Microsoft DSL Tools Lab

- Aim of section:
  - Create a simple UML tool using a template
  - Play with a multi-view example
  - Add a Component Diagram type to the multi-view example
  - Hands-on help from Karen

Create a simple UML tool

- Start Microsoft Visual Studio 2008

- Create a new project (File → New Project)
  - Choose Extensibility → Domain-specific Language Designer
  - Choose the Class Diagrams template and follow the wizard

- Transform (icon on Solution Explorer window)

- Run ▶ (icon on tool bar)
  - Play with the provided model examples in the Experimental Hive - investigate structure and behaviour

- Perform the above steps again to create another project using the Component Models template and study the content (this is to be integrated into the multi-view example – see later)

Multi-view implementation

- Download the multi-view tool example (Extended Class Diagram) from http://www.netfxfactory.org/blogs/papers/archive/2009/01/13/multiply-dsl-points-of-view.aspx
  - Read the blog to understand how multi-views can be implemented
- Run the project and create examples
  - Explore how to open and edit the multi-views (Flow Diagram and Package Diagram)

Add a Component Diagram type

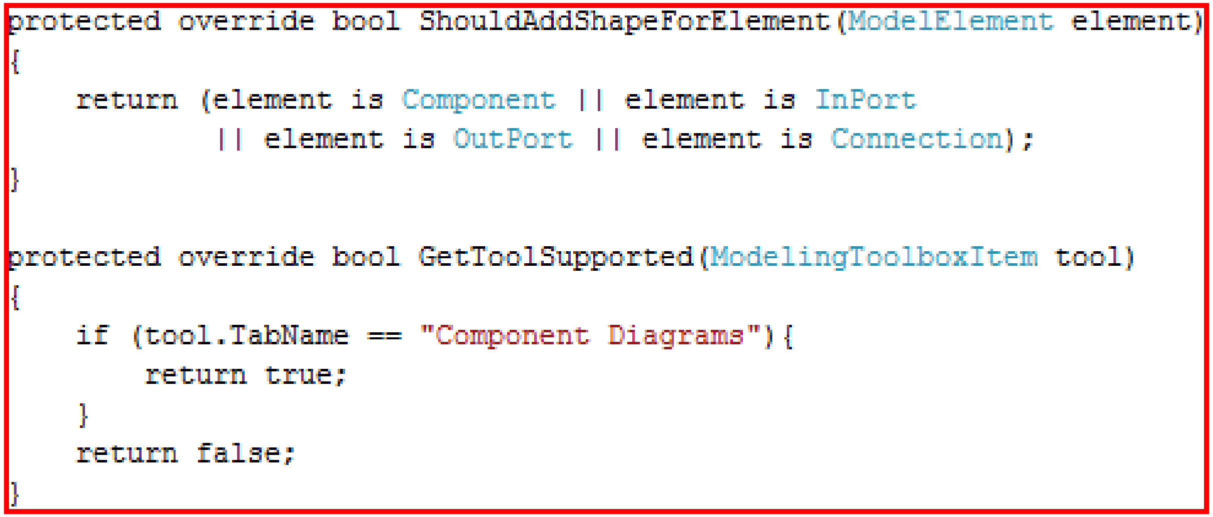
1. In the Extended Class Diagram project, add to DslDefinition.dsl
   - Domain classes (ref the Component Models based project)
     - Component, ComponentPort, InPort, OutPort
   - Domain relationships
     - ModelRootHasComponents, ComponentHasPorts, Connection (OutPort to InPort)
   - Inheritances
     - InPort and OutPort inherit ComponentPort
   - Diagram Elements
     - ComponentShape, InPortShape, OutPortShape, AssociationLink (for port connection)
   - Maps
     - Component - ComponentShape, InPort - InPortShape, OutPort - OutPortShape, Connection - AssociationLink

Add a Component Diagram type

2. Create a Toolbox Tab for Component Diagram
    - Go to DSL Explorer → Editor → Add New Toolbox Tab: Component Diagrams
    - Add New Element Tool
        - Component, InPort and OutPort
    - Add New Connection Tool
        - Connection

Add a Component Diagram type

3. Create a ComponentDiagram.cs file (follow FlowDiagram.cs)
   - New GUID (can use an online GUID generator)
   - ShouldAddShapeForElement
   - GetToolSupported



