Rational. software

Unified Modeling Language (UML) modeling for .NET users

Steve Arnold
Technical Consultant, IBM
steve.arnold@uk.ibm.com

IBM Rational Software Development Conference UK 2007

What keeps me **Rational**?

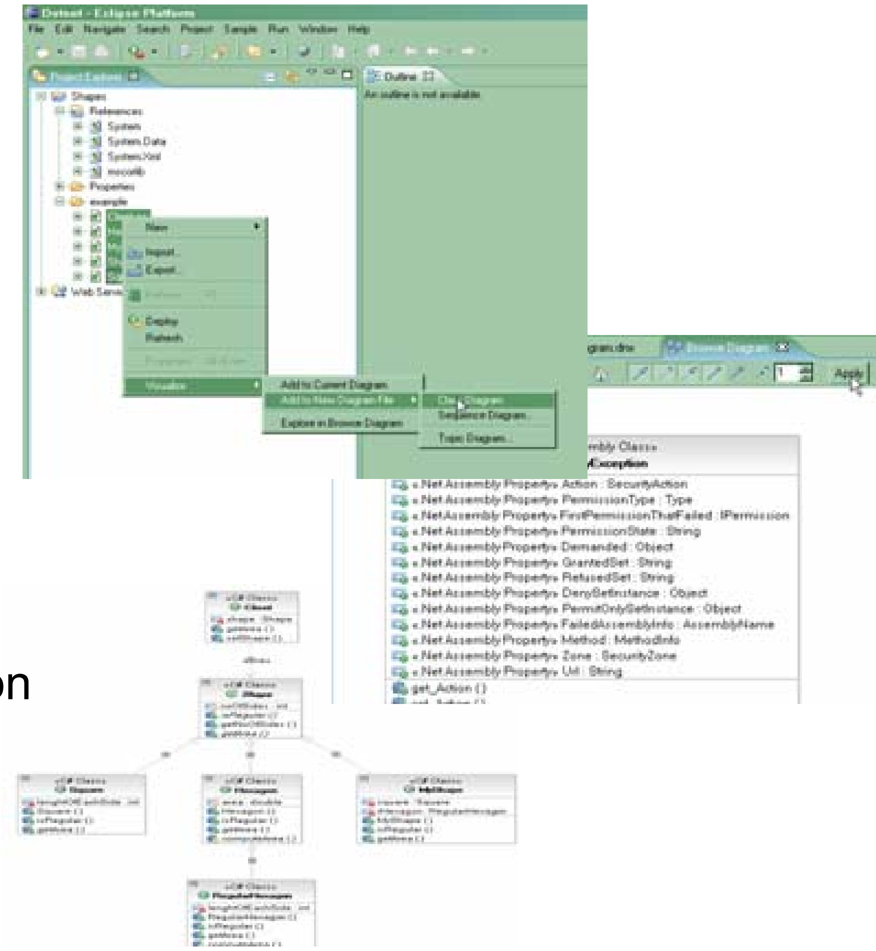
TD4

Agenda

- What is RME .NET?
- MDD theories of operations with RME .NET
- Migrate XDE code models to RME .NET

Rational Modeling Extension for .NET

- Facilitates team communication in heterogeneous environments
  - Enables conceptual modeling of architectures and applications using UML 2
- UML-based model-driven development of applications
  - Whether implemented fully or partially on the Microsoft .NET platform
- Understand your application with C# source visualization
- Evolve design with UML to C# transformation
  - Also C# to UML inverse transformation and reconciliation
  - Supports “True Round Trip Engineering”
- Migration of XDE C# code models



Complements and integrates with Microsoft Visual Studio 2005 technology

Agenda

- What is RME .NET?
- MDD theories of operations with RME .NET
- Migrate XDE code models to RME .NET

MDD Theories of operation

- The theories will be explained as follows:
  - What is the theory
  - When it is useful
  - Ways in which RME .NET supports the theory
  - Demo

MDD Theories of operation

- Concrete model drives development
- Mixed Modeling
- RTE
- Conceptual model drives development

What is a concrete model?

