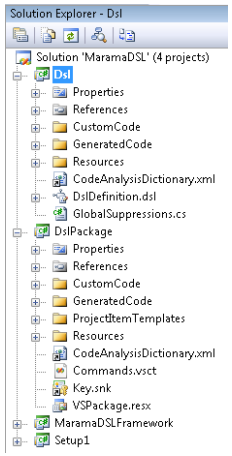
Microsoft DSL Tools

- **Aim of section:**
  - Examine Microsoft DSL Tools, yet another meta tool for constructing DSVL environments
    - As a Visual Studio extension
- **Contents**
  - DSL Tools architecture
  - Tools/notations for domain model elements, shapes and connectors, and diagram element maps
  - Behaviour implementation
  - Multiple views hacking
  - Assignment

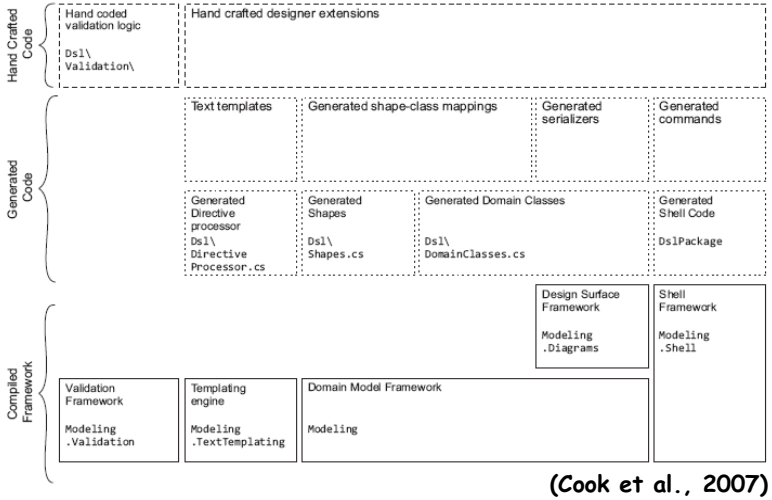
References:
1. Cook et al., Domain-specific development with Visual Studio DSL tools, Addison-Wesley, c2007.
2. Getting Started with DSL Tools
http://msdn.microsoft.com/en-us/vsx/cc677260.aspx

DSL Solution

- Wizard and template-based creation
- Dsl project
  - Visual Dsl Designer
  - Generated model and diagram element classes, multiplicity (hard constraint) validation, serialiser, template directive processor, editor helpers...
- DslPackage project
  - Coupling to Visual Studio
  - Generated model and view handling code, context menu commands, model explorer ...
- Generated code allows customisation (more later)
- Run/debug in "experimental hive"



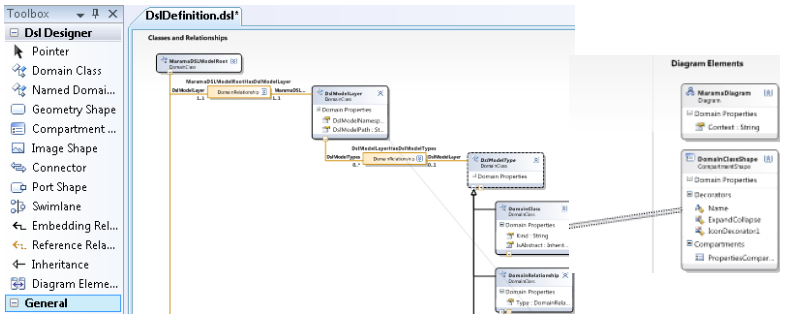
Architecture of the DSL Tools



(Cook et al., 2007)

Meta modelling

- One integrated DSL Designer (DslDefinition.dsl) for
  - Meta model (termed "Domain Model" / "Classes and Relationships" in DSL Tools)
  - Visual notation (termed "Diagram Elements")
  - Model-to-diagram element maps (termed "Diagram Element Map")



Domain model

- Domain Class
  - Named Domain Class
- Domain Relationship
  - Embedding
  - Reference
- Inheritance

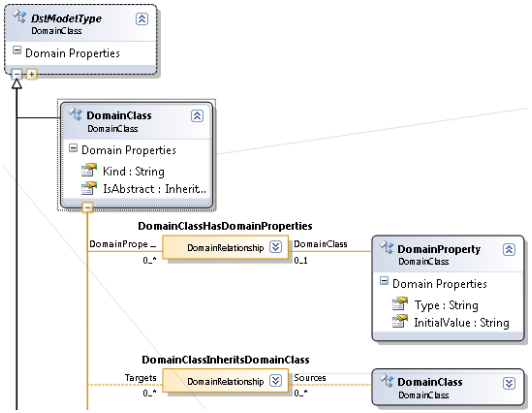


Diagram elements

- Shape
  - Geometry shape
  - Compartment shape
  - Image shape
  - Port shape
  - Swimlane
- Connector
- Properties window based edits

