devdefined

Model Driven Development

With Enterprise Architect

Presented By Alex Henderson

Introductions

- Senior developer and managing director of a Boutique software development company (DevDefined Limited) based in Auckland

- Active participant in the New Zealand .Net Development community, presenting at user groups, code camps etc.

- Organiser of the Auckland “Architecture Chat” group – which hosts fortnightly forums to discuss architectural and technical topics affecting .Net and the wider development community.

- Keen technical blogger and open source contributor.

What am I talking about?

- Model Driven Development basics
- MDA Transformations in EA
- MDG Integration for Visual Studio .Net
- Real-world MDD
- Conclusions
- Questions

Overview

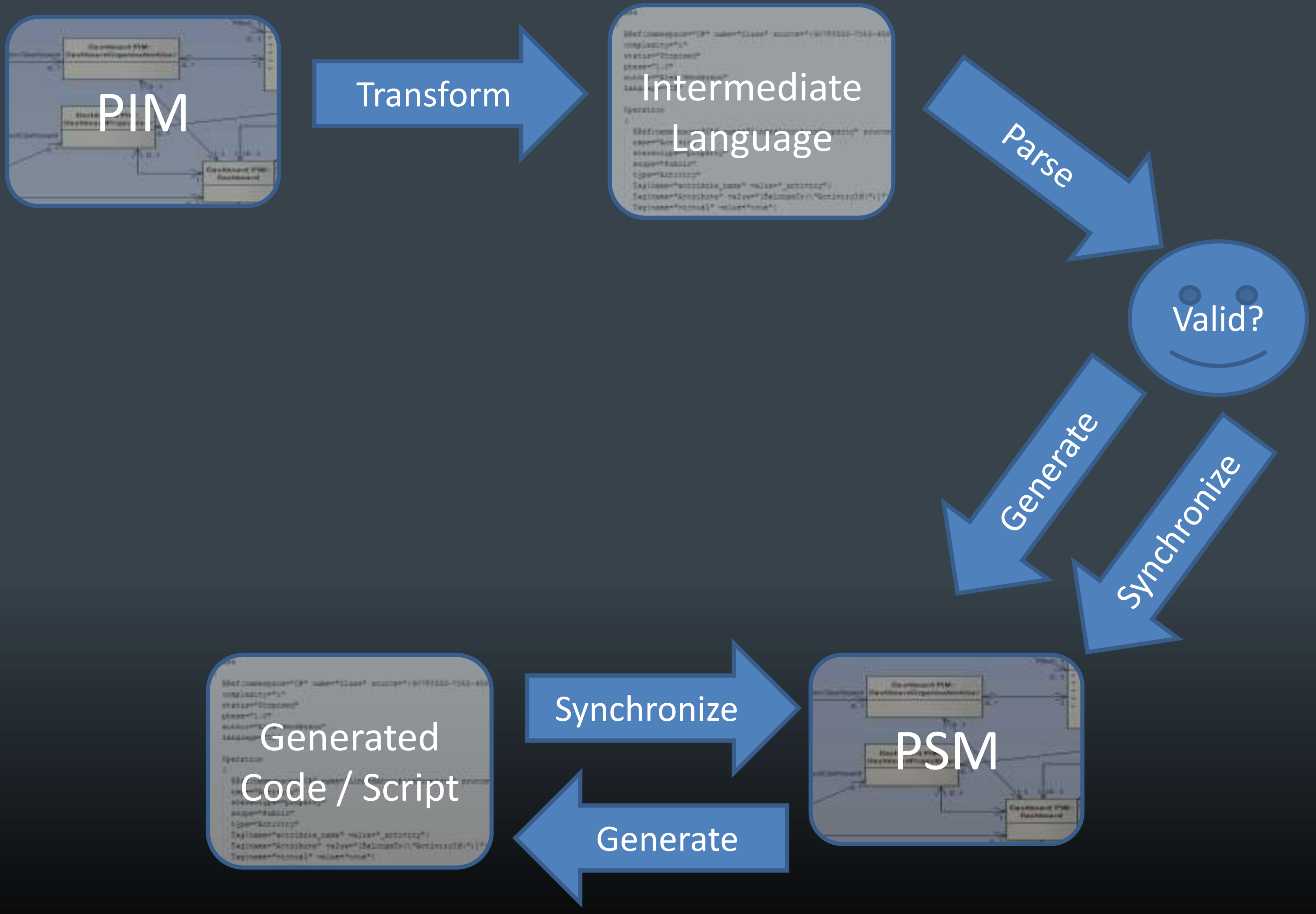
• Model driven development in a nut shell is using models (and in this case UML models) to forward engineer into code.

