Towards an MDA-Oriented UML Profile for Distribution

Raul Silaghi
Frédéric Fondement
Alfred Strohmeier

- SEL – EPFL

Contents

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?

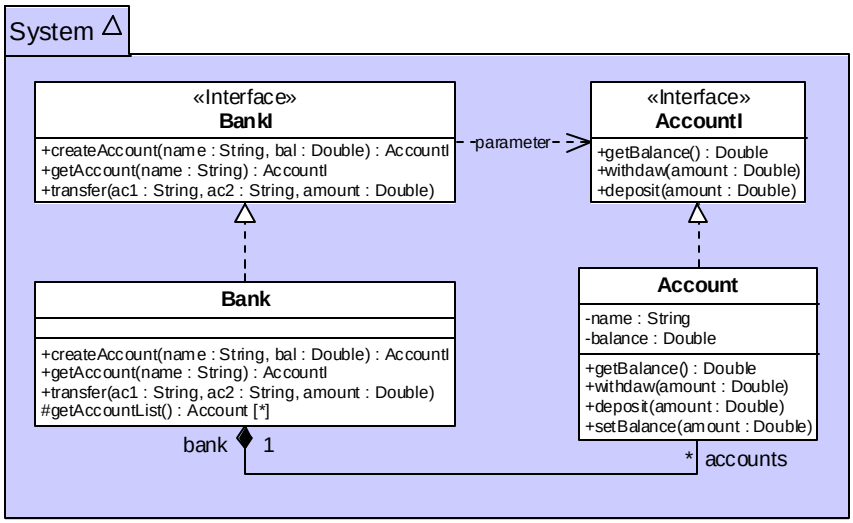
- Support for *understanding*, *describing*, and *implementing* middleware-specific concerns:
  - distribution, concurrency, transactions, security, etc.

UML Extension Mechanisms

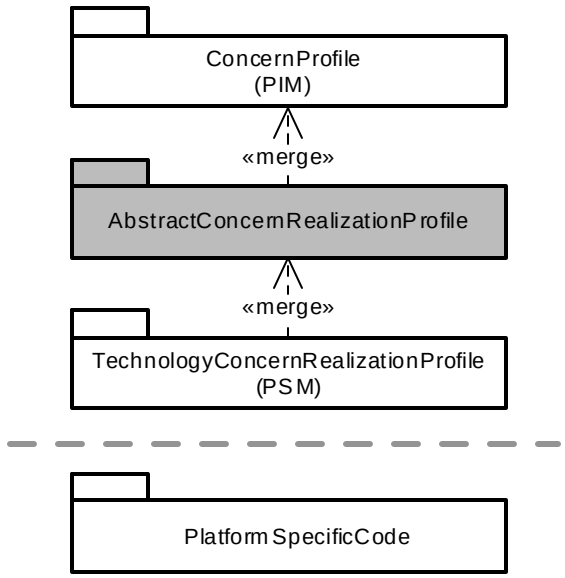
- Stereotypes
- Constraints
- Tag Definitions
- Tag Values

