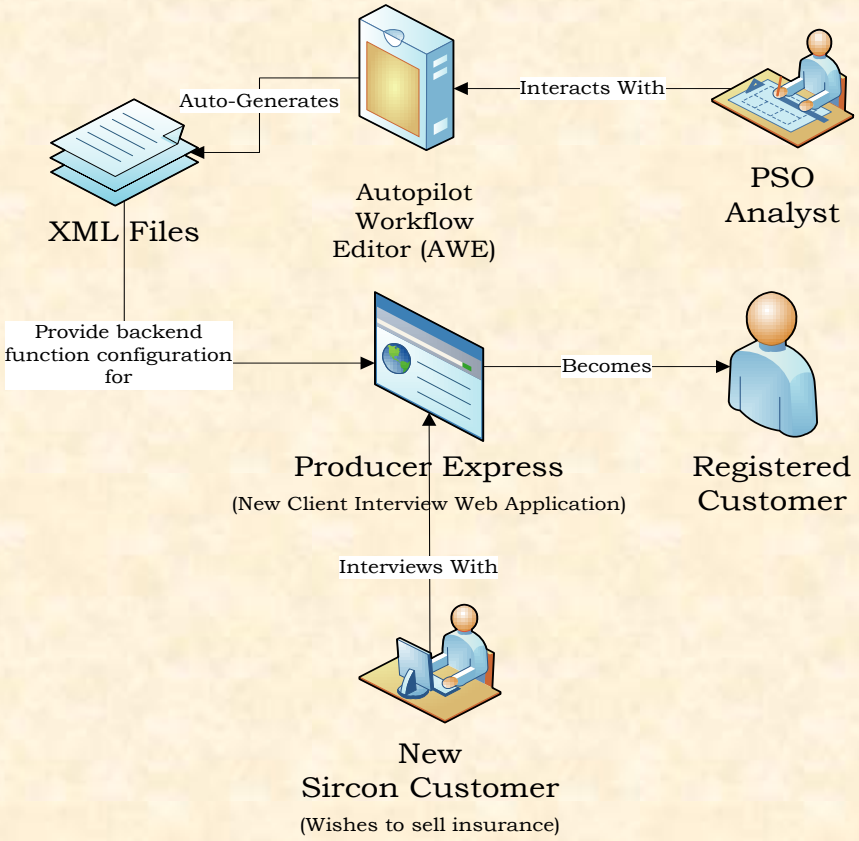
- Software Platforms / Technologies
  - Java
  - NetBeans IDE
  - SWING UI
  - XMLBeans Parser Library

Functional Overview

- Current Workflow Editing Process



- New Workflow Editing Process



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

Project Schedule

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 Definitions

- **XML Tag Defintions (backend)**
  - Read and writng 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

Result
XML Syntax

```
<xs:element name="result">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="conditions" />
   <xs:element ref="validators" minOccurs="0" />
   <xs:element ref="pre-functions" minOccurs="0" />
   <xs:element ref="post-functions" minOccurs="0" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="optional" />
  <xs:attribute name="owner" type="xs:string" use="optional" />
  <xs:attribute name="join" type="xs:string" use="optional" />
  <xs:attribute name="display-name" type="xs:string" use="optional" />
  <xs:attribute name="step" type="xs:string" use="optional" />
  <xs:attribute name="old-status" type="xs:string" use="required" />
  <xs:attribute name="status" type="xs:string" use="optional" />
  <xs:attribute name="split" type="xs:string" use="optional" />
  <xs:attribute name="due-date" type="xs:string" use="optional" />
 </xs:complexType>
</xs:element>
```

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

Function (type="class")
Example
<function type="class"> <arg name="class.name">com.acme.FooFunction</arg> <arg name="message">The message is ${message}</arg> </function>