• The dream of MDD is to decouple the design, realizing the functional requirements of a system through a platform independent model (**PIM**), while implementing the system using a set of platform specific models (**PSM**’s) - allowing the underlying technology of the system to change while the PIM remains the same.

• Within Enterprise Architect A PIM is <u>transformed</u> into a PSM via an “MDA Transformation” – which is in effect a simple templating language (the same one used for code generation in EA).

devdefined

Overview



MDA Transformations

• MDA Transforms are used to read a model (normally Platform independent) and to create one or more resulting models (which are normally Platform specific).

• MDA Transformations in EA share the same scripting language as code generation.

• The intermediate output of an MDA transformation is a single text file, with all the packages, classes etc. Declared in a simple hierarchical format.

• EA reads the intermediate text output, translates it into the corresponding elements and attributes and either creates new elements or attempts to synchronize changes to existing elements (which may have been created as the result of a previous transform run).

devdefined

MDA Transformations

```
Package
{
  name="C# Model"
  namespaceroot="true"

  Class
  {
    Xref { namespace="C#" name="Class" source="{907F0550...}" }
    complexity="1"
    status="Proposed"
    author="Alex Henderson"
    language="C#"

    Operation
    {
      Xref { namespace="C#" name="LinkedAttributeProperty" source="{907F0550-...}" }
      name="Activity"
      notes="An Activity"
      stereotype="property"
      scope="Public"
      type="Activity"
      Tag { name="attribute_name" value="_activity" }
      Tag { name="Attribute" value="[BelongsTo(\"ActivityId\")]" }
      Tag { name="virtual" value="true" }
    }
  }
}
```

This is the objectType i.e. Package, Class etc.

This is an objectProperty – often these have a direct counterpart in the properties dialog within EA.

XRef provides the mechanism for forward synchronizing changes from your PIM to your PSM.

Tag is an element (basically a property with child properties) – other common elements include Attribute, Parameter or any valid objectType.

Backslash is used to escape characters, such as the double quote within a value string.

devdefined

MDA Transformations

```
%elemType%
{
  %TRANSFORM_REFERENCE("Class")%
  %TRANSFORM_CURRENT("language", "name")%
  name=%qt%$name%qt%
  language="C#"
  %list="ClassBase" @separator="\n" @indent="  "%
  %list="ClassInterface" @separator="\n" @indent="  "%
  %list="InnerClass" @separator="\n" @indent="  "%
  %if elementType =="Class" and classStereotype == "picklist"%
  %list="Attribute" @separator="\n"  attStereotype != "enum"%
  %else%
  %if elemType == "Class"%
  %list="Attribute" @separator="\n" @indent="  "%
  %endIf%
  %endIf%
  %if elemType == "Class" and classStereotype != "enumeration"%
  Tag
  {
    name="partial"
    value="true"
  }
```

This will output the contents of the variable “elemType”.

Outputs all the properties of the current class excluding the list of properties supplied...

List calls another template, passing a list of elements...

Support exists for basic if/else statements.