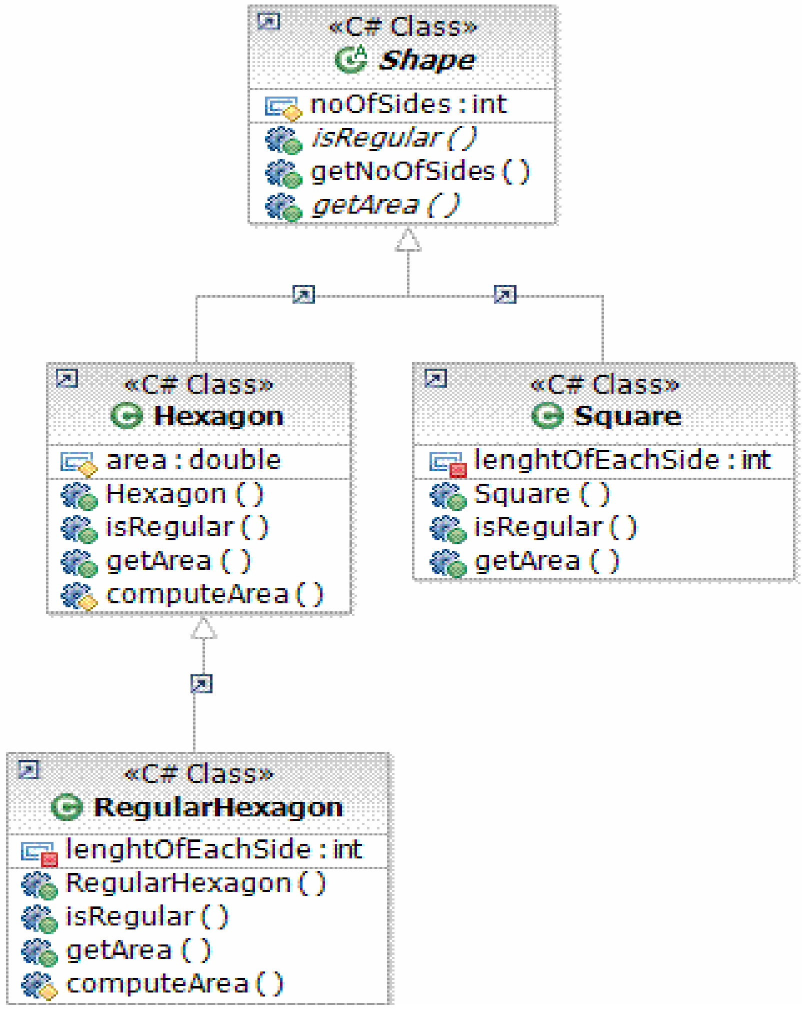
- A domain-specific model implemented on a custom meta-model (in EMF or other technology)
- Code modeling is technology that can depict a piece of code as a UML model. No need to reverse engineer an entire application.

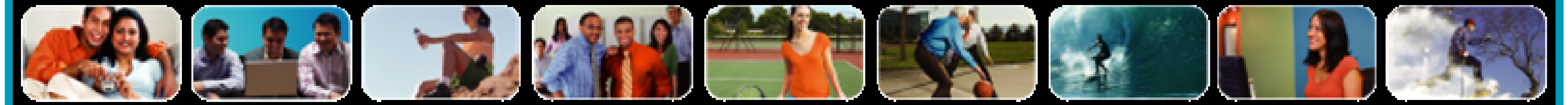
When this is useful

- I’m a developer, I don’t want to have to learn the details of UML or work with model files, but I like the clarity and rigor that diagrams bring to my design process
- To understand code better
- Explore code visually using exploration tools like SRE, browse and topic diagrams

RME .NET complements the designing capabilities available in Visual Studio

- Sequence diagrams
- Show related elements, Show/Hide relationships
- Topic and browse diagrams
- Easy navigability from RME .NET to Visual Studio; by double-clicking
- Harvest code elements (vized) elements