Back-end Scope

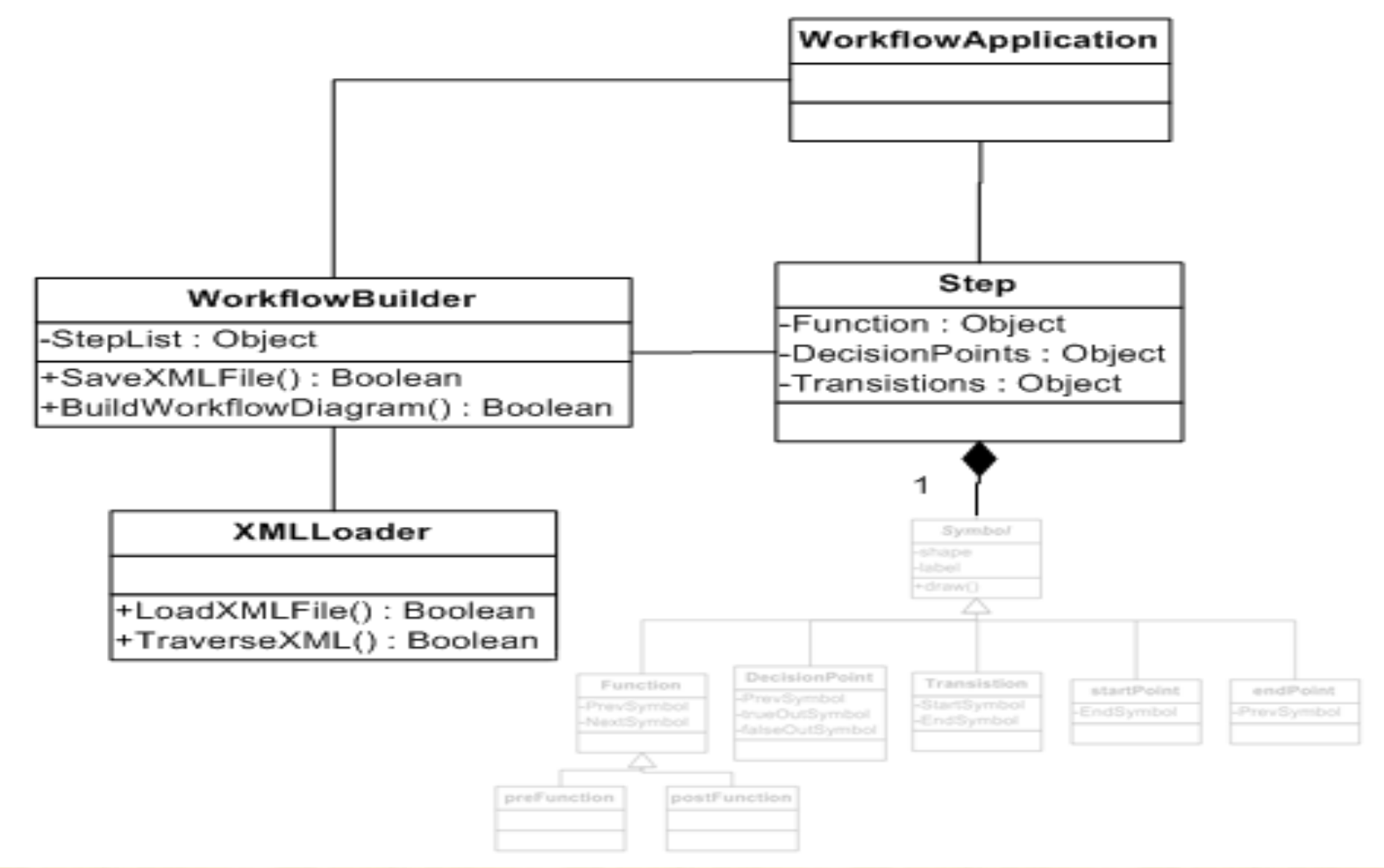
- The OSWorkflow contains a large number of possible tags that must be accounted for within our application
  - Action, Arg, Common-Action, Condition, Function, Global-Actions, Global-Conditions, Initial-Actions, Join, Meta, Permission, Post-Functions, Pre-Functions, Register, Restrict-to, Results, Split, 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