Anything not inside %quotes% is output “as-is”

MDA Transformations – Object Types

- Action
- ActionPin
- Activity
- ActivityParameter
- ActivityPartition
- ActivityRegion
- Actor
- Association
- Change
- Class
- Collaboration
- CollaborationOccurrence
- Component
- DeploymentSpecification
- DiagramFrame
- Decision
- EntryPoint
- Event
- ExitPoint
- ExceptionHandler
- ExpansionNode
- ExpansionRegion
- ExposedInterface
- GUIElement
- InteractionFragment
- InteractionOccurrence
- InteractionState
- Interface
- InterruptibleActivityRegion
- Issue
- Iteration
- Object
- ObjectNode
- MessageEndpoint
- Node
- Package
- Parameter
- Part
- Port
- ProvidedInterface
- RequiredInterface
- Requirement
- Sequence
- State
- StateNode
- Synchronization
- Table
- TimeLine
- UMLDiagram
- UseCase

There are a lot of object types, but in most cases you will only need a few to handle the basic scenarios!

devdefined

MDA Transformations – Object Properties & Elements

Properties

- Abstract
- Alias
- Arguments
- Author
- Cardinality
- Classifier
- Complexity
- Concurrency
- Filename
- Header
- Import
- IsActive
- IsLeaf
- IsRoot
- IsSpecification
- Keyword
- Language
- Multiplicity
- Name
- Notes
- Persistence
- Phase
- Scope
- Status
- Stereotype
- Version
- Visibility

Elements

- Attribute
- Classifier
- Parameter
- Operation
- Parent
- Tag
- XRef

This is the list of documented properties – there are more, but they're undocumented – some can be discovered by reviewing the source for the "out of the box" MDA templates or searching the EA forums.

Real world MDD

MDG Integration with Visual Studio .Net.

A quick look at what it does and does not do...

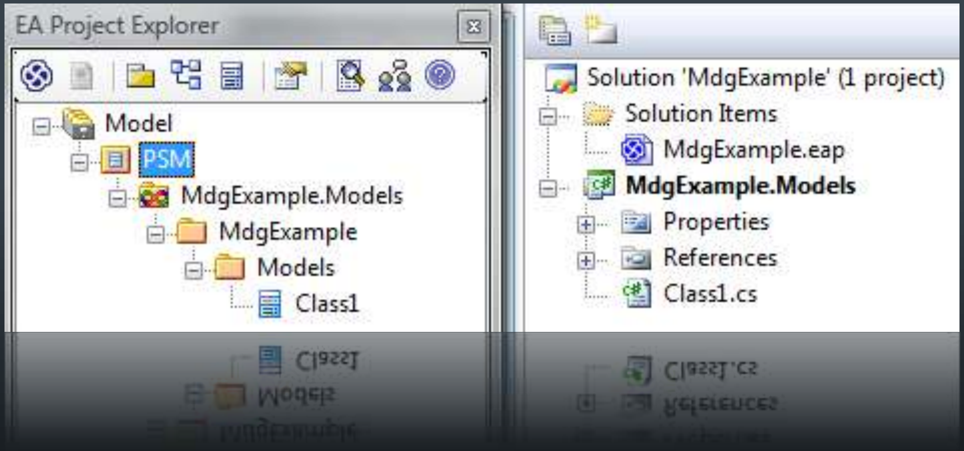
MDG Integration for Visual Studio 2008

- Allows developers to avoid having to leave the confines of the Visual Studio IDE to achieve functions such as:
  - Basic model changes i.e. Element properties.
  - Version control
  - Code Generation & MDA Transformations
  - Generate HTML & RTF Documentation.
  - Navigation between code and model and back again.
- Makes generating and synchronizing code for a single class very easy.
- Information about the model is generally displayed as HTML served under the hood by an inbuilt web server within EA.
- Import & Synchronization of Team Foundation Server (TFS) work items.

devdefined

MDG Integration for Visual Studio 2008

- Users connect projects in Visual Studio to packages in Enterprise Architect.
- These packages would be the target package used when applying your language specific MDA transforms i.e. To VB.Net, C++ or C#.
- Packages linked to Visual Studio Projects are designated by the Visual Studio infinite loop icon.



devdefined

MDG Integration for Visual Studio 2008

- MDG integration is just that – <u>integration</u> – it doesn't provide any functionality that isn't already part of Enterprise Architect, it just makes it more convenient for a developer to use.