Demo

The .NET Visualizer

MDD Theories of operation

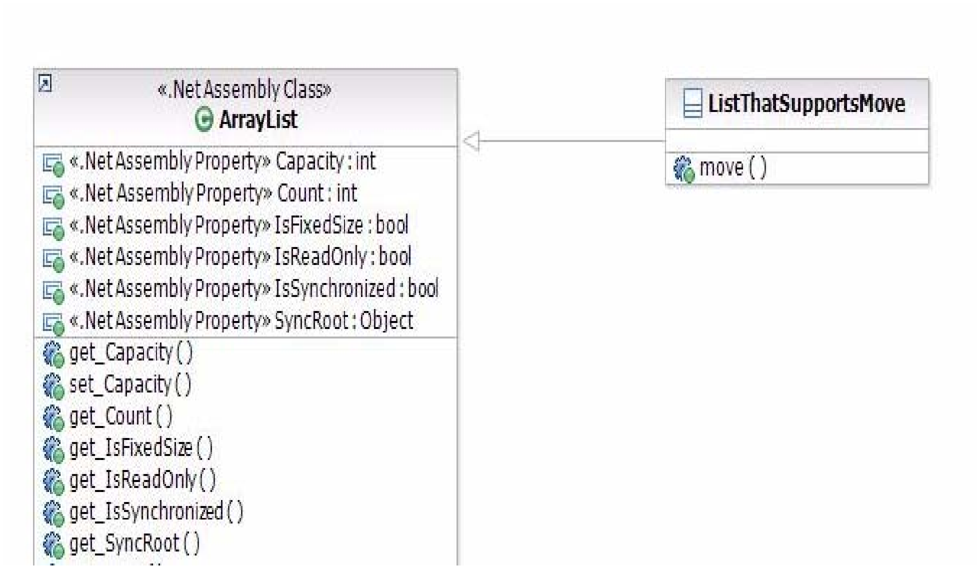
- Concrete model drives development
- Mixed Modeling
- RTE
- Conceptual model drives development

Mixed modeling

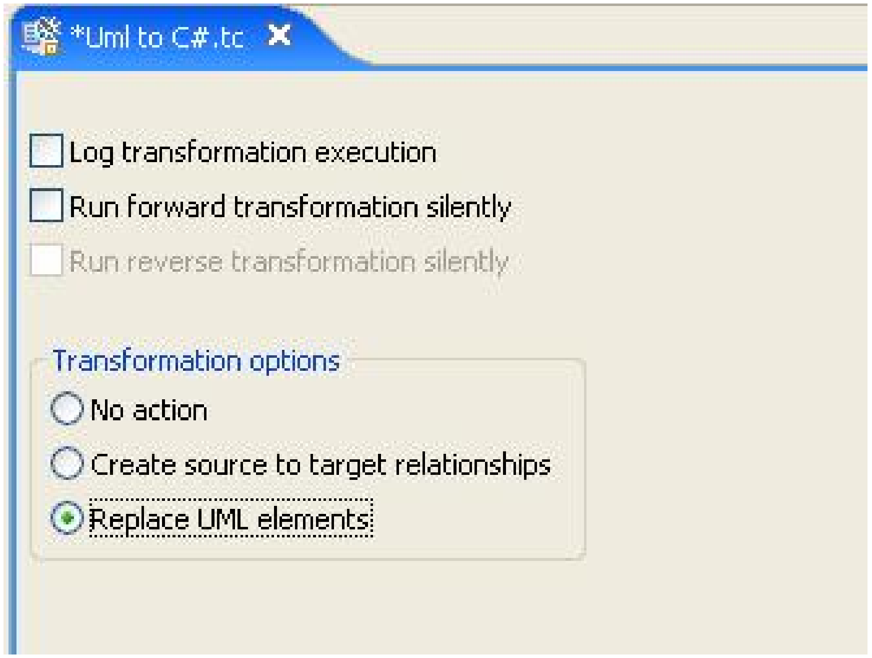
- Conceptual models are “morphed” into mixed models during transformation
- Thereafter, “Code becomes King”. Subsequent changes to code reflect immediately in the diagrams that depict it.
- New conceptual content can be added in subsequent iterations

When this is useful

- If you don't want to maintain separate artifacts, and worry about synchronization between model and code
- Iteratively create new designs in UML and convert them into implementations in a specific domain
- You can use types from .NET Framework and third party assemblies in your models.