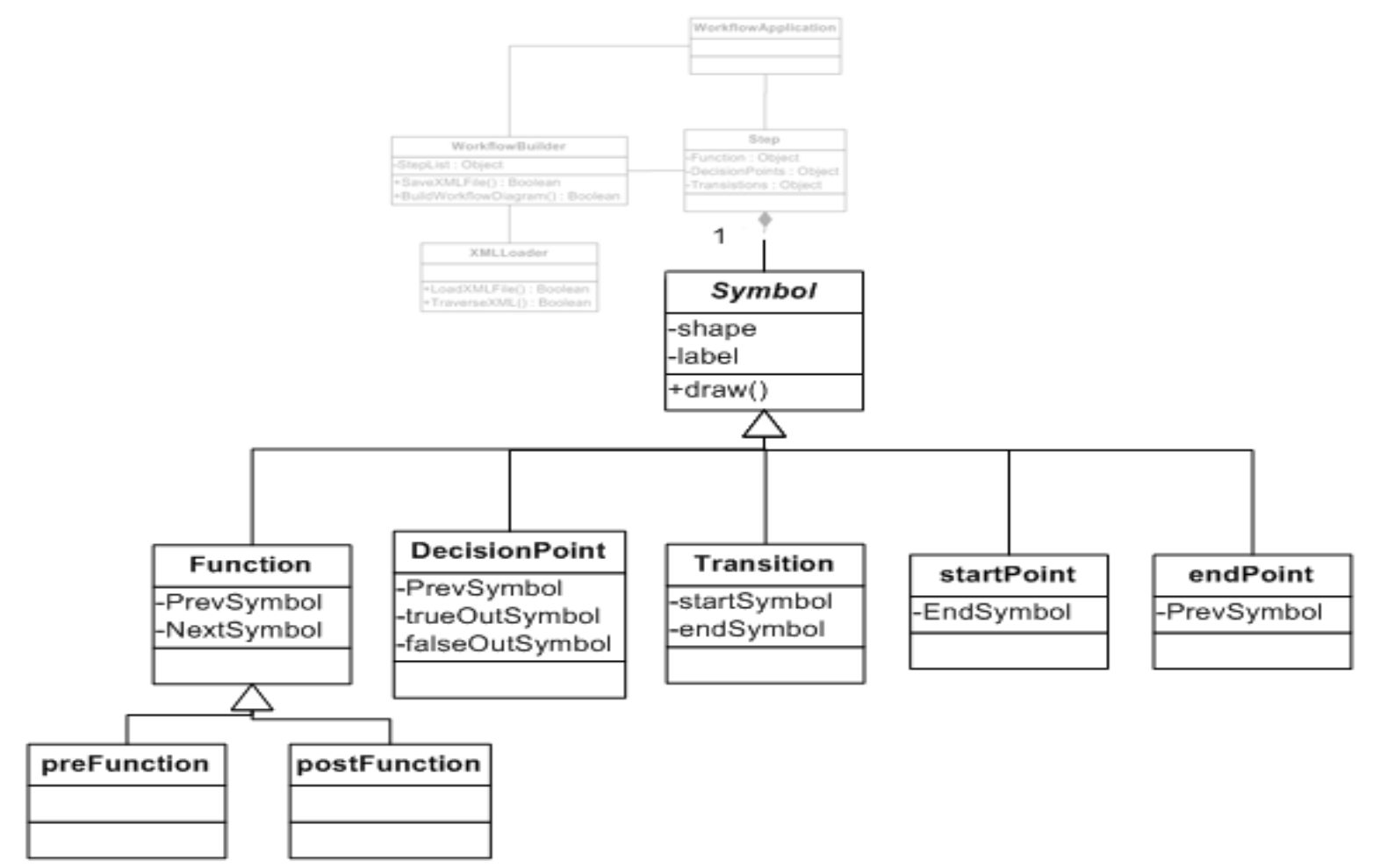
Class Diagrams

- Autopilot Workflow Editor: Class Diagram Part 1



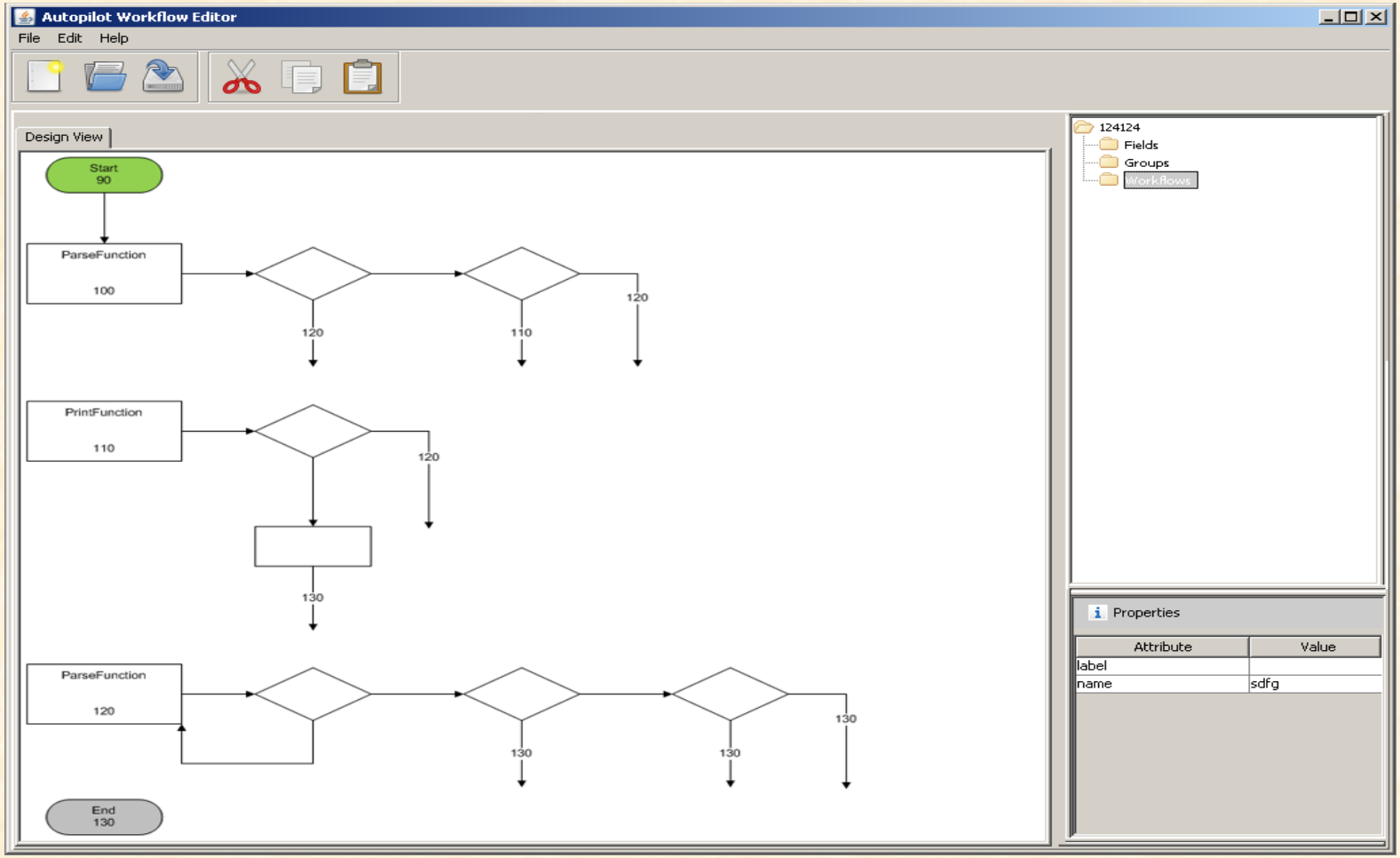
Class Diagram- Cont.