- In some ways it's too crippled for a developer who is responsible for maintaining the model – for example you can't delete packages or Elements without launching EA, and worse yet you can not view Tagged values for an element – which are often vital for Platform independent models.

- Hampered by the same limitations as EA when synchronizing between code and models i.e. Refactoring code will break your model (for example renaming a class in code will result in your model having two classes, one with the old name, one with the new).

devdefined

Real world MDD

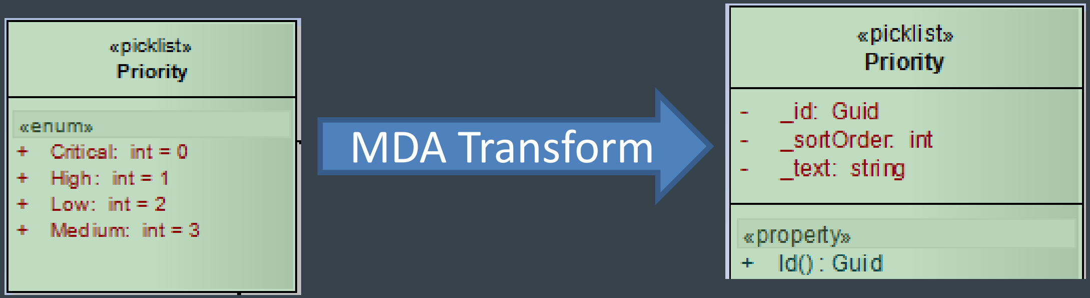
MDA is often discussed in high level terms – but let’s bring it back down to a practical level, by looking at some real examples of transformations.

devdefined

Real world MDD

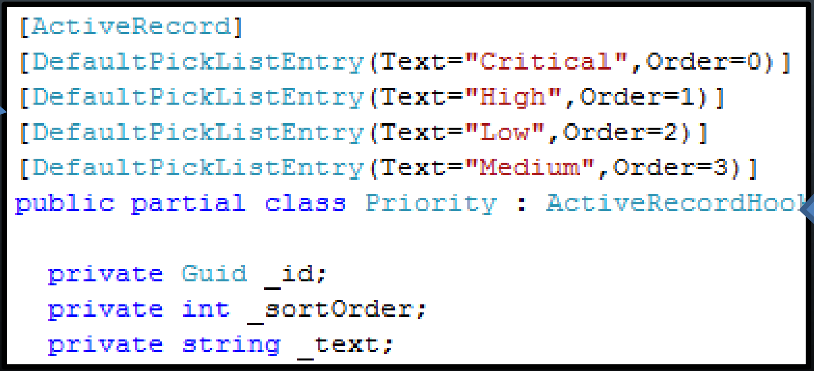
Scenario 1 – pick lists / lookup tables.

Often enumeration classes are used to model pick lists or lookup tables – problem here is that in some cases an enumeration class should be an enumeration in code as well (if for example the value isn't persisted, or the set of values / labels can not be changed).



Enumeration members become declarative attributes in code (which can be examined to generate data when application starts up for the first time...