«.Net Assembly Class»
ArrayList
«.Net Assembly Property» Capacity : int
«.Net Assembly Property» Count : int
«.Net Assembly Property» IsFixedSize : bool
«.Net Assembly Property» IsReadOnly : bool
«.Net Assembly Property» IsSynchronized : bool
«.Net Assembly Property» SyncRoot : Object
get_Capacity ( )
set_Capacity ( )
get_Count ( )
get_IsFixedSize ( )
get_IsReadOnly ( )
get_IsSynchronized ( )
get_SyncRoot ( )
ListThatSupportsMove
move ( )

Run transformation with “Replace UML elements” option


*Uml to C#.tc

☐ Log transformation execution

☐ Run forward transformation silently

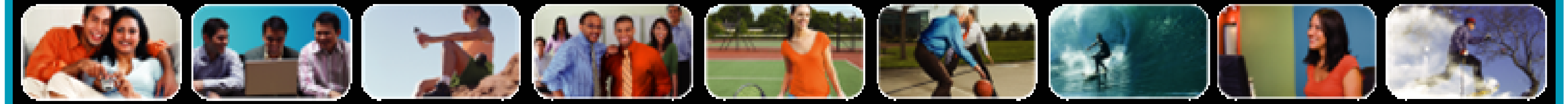
☐ Run reverse transformation silently

Transformation options

○ No action

○ Create source to target relationships

● Replace UML elements



Demo

The C# transform with “Replace Elements Option” (Morph)