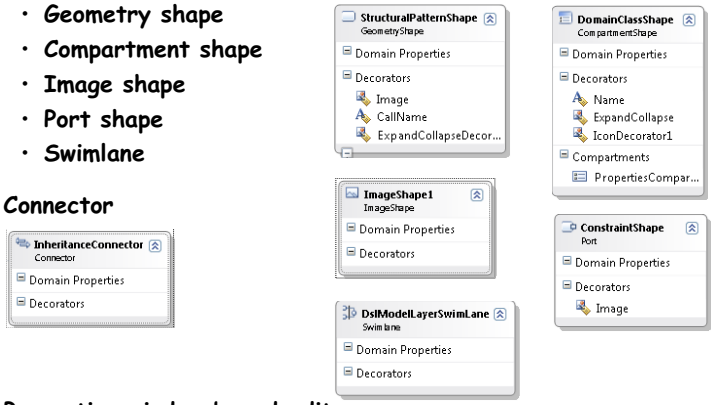


Diagram element maps

- Diagram element to domain model element maps
- Diagram element property to domain model element property maps
- Compartment maps (if compartment shape)

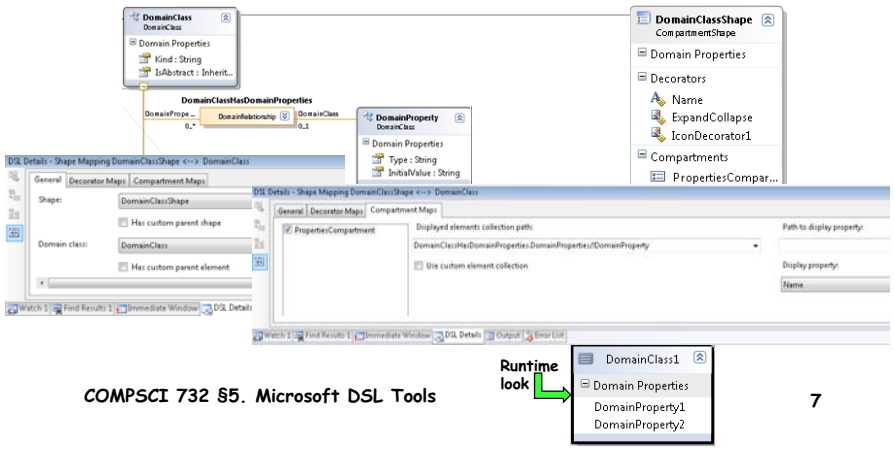
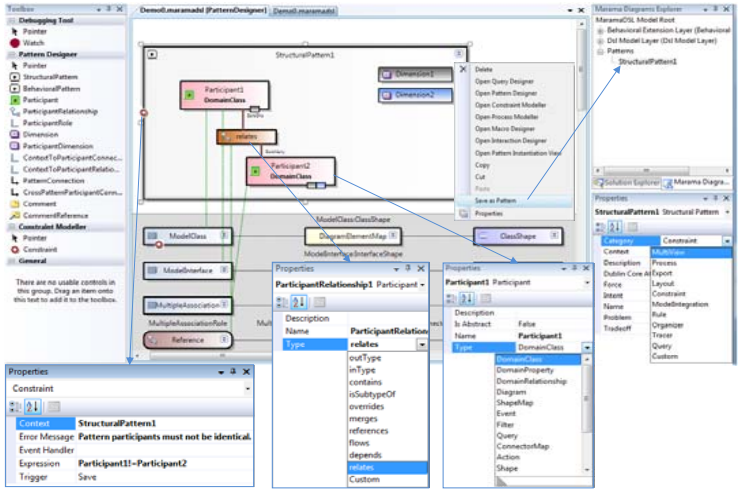


Diagram and editor

- Diagram
  - Only one! Need customisation to allow multi-views!
- Editor (also called "Generated Designer")
  - Design surface - links to diagram
  - Toolbox tabs and items
  - Properties window
  - Menu commands
  - Tree-structured, customisable Model Explorer – very COOL!
- XML serialisation (separated model and diagram files in domain-specific XML format) for saving and loading
  - Domain concepts – in a .yourdslextension file
  - Shapes, connectors and layout – in a .yourdslextension.diagram file

Generated designer example



Behaviours in generated designer

- Custom code, using Modelling and Diagrams etc APIs in VS environment
  - Write partial classes to override/integrate generated code/behaviour
- Handling changes in a model
  - E.g. maintaining consistency, calculating dependent value, refactoring models, updating visualisations, raising an exception, disallow an attempted change, propagate a change through a model, generate code/artefact ...
  - Want (semi-) automation
- Techniques
  - Validation framework
  - Rules
  - Events and overridable methods