Code Generation



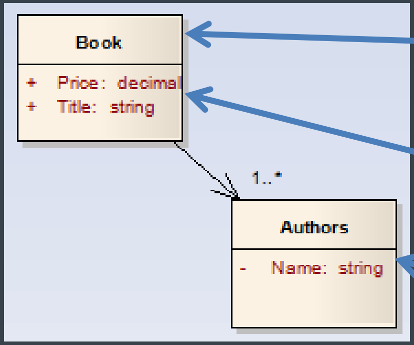
```
[ActiveRecord]
[DefaultPickListEntry(Text="Critical",Order=0)]
[DefaultPickListEntry(Text="High",Order=1)]
[DefaultPickListEntry(Text="Low",Order=2)]
[DefaultPickListEntry(Text="Medium",Order=3)]
public partial class Priority : ActiveRecordHoo

  private Guid _id;
  private int _sortOrder;
  private string _text;
```

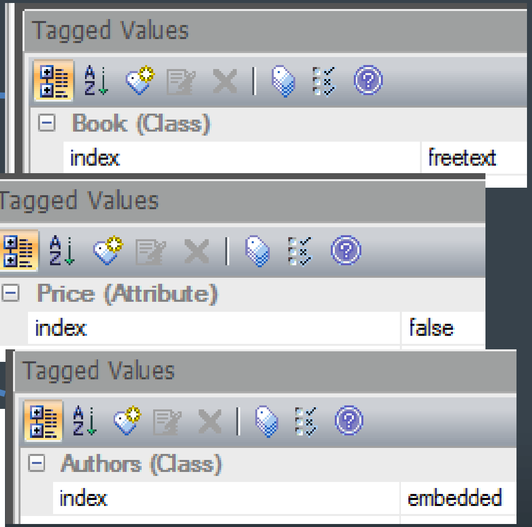
devdefined

Real world MDD

Scenario 2 – free text searching

Relational databases are not great candidates for free-text searching, but there exists search engines (such as Lucene for Java or Lucene.Net for the .Net Framework) which excel at “google” style document searching. But indexing all your entities is often too granular / risky.





Tagged Values

Book (Class)	
index	freetext

Tagged Values

Price (Attribute)	
index	false

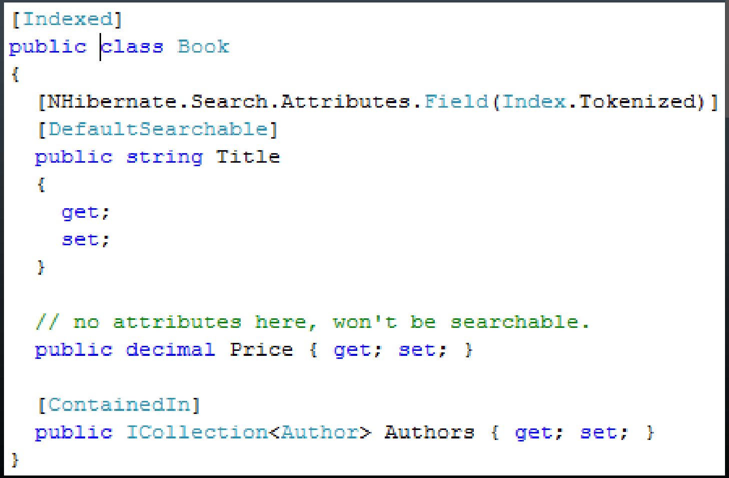
Tagged Values

Authors (Class)	
index	embedded

Make this entity searchable.

Make price unsearchable.

Embed the authors entity in the book entity for searching purposes (denormalize)