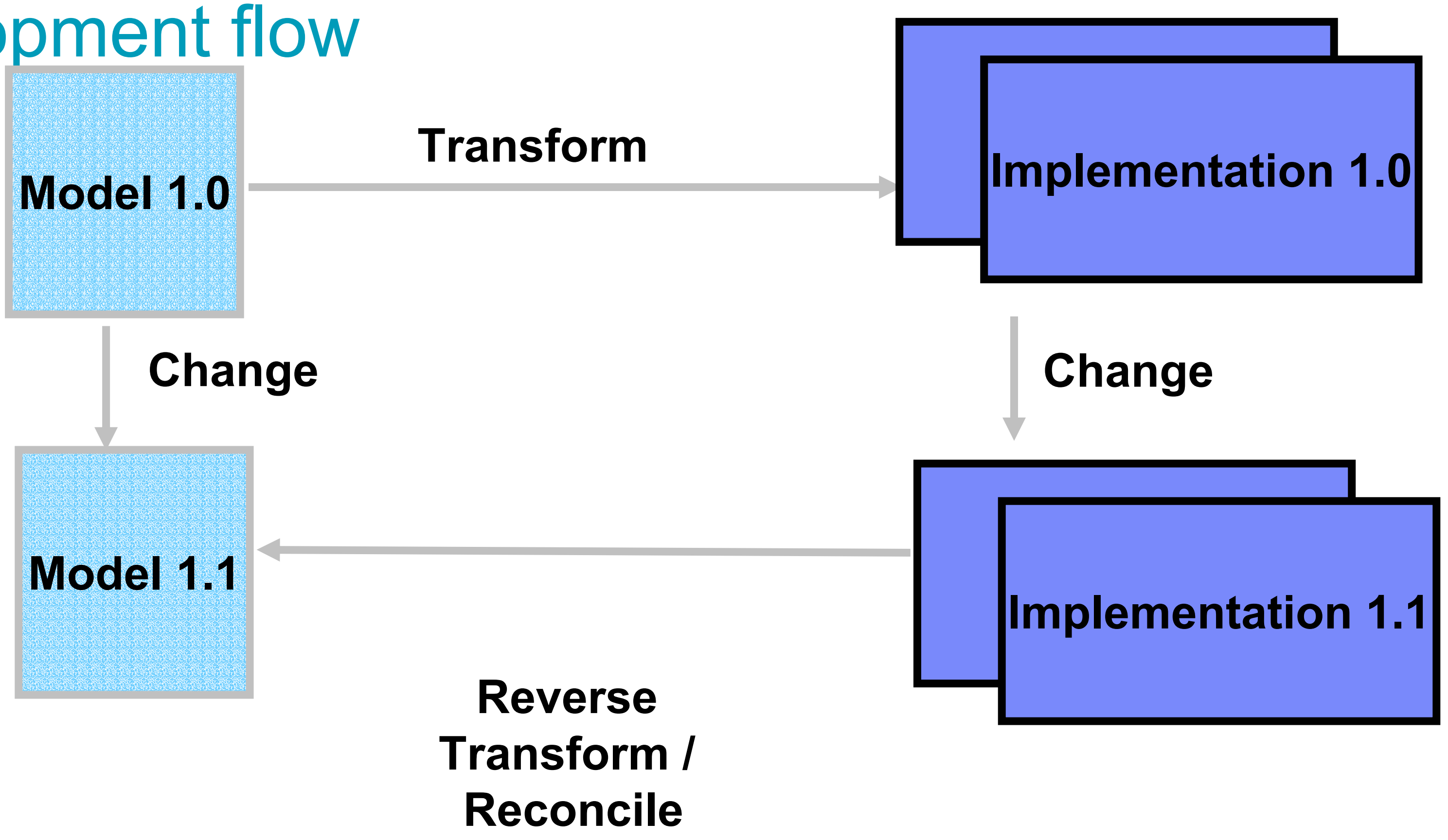
MDD Theories of operation

- Concrete model drives development
- Mixed Modeling
- RTE
- Conceptual model drives development

Round trip engineering

- Create and preserve conceptual models
- Conceptual models and implementations evolve independently after implementation is seeded
- Periodically reconcile conceptual models to implementations