- Autopilot Workflow Editor: Class Diagram Part 2



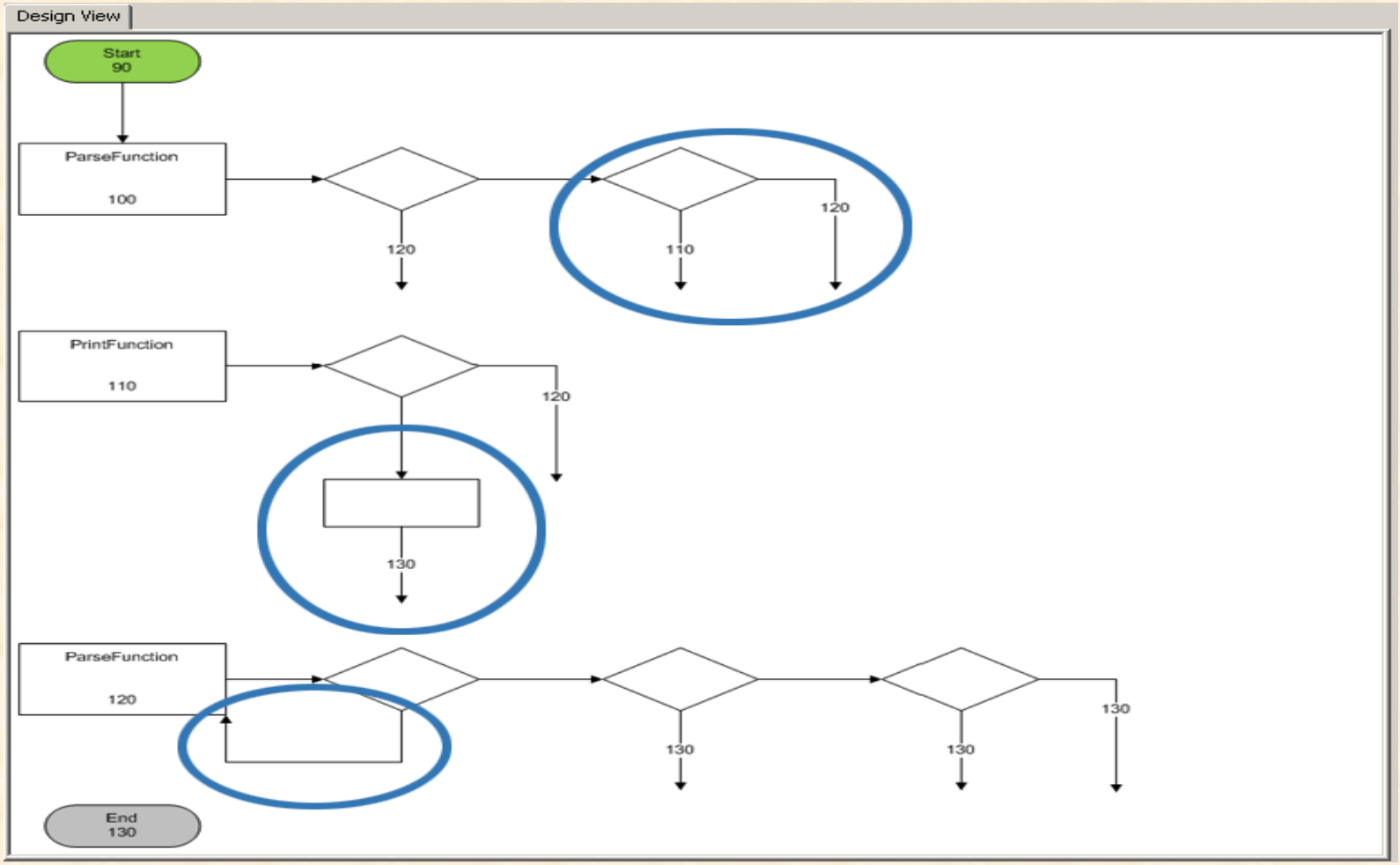
Prototype

- Autopilot Workflow Editor: Prototype Part 1



Prototype – Cont.

- Autopilot Workflow Editor: Prototype Part 2



Team 08: Sircon

Prototype – Cont.

- Title: Autopilot Workflow Editor: Prototype Part 3