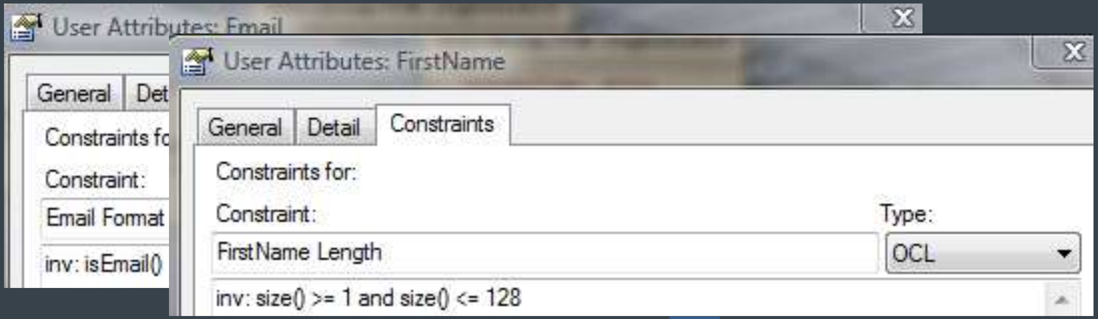


```
[Indexed]
public class Book
{
  [NHibernate.Search.Attributes.Field(Index.Tokenized)]
  [DefaultSearchable]
  public string Title
  {
    get;
    set;
  }

  // no attributes here, won't be searchable.
  public decimal Price { get; set; }

  [ContainedIn]
  public ICollection<Author> Authors { get; set; }
}
```

Real world MDD

Scenario 3 – validation framework

Validation frameworks exist for most languages – allowing entities to be validated at point of saving against business rules etc.


User Attributes: Email

General | Det

Constraints fo

Constraint:

Email Format

inv: isEmail()

User Attributes: FirstName

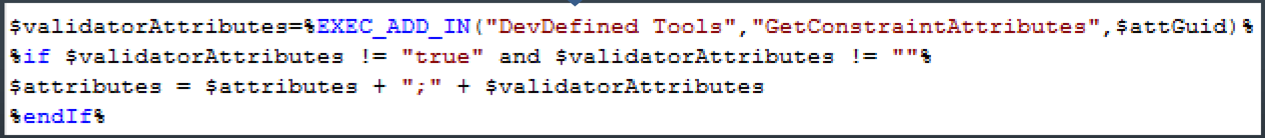
General | Detail | Constraints

Constraints for:

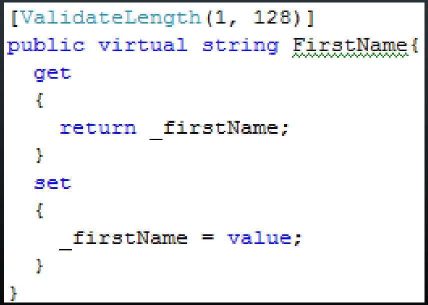
Constraint: FirstName Length

Type: OCL

inv: size() >= 1 and size() <= 128


```
$validatorAttributes=%EXEC_ADD_IN("DevDefined Tools","GetConstraintAttributes",$attGuid)%
%if $validatorAttributes != "true" and $validatorAttributes != ""%
$attributes = $attributes + ";" + $validatorAttributes
%endIf%
```

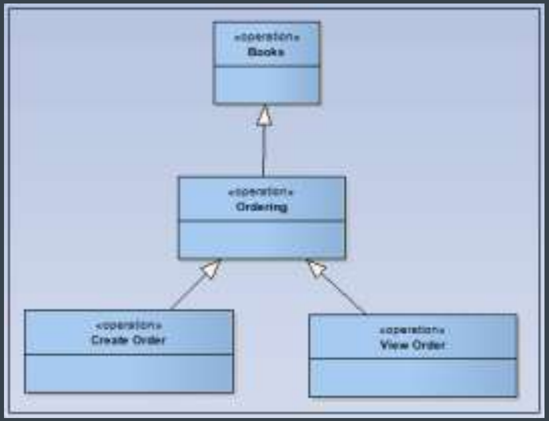
Using OCL we define simple constraints, then Use a custom developed add-in to parse the constraints to create the associated attribute declarations.

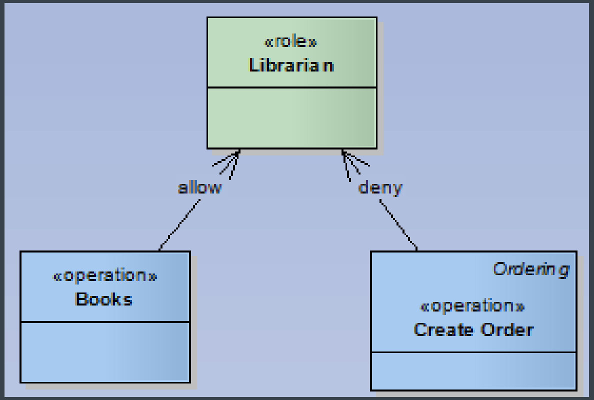

```
[ValidateLength(1, 128)]
public virtual string FirstName{
  get
  {
    return _firstName;
  }
  set
  {
    _firstName = value;
  }
}
```

Real world MDD

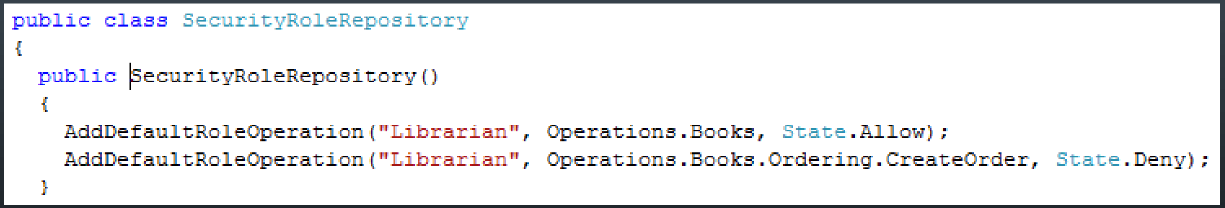
Scenario 4 – security

Role based security is common but more often then not each role represents a set of operations...

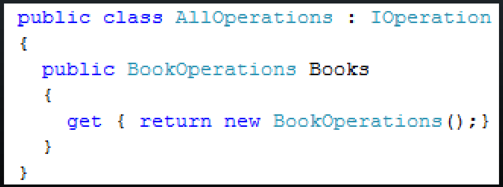




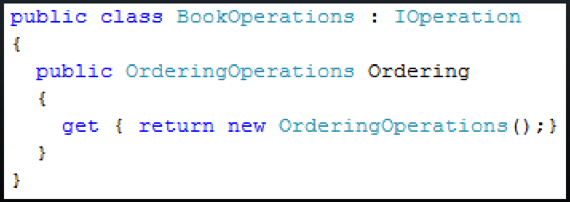
Librarian role can access all book operations, Except for "Create Order" which is explicitly Denied to them.



```
public class SecurityRoleRepository
{
  public SecurityRoleRepository()
  {
    AddDefaultRoleOperation("Librarian", Operations.Books, State.Allow);
    AddDefaultRoleOperation("Librarian", Operations.Books.Ordering.CreateOrder, State.Deny);
  }
```



```
public class AllOperations : IOperation
{
  public BookOperations Books
  {
    get { return new BookOperations();}
  }
}
```



```
public class BookOperations : IOperation
{
  public OrderingOperations Ordering
  {
    get { return new OrderingOperations();}
  }
}
```

A security role repository is generated with all the default role to operation mappings. Operations are generated as a set of nested classes – avoid operation names being miss-typed etc. In code.

devdefined

Conclusions

- If you’re working on multiple products/projects that are reasonably similar the benefits of MDA will be greater (i.e. If you’re leaning towards becoming a software factory).

- Don’t underestimate the time it takes to get your templates right – there is no debugging support and the scripts are not pre-compiled – and be prepared to write add-ins to achieve what you need.

- MDG Integration is nice to have but certainly not essential.

- There are a number of companies performing MDA/MDG with EA but so far there hasn’t been much effort put towards sharing templates etc. (partly because the template language leans towards you embedding lots of business logic/business knowledge in your templates, because you can say inherit from a more abstract “generic” template and just override features for your specific project requirements.

Any questions?

Website: http://www.devdefined.com/

Blog: http://www.devdefined.com/blogs.aspx

Email: alex@devdefined.com