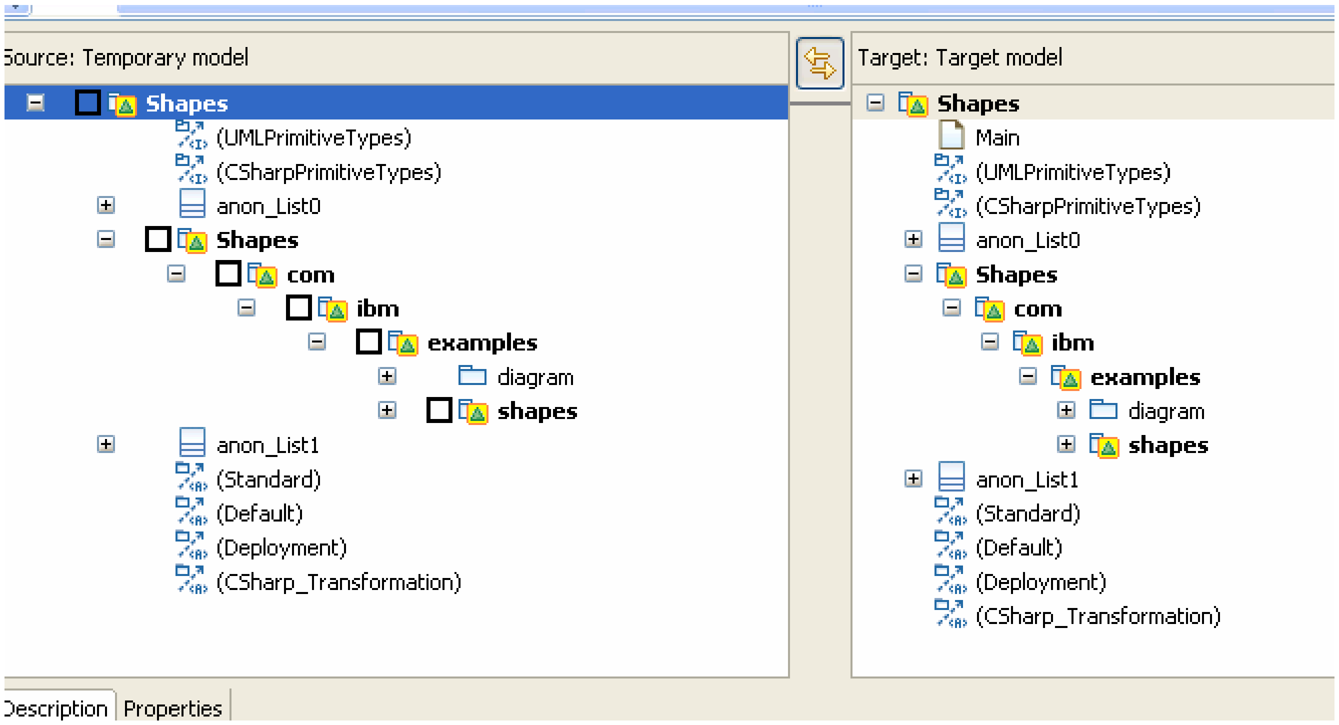
Development flow



When this is useful

- A situation where the architecture and the implementation can evolve independently. Periodically reconcile changes from code to model and resolve issues that require corrective measures.
- Particularly well suited to outsourcing/offshore scenarios

Use fuse framework to merge changes into model



ource: Temporary model

```
Shapes
    (UMLPrimitiveTypes)
    (CSharpPrimitiveTypes)
    anon_List0
    Shapes
        com
            ibm
                examples
                    diagram
                    shapes
    anon_List1
    (Standard)
    (Default)
    (Deployment)
    (CSharp_Transformation)
```

Target: Target model

```
Shapes
    Main
    (UMLPrimitiveTypes)
    (CSharpPrimitiveTypes)
    anon_List0
    Shapes
        com
            ibm
                examples
                    diagram
                    shapes
    anon_List1
    (Standard)
    (Default)
    (Deployment)
    (CSharp_Transformation)
```