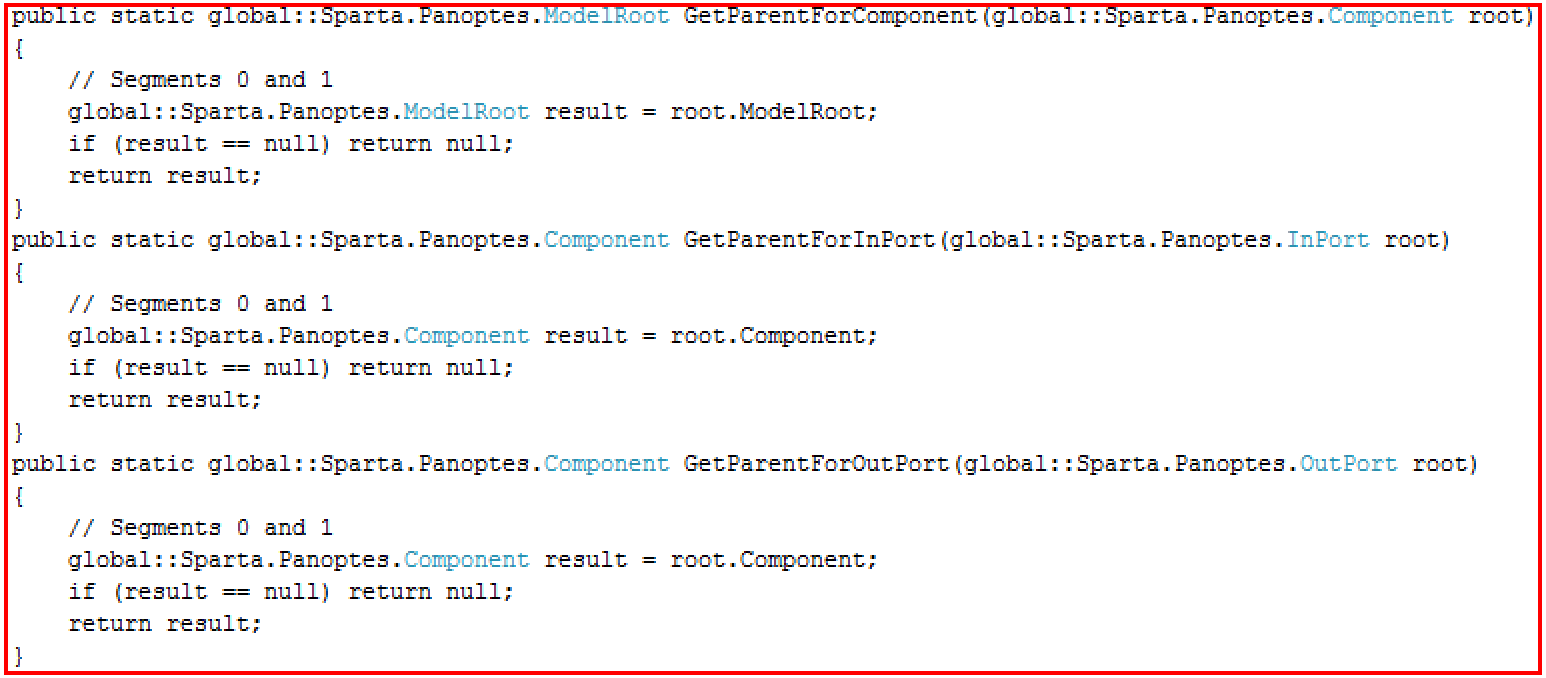
```
protected override bool ShouldAddShapeForElement(ModelElement element)
{
    return (element is Component || element is InPort
            || element is OutPort || element is Connection);
}

protected override bool GetToolSupported(ModelingToolboxItem tool)
{
    if (tool.TabName == "Component Diagrams"){
        return true;
    }
    return false;
}
```

4. Add to DomainModel.cs

Add a Component Diagram type

5. FixUpDiagramOnElementAddedRule.cs

- Add RuleOn attributes for Component, InPort, OutPort and Connection
- Create GetParentForComponent, GetParentForInPort, GetParentForOutPort methods and their calls



```
public static global::Sparta.Panoptes.ModelRoot GetParentForComponent(global::Sparta.Panoptes.Component root)
{
    // Segments 0 and 1
    global::Sparta.Panoptes.ModelRoot result = root.ModelRoot;
    if (result == null) return null;
    return result;
}
public static global::Sparta.Panoptes.Component GetParentForInPort(global::Sparta.Panoptes.InPort root)
{
    // Segments 0 and 1
    global::Sparta.Panoptes.Component result = root.Component;
    if (result == null) return null;
    return result;
}
public static global::Sparta.Panoptes.Component GetParentForOutPort(global::Sparta.Panoptes.OutPort root)
{
    // Segments 0 and 1
    global::Sparta.Panoptes.Component result = root.Component;
    if (result == null) return null;
    return result;
}
```

Add a Component Diagram type

6. Add a context menu for opening Component Diagram
   - Commands.vsct
   - Commands.cs
   - CommandSet.cs

7. Add to OpenView in DocData.cs

8. Transform, run and test the multi-view example for the integrated Component Diagram