Validation framework

- Microsoft.VisualStudio.Modelling.Validation
  - Validation methods over model elements and links
  - Creating error objects when validation fails, posted in Visual Studio errors window

```
using ...

namespace UoA.marama.metadsl.Validation
{
    [ValidationState(ValidationState.Enabled)]
    public partial class ModelClass
    {
        [ValidationMethod(ValidationCategories.Open | ValidationCategories.Save | ValidationCategories.Menu)]
        public void Constraint0(ValidationContext context)
        {
            if (context == null) throw new global::System.ArgumentNullException("context");
            if (!(this != this.Superclass))
            {
                context.LogError("Self Inheritance Not Allowed", "ModelClassError", this);
            }
        }
    }
```


Rules

- Can associate with any domain class, relationship and diagram element
- Can be set to execute when a domain property changes, an instance is added/deleted, and on other custom conditions
- Need to register in Domain Model

```
using ...

namespace UoA.marama.metadsl
{
    [RuleOn(typeof(DomainClass), FireTime = TimeToFire.TopLevelCommit, Priority = DiagramFixupConstants.AddShape
    internal sealed partial class FixUpDiagramOnElementAddedRule : AddRule
    {
        public override void ElementAdded(ElementAddedEventArgs e)
        {
            if (e == null) throw new ArgumentNullException("e");
            ModelElement childElement = e.ModelElement;
            if (childElement.IsDeleted)
                return;

            ModelElement parentElement;
            parentElement = GetParentForDomainClass(childElement);
            if (parentElement != null)
            {
                Diagram.FixUpDiagram(parentElement, childElement);
            }
        }
```

- AddRule
- ChangeRule
- DeletingRule
- RolePlayerChangedRule
- ...
- RuleOn attribute
- Argument yielding change details

Events and overridable methods

- Domain classes
  - OnDeleted, OnDeleting ...
- Shape
  - OnDoubleClick, Collapse(), Expand() ...
- Property handlers
  - OnValueChanged, OnValueChanging ...
- Override these methods in a partial class to change the default design behaviours

```
using ...

namespace UoA.marama.metadsl
{
    #region Collapse/Expand size of StructuralPatternShape
    partial class StructuralPatternShape
    {
        protected RectangleD ExpandedBounds;

        protected override void Collapse()
        {
            base.Collapse();
            this.ExpandedBounds = this.Bounds;
            this.Bounds = this.AbsoluteBounds;
            this.AbsoluteBounds = new RectangleD(this.Location, new SizeD(1.6, 0.3));
        }

        protected override void Expand()
        {
            base.Expand();
            this.Bounds = this.ExpandedBounds;
        }

        public override bool AllowsChildrenToResizeParent...

        public override SizeD MinimumResizableSize...
    }
    #endregion
```


In-memory store

- Stores runtime state of a generated designer
- Gives runtime access for:
  - Looking up domain model elements
  - Creating, updating and deleting model elements and links
  - Transactions, undo/redo
  - Firing rules and events (executed within the transaction in which the change occurred)

Programming with the store

SELECT

```
public static IList<ModelElement> SelectAllModelElements(ModelElement root)
{
    IList<ModelElement> resultSet = new List<ModelElement>();
    foreach (var element in root.Store.ElementDirectory.AllElements)
    {
        IList<DomainRoleInfo> domainRoles = element.GetDomainClass().LocalDomainRolesPlayed;
        foreach (var domainRoleInfo in domainRoles)
        {
            if (!domainRoleInfo.IsEmbedding)
            {
                if (!resultSet.Contains(element)) resultSet.Add(element);
            }
        }
    }
    return resultSet;
}
```

INSERT

```
public static ModelElement InsertModelElement(ModelElement root, Type elementType)
{
    Store store = root.Store;
    ModelElement element = null;

    using (Transaction t = store.TransactionManager.BeginTransaction("Create element"))
    {
        DomainClassInfo dci = store.DomainDataDirectory.FindDomainClass(elementType);
        element = store.ElementFactory.CreateElement(dci.Id);
        ElementOperations elementOperations = new ElementOperations(store as IServiceProvider, store);
        ElementGroup elementGroup = new ElementGroup(store);
        elementGroup.Add(element);
        elementGroup.MarkAsRoot(element);
        elementOperations.MergeElementGroup(root, elementGroup);
        t.Commit();
    }
    return element;
}
```


