Towards an MDA-Oriented UML Profile for Distribution

Raul Silaghi
Frédéric Fondement
Alfred Strohmeier

• SEL – EPFL

Contents

• Motivation
• MDA-Oriented UML-D Profiles
• Enterprise Fondue Refinement Process
• Example and MTL Model Transformations
• Parallax Tool Support
• Conclusions and Future Work
Motivation (1)

- Middleware Plethora
  - COM/DCOM/COM+
  - RMI
  - CORBA/CCM
  - EJB/J2EE
  - .NET
  - Jini
  - Web Services
  - MOM: MQSeries, JMS, MSMQ

- Model-Driven Architecture (MDA)
  - PIMs?  PSMs?  Code

MDA Context

- MDA Platform Relativism
  - Middleware = MDA Platform

- MDA Modeling Language
  - UML = de facto industry standard
Motivation (2)

- MDA?
- Distributed Enterprise Systems?
- Middleware-Mediated Distributed Systems?
- Middleware Code Generation?

- Support for *understanding*, *describing*, and *implementing* middleware-specific concerns:
  - distribution, concurrency, transactions, security, etc.

UML Extension Mechanisms

- Stereotypes
- Constraints
- Tag Definitions
- Tag Values

- **UML Profiles** = coherent set of extensions, defined for specific purposes
(Simplified) Bank Case Study

System

+createAccount(name : String, bal : Double) : AccountI
+getAccount(name : String) : AccountI
+transfer(ac1 : String, ac2 : String, amount : Double)

-Parameter- 

+getBalance() : Double
+withdraw(amount : Double)
+deposit(amount : Double)

Account

Bank

ConcernProfile (PIM)

AbstractConcernRealizationProfile

TechnologyConcernRealizationProfile (PSM)

Platform Specific Code

Abstraction Levels
Distribution Profile

- UML 2.0 profile
- UML 1.5 compliance:
  - rename `InstanceSpecification` to `Instance`

- Remote interfaces are «distributed»
- A «servant» is a root object to access functionalities of the system

Abstract Distribution Realization Profile

- Servants are published in specialized publisher systems
  - Publisher are servant objects
  - Publisher inside publishers

- A servant is published with expositions that embed information such as
  - Names
  - Behaviors (e.g., give me a print operation)
**CORBA Distribution Realization Profile**

- **«CORBA»**
  - NameExposition
  - NamingService
    - host
    - port

Code for CORBA Distribution

CORBA-XML-Config-File (Deployment Configuration)

**Enterprise Fondue Refinement Process**

- 5 layers
  - components, concerns, technologies, platforms, languages

- Refinements along concern-dimensions:
  - Middleware-specific concern-dimensions
  - Technology-dimension
  - Platform- and Language-dimension
Enterprise Fondue Refinement Process (1)

UML Metamodel

UML Design Model

Configuration

MTL1

MTL2

Technology Profile

UML Profile for Concern C

UML Model with elements for C

MTL1 - MTL Model Transformation

C - a middleware-specific concern, e.g., distribution
T - a middleware technology, e.g., CORBA
P - a middleware platform, e.g., OpenORB
L - a programming language, e.g., Java
MTL - MTL Model Transformation

Enterprise Fondue Refinement Process (2)

UML Model with elements for C

MTL2

MTL21

MTL22

UML Profile for Technology T

UML Profile for Concern C

UML Profile for C, on T

Configuration

Refinement along technology-dimension

Merge

L2

L1

C - a middleware-specific concern, e.g., distribution
T - a middleware technology, e.g., CORBA
P - a middleware platform, e.g., OpenORB
L - a programming language, e.g., Java
MTL - MTL Model Transformation
Enterprise Fondue Refinement Process (3)

C - a middleware-specific concern, e.g., distribution
T - a middleware technology, e.g., CORBA
P - a middleware platform, e.g., OpenORB
L - a programming language, e.g., Java
MTL - MTL Model Transformation