Demo

The C# forward and reverse transforms

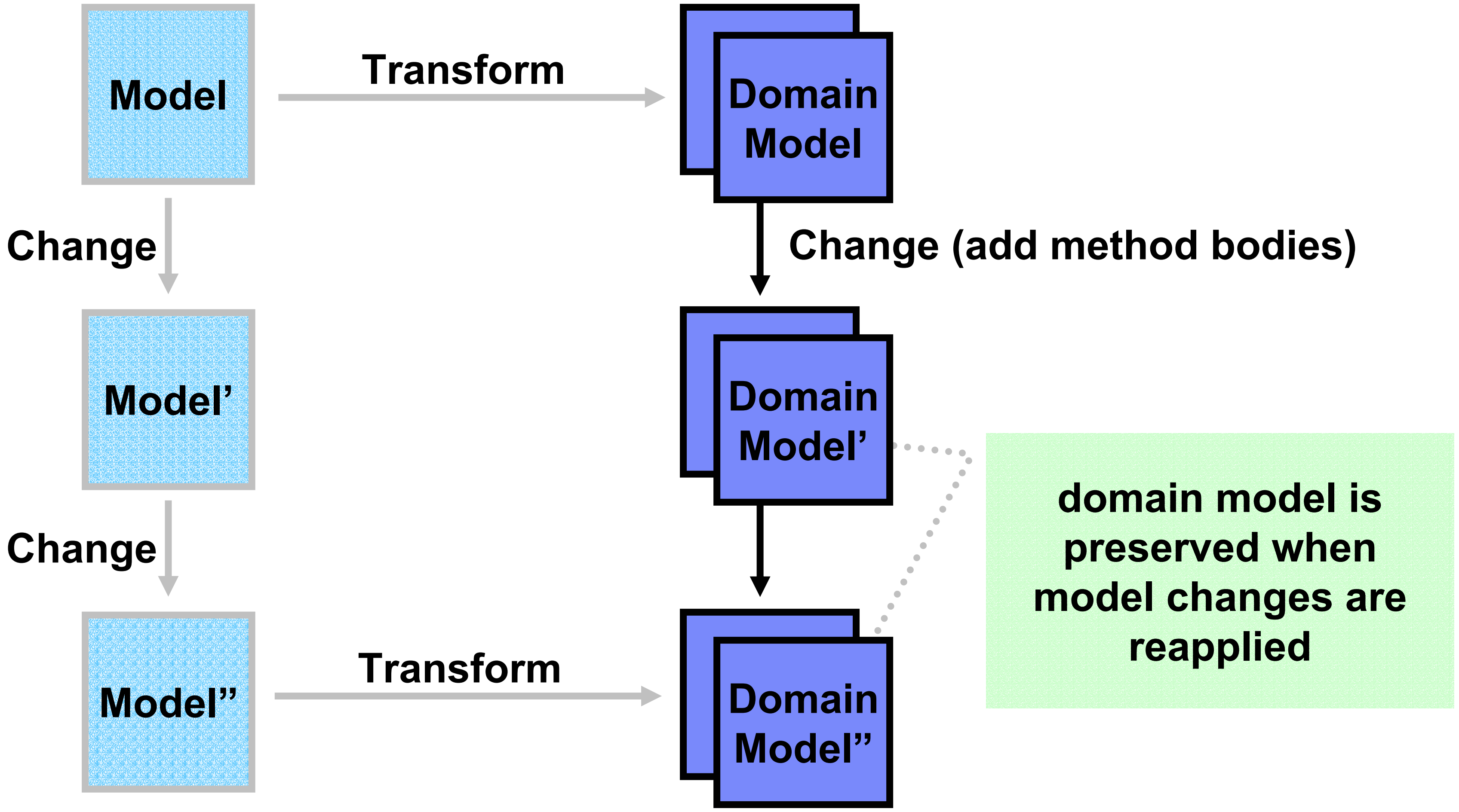
MDD Theories of operation

- Concrete model drives development
- Mixed Modeling
- RTE
- Conceptual model drives development

Conceptual model drives development

- All changes are made in conceptual model and driven (generated) into the implementation.
- Architects have complete control over how the design contract is implemented