Programming with the store

UPDATE

```
public static void SetProperty(ModelElement element, String propertyName, object value)
{
    if (element == null || propertyName == null || value == null) return;
    using (Transaction t = element.Store.TransactionManager.BeginTransaction("Update property value"))
    {
        if (element.GetType().GetProperty(propertyName).CanWrite)
        {
            element.GetType().GetProperty(propertyName).SetValue(element, value, null);
        }
        t.Commit();
    }
}
```

DELETE

```
public static void DeleteModelElement(ModelElement element)
{
    if (element == null) return;
    if (!element.Store.InUndoRedoOrRollback)
    {
        using (Transaction t = element.Store.TransactionManager.BeginTransaction("delete model element"))
        {
            element.Delete();
            t.Commit();
        }
    }
}
```

Code/Artefact generator

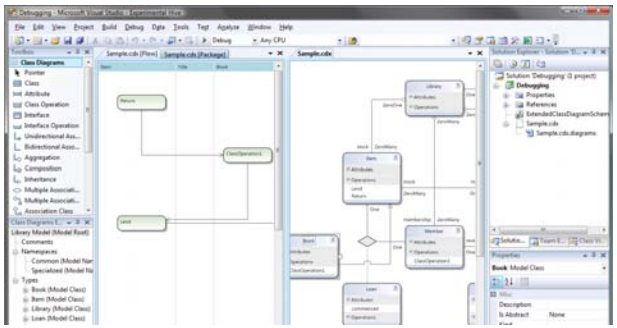
- Traverse over model and map model elements to code modules
- Text template
  - Directives in <#@ ... #>
  - Code-gen statements enclosed in <# ... #>
  - Expressions enclosed in <=# ... #>
  - Transform
  - Can output any kind of file
- T4 engine (cf Jet in Eclipse)

```
<#@ template inherits="Microsoft.VisualStudio.TextTemplating.VSHost.ModelingTextTransformatio
<#@ MaramaDSL processor="MaramaDSLDirectiveProcessor" requires="fileName='%MODELFILENAME%'" #
using System;
using System.Collections.Generic;
using System.Text;
using System.Collections.ObjectModel;
using Microsoft.VisualStudio.Modeling.Validation;

namespace UoA.marama.metadsl.Validation
{
<#
    foreach(DslModelType type in this.MaramaDSLModelRoot.DslModelLayer.DslModelTypes)
    {
#>
<#
        if (type.Constraints != null)
        {
#>
    [ValidationState(ValidationState.Enabled)]
    public partial class <#= type.Name #>
    {
<#
            int count = 0;
            foreach(Constraint constraint in type.Constraints)
            {
#>
        [ValidationMethod(ValidationCategories.Open | ValidationCategories.Save | ValidationC
        public void Constraint<#= count #>(ValidationContext context)
        {
            if (context == null) throw new global::System.ArgumentNullException("context");
            if (!(<#= constraint.Expression #>))
            {
                context.LogError("<#= constraint.ErrorMessage #>", "<#= type.Name+"Error" #>",
            }
        }
<#
                count++;
            }
```


Multi-views

- Microsoft DSL Tools does not support modelling multiple views in its current version (v2008)



- Community contributions
  - http://www.netfxfactory.org/blogs/papers/archive/2009/01/13/multiply-dsl-points-of-view.aspx

Multi-views

- Customisation in DslPackage project
  - DocData
  - DocView
  - EditorFactory
  - File extension
  - ...
- Customisation in Dsl project
  - Diagrams
  - Rules
  - Serialisation
  - ...

Assignment #1

- Team assignment, due Friday 23rd April, 25%
- Development of a simple multiple view, visual software development support tool using a meta tool technology – your choice of Marama or MS DSL Tools
- Deliverables:
  - Collaborated tool (10%)
  - Individual report (10%)
  - Collaborated evaluation form and peer team evaluation (5%)

Assignment #1 Marking Schedule

		Marks
Prototype		
	Appropriate domain meta-model constructs and their associations (/5)	
	Appropriate shapes and connectors (/5)	
	At least two DIFFERENT view types (/5)	
	At least three dynamic behaviors (/10)	
	Appropriate model example (/5)	
	Flair in visual language design/creativity (/10)	
	Subtotal Prototype (/40)	0
Report		
	Motivation (/5)	
	Description of tool features (/10)	
	Example illustration using screen dumps (/5)	
	Description of how you built the tool (/5)	
	Assessment of your tool (/5)	
	Assessment of the meta-tool's suitability and suggestions for improvements (/5)	
	Report organisation and references used (/5)	
	Subtotal Report (/40)	0
Peer Evaluation		
	Appropriate evaluation form (/10)	
	Peer evaluation result (/10)	
	Subtotal Peer Evaluation (/20)	0
	Total Overall (/100)	0
	Total Overall (/25)	0