Refinement along platform-dimension & language-dimension

(Simplified) Bank Case Study

System

«Interface»
Bank
+createAccount(name : String, bal : Double) : AccountI
+getAccount(name : String) : AccountI
+transfer(ac1 : String, ac2 : String, amount : Double)

«Interface»
Account
+getBalance() : Double
+withdraw(amount : Double)
+deposit(amount : Double)

Bank
+createAccount(name : String, bal : Double) : AccountI
+getAccount(name : String) : AccountI
+transfer(ac1 : String, ac2 : String, amount : Double)
+getAccountList() : Account[*]

Account
+name : String
+balance : Double
+getBalance() : Double
+withdraw(amount : Double)
+deposit(amount : Double)
+setBalance(amount : Double)

bank 1
accounts
(Simplified) Bank Case Study

System Δ

```
+createAccount(name : String, bal : Double) : AccountI
+getAccount(name : String) : AccountI
+transfer(ac1 : String, ac2 : String, amount : Double)

«Interface»
AccountI
+getBalance() : Double
+withdraw(amount : Double)
+deposit(amount : Double)

Account
-name : String
-balance : Double
+getBalance() : Double
+withdraw(amount : Double)
+deposit(amount : Double)
+setBalance(amount : Double)
```

PIM-Level Distributed System

```
+metamodel (UML, UML::Interface, UML)
+profile DistributionProfile
+apply MTL1-D

Distributed System
```

Sep 24, 2004 EDOC'04, © Raul Silaghi
Why MTL?

- Transforms XMI-serialized UML models
- Supports the UML profiling mechanism
- Independent of CASE tools and model repositories
- Imperative language
- Easily available [INRIA, http://modelware.inria.fr/]
- Active community, maintained compiler

MTL vs. ATL vs. MTL-Transf [Jim Steel, Sep 17, 2004]

Distribution – MTL Snippets (1)

```java
//Within the MTL class Distributor
//-- toDistribute are the interfaces to be distributed
run() {

    if toDistribute.oclIsKindOf(m::Core::Interface!) {
        if (toDistribute.oclAsType(m::Core::Classifier!).stereotype.includes(distributedProfile.distributed).not()) {

            distributedProfile.applyStereotype(
                toDistribute,
                distributedProfile.distributed);
            //looks for classes and interfaces used in parameters and
            //return types of included operations
            treatOperationDependencies();
            //if called externally creates the corresponding
            //servant object
            treatServant();
        }
    }
}
```
Distribution – MTL Snippets (2)

```java
// Within the MTL class Distributor
// toDistribute are the interfaces to be distributed

treatOperationDependencies() {

    foreach (op : m::Core::Feature)
        in (toDistribute.feature)
        where (op.oclIsKindOf(!m::Core::Operation!)) {

        foreach (pa : m::Core::Parameter)
            in (op.parameter) {

            new Distributor().init(self, pa.type).run();
        }
    }
}
```

PIM-Level Distributed System

```
mtl-Distribute (UML::Interface, UML::Interface)
```

- `System` to `Configuration`
- `profile` `DistributedProfile`
- `conform` `DistributionProfile`
- `apply` `Distributed System`
- `Interface` `Distributed-Bank`
- `Distributed-servants = Set(b)`
- `Distributed-servants = Set(Bank)`
- `Interface` `Distributed-Account`
- `Distributed-servants = Set(l)`
- `Class Diagram` `Object Diagram`
XMI Snippet [Stereotype Definition]

```xml
<uml:stereotype xmi:id = 'a23' name = 'Distributed'
    isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
    isAbstract = 'false'>
    <uml:stereotype.baseClass>
        Interface
    </uml:stereotype.baseClass>
    <uml:stereotype.definedTag>
        <uml:tagDefinition xmi:id = 'a25' name = 'servants'
            isSpecification = 'false' tagType = 'Serveant'>
            <uml:Multiplicity xmi:id = 'a97'>
                <uml:Multiplicity.range>
                    <uml:MultiplicityRange xmi:id = 'a98'
                        lower = '0' upper = '-' />
                </uml:Multiplicity.range>
            </uml:Multiplicity>
        </uml:TagDefinition>
    </uml:stereotype.definedTag>
</uml:stereotype>
```