Model is master

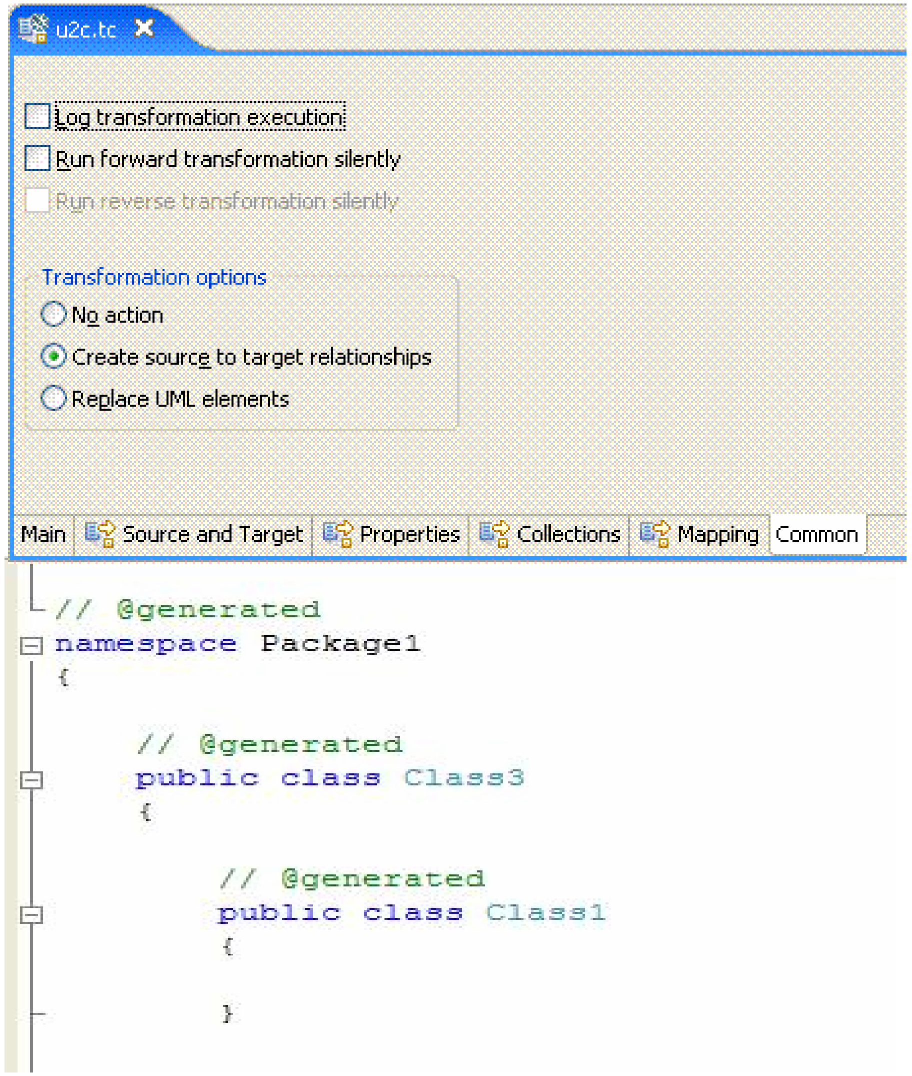


When this is useful

- Clear separation between model and code
- Model is the master but there is no code in the model
- Diagrams, traceability relationships between conceptual and code elements

Generate C# code from UML models

- Create conceptual models in RME .NET using the C# profile.
- Profile provides C# specific stereotypes; for modeling delegates and events, partial classes, generics etc.
- Transformation can be re-applied; preserves code bodies and other user made changes



- [ ] Log transformation execution
- [ ] Run forward transformation silently
- [ ] Run reverse transformation silently

Transformation options
- ○ No action
- [✓ Create source to target relationships]
- ○ Replace UML elements

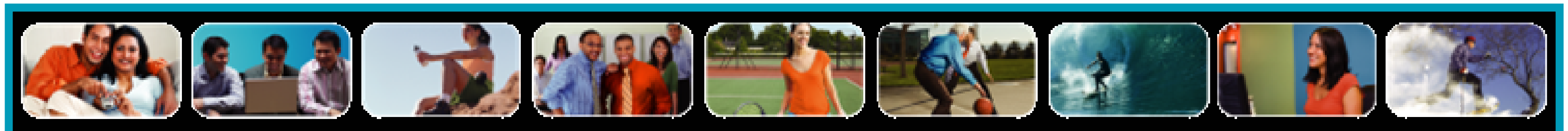
```
// @generated
namespace Package1
{

    // @generated
    public class Class3
    {

        // @generated
        public class Class1
        {

        }
```

Agenda

- Why RME .NET?
- MDD theories of operations with RME .NET
- Migrate XDE code models to RME .NET

XDE Code Model Importer

- Migrate from XDE .NET to RME .NET
- First use XDE Model Importer to import XDE .NET code models
- Then use XDE Code Model Importer to import the associated code. This will:
  - Apply RME .NET's C# profile to your conceptual model
  - Add @generated tags to code that was generated by XDE; UML to C# transform can re-apply itself correctly with these tags present
  - "Replace" framework elements with vized elements
- Now continue with RME .NET and use its features for developing your imported artifacts



Questions

Thank You

Steve Arnold (steve.arnold@uk.ibm.com)