- UML **Profiles** = coherent set of extensions, defined for specific purposes

(Simplified) Bank Case Study

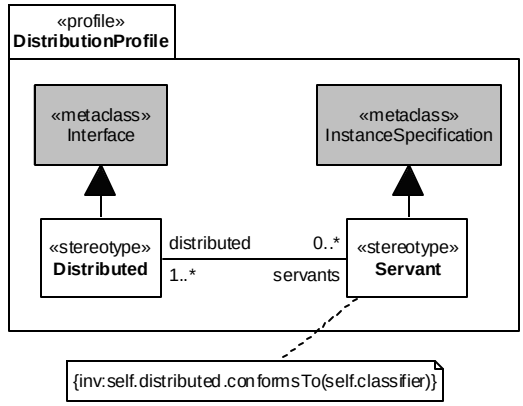


Abstraction Levels

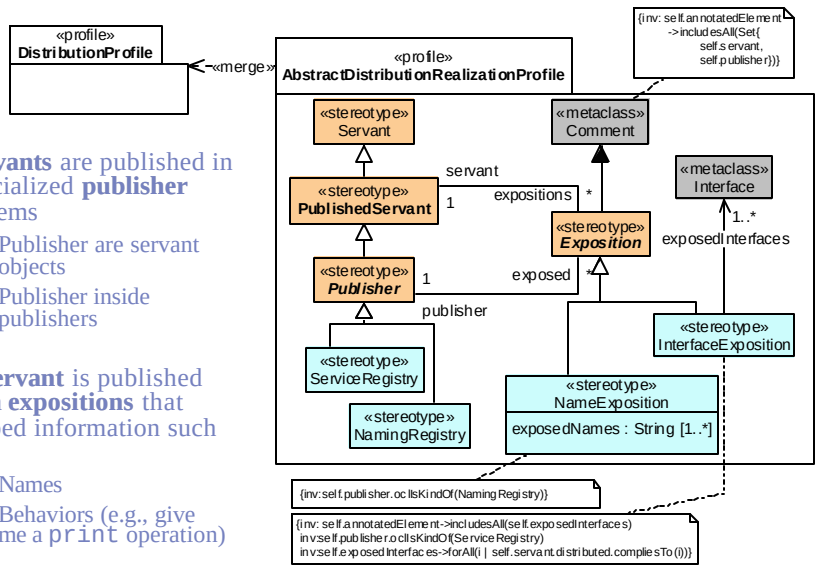


Distribution Profile

- 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

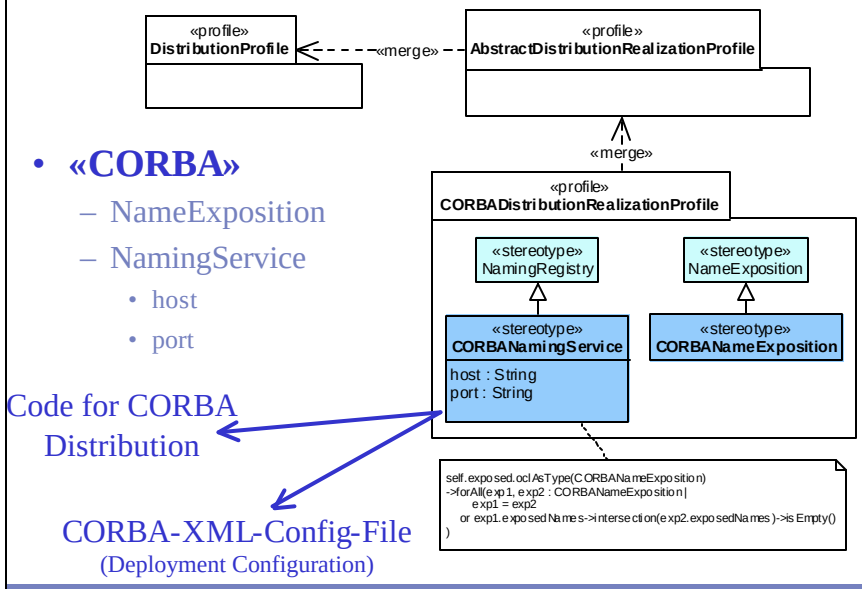


Abstract Distribution Realization Profile



- **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

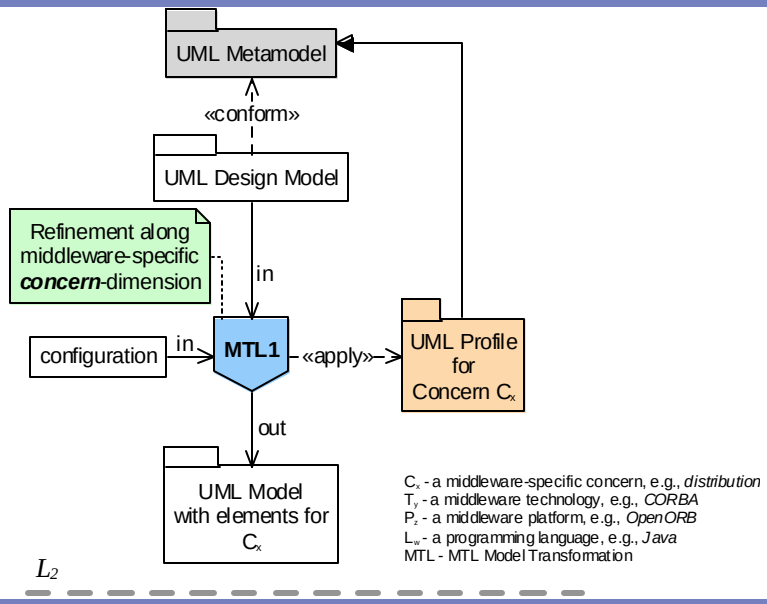
Code for CORBA Distribution

CORBA-XML-Config-File
(Deployment Configuration)

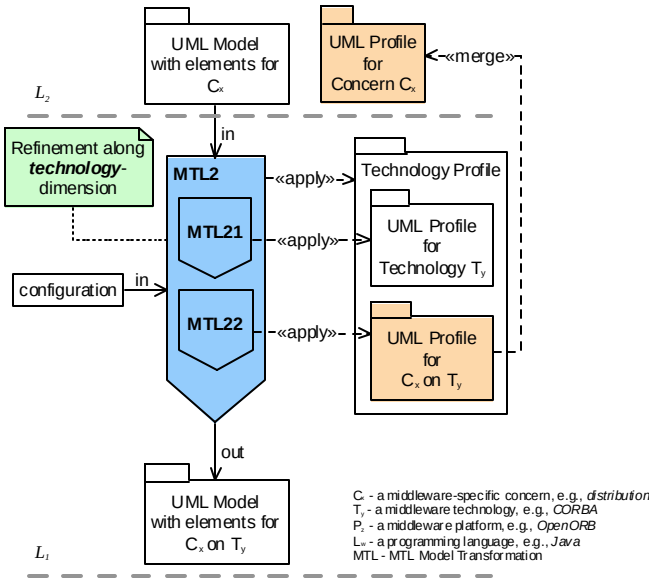
Enterprise Fondue Refinement Process

- 5 layers
  - components, concerns, technologies, platforms, languages
- Refinements along concern-dimensions:
  - Middleware-specific concern-dimensions
  - Technology-dimension
  - Platform- and Language-dimension

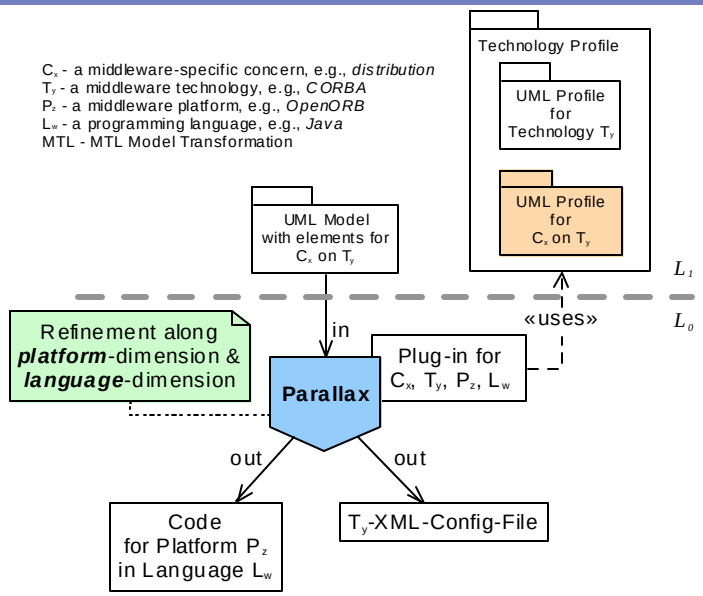
Enterprise Fondue Refinement Process (1)



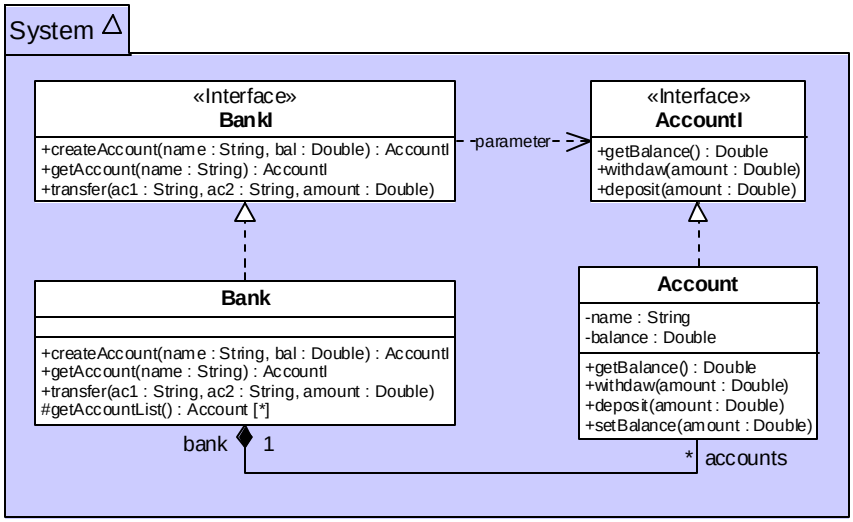
Enterprise Fondue Refinement Process (2)



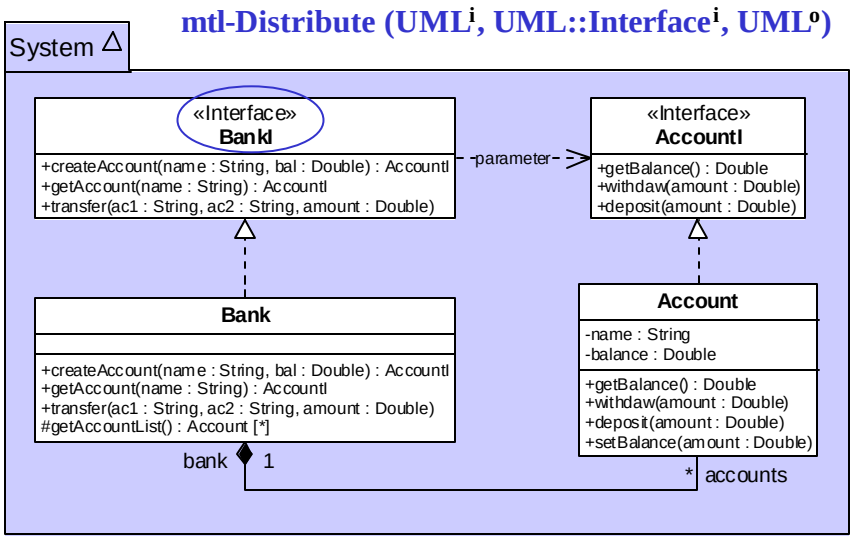
Enterprise Fondue Refinement Process (3)



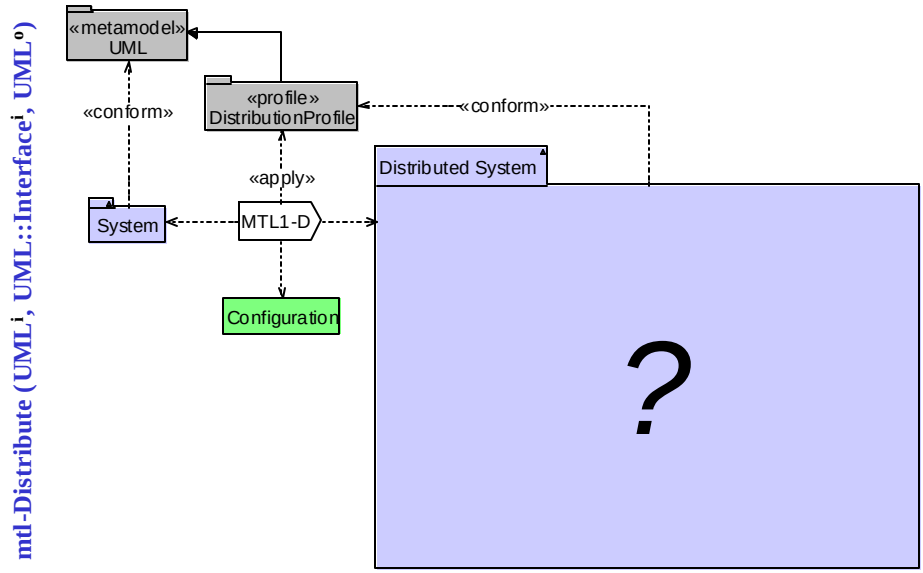
(Simplified) Bank Case Study



(Simplified) Bank Case Study



PIM-Level Distributed System



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)

```
//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)

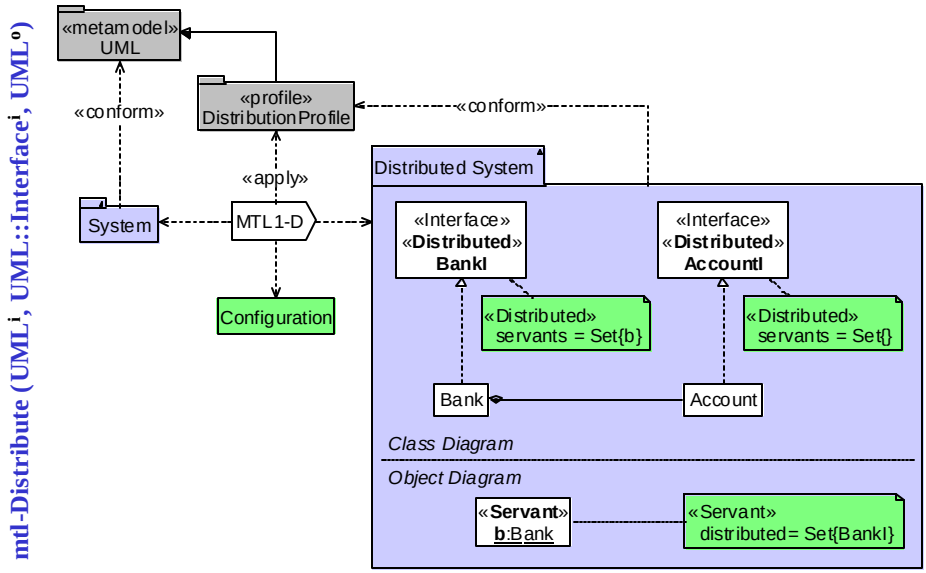
```
//Within the MTL class Distributor
//- toDistribute are the interfaces to be distributed
treatOperationDependencies() {

  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



XMI Snippet [Stereotype Definition]

```
<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 = 'Servant'>
      <UML:TagDefinition.multiplicity>
        <UML:Multiplicity xmi.id = 'a97'>
          <UML:Multiplicity.range>
            <UML:MultiplicityRange xmi.id = 'a98'
              lower = '0' upper = '-1'/>
          </UML:Multiplicity.range>
        </UML:Multiplicity>
      </UML:TagDefinition.multiplicity>
    </UML:TagDefinition>
  </UML:Stereotype.definedTag>
</UML:Stereotype>
```

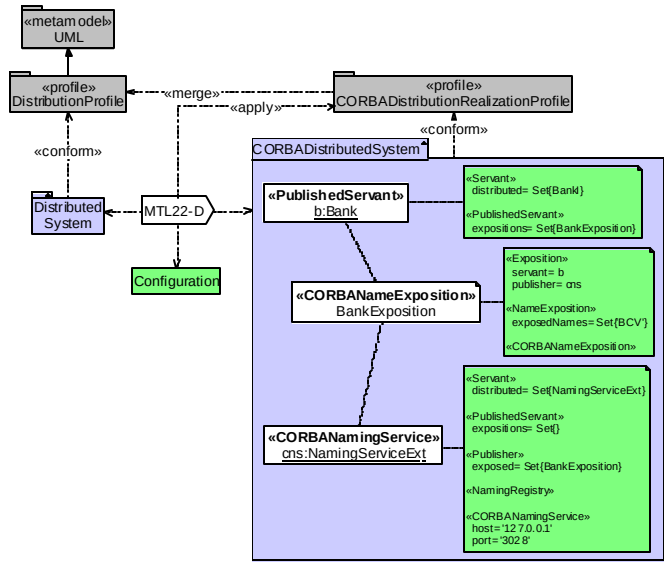

XMI Snippet [Stereotype Application]

```
<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:TaggedValue.type>
        <UML:TagDefinition xmi.idref = 'a25'/>
      </UML:TaggedValue.type>
      <UML:TaggedValue.referenceValue>
        <UML:Object xmi.idref = 'a26'/>
      </UML:TaggedValue.referenceValue>
    </UML:TaggedValue>
  </UML:ModelElement.taggedValue>
  <UML:Classifier.feature>
    ...
  </UML:Classifier.feature>
</UML:Interface>
```

```
<UML:Object
  xmi.id = 'a26'
  name = 'b'
  isSpecification = 'false'>
```