XMI Snippet [Stereotype Application]

```xml
<uml:interface xmi:id = 'a22' name = 'BankI'
    visibility = 'public'
    isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
    isAbstract = 'false'>
    <uml:modelElement.stereotype>
        <uml:stereotype xmi:idref = 'a23'/>
    </uml:modelElement.stereotype>
    <uml:modelElement.taggedValue>
        <uml:taggedValue xmi:id = 'a24' name = 'servants'
            isSpecification = 'false'>
            <uml:tagDefinition xmi:idref = 'a25'/>
            <uml:taggedValue.type>
                <uml:object xmi:idref = 'a26'/>
            </uml:taggedValue.type>
            <uml:taggedValue.referenceValue>
                <uml:object xmi:idref = 'a26'/>
            </uml:taggedValue.referenceValue>
        </uml:TagDefinition>
    </uml:modelElement.taggedValue>
</uml:Interface>
```
PSM-Level CORBA Distributed System

Abstract Distribution Realization Profile

- **Servants** are published in specialized **publisher** systems
  - Publisher are servant objects
  - Publisher inside publishers
- A **servant** is published with **expositions** that embed information such as
  - Names
  - Behaviors (e.g., give me a print operation)
Parallax Tool Support

- Eclipse plug-in
- http://parallax-lgl.epfl.ch/

- Modularization
  - Framework of plug-ins, extension points
- Separation of Concerns
  - Aspect-Oriented Support (AspectJ)

Framework of Parallax Plug-ins

- PrlxPlugin
- PrlxInputAdaptorPlugin
- PrlxCodeGeneratorPlugin
- PrlxConcernAspectPlugin
- PrlxConcernPlatformAspectPlugin
- PrlxConcernTechnologyAspectPlugin

«require»
«weave in»
«promote»
«require»
### Dependency Dimensions of PrlxPlugins

1. **PrlxInputAdaptorPlugins (2-DD)**
   - UML-Specification
   - UML-Case-Tool (UML-Case-Tool-Exporter)
   - XMI-Specification (???)

2. **PrlxCodeGeneratorPlugins (1-DD)**
   - Programming-Language

3. **PrlxConcernPlatformAspectPlugins (4-DD)**
   - Middleware-Concern
   - Technology
   - Platform
   - Programming-Language

### Distribution – XMI & Parallax Support

1. Stereotypes/TagValues → XMI
2. Deployment Configuration → CORBA-XML-Config-File
3. «Distributed»\textsuperscript{xmi} → CORBA IDLs
4. «Servant»\textsuperscript{xmi} → ORB + Naming Service
Parallax Output

```java
org.omg.CORBA.ORB orb = null;
org.omg.PortableServer.POA poa = null;
org.omg.CosNaming.NamingContextExt nc = null;
org.omg.CORBA.Object so = null;

orb = org.omg.CORBA.ORB.init("–ORBProfile=default", null);
poa = org.omg.PortableServer.POA.Helper.narrow(
    orb.resolve_initial_references("RootPOA"));
poa.the_POAManager().activate();
Bank b = new Bank();
so = poa.servant_to_reference(b);

    orb.resolve_initial_references("NameService"));
nc.rebind(nc.to_name("Bank"), so);

orb.run();
```

Conclusions

- Modeling support for middleware-specific concerns
- Enterprise Fondue refinement process
- MDA-Oriented UML-D Profiles
- MTL Model Transformations
- Parallax support for Code Generation
- Separation of Concerns, Several Abstraction Levels
Future Work

• MDA-oriented profiling for other middleware-specific concerns
  – UML-C, UML-T, UML-S, etc. => **UML-MS**

• Online Auction System
• Component and deployment diagrams
• Other middleware infrastructures
  (technologies and platforms)

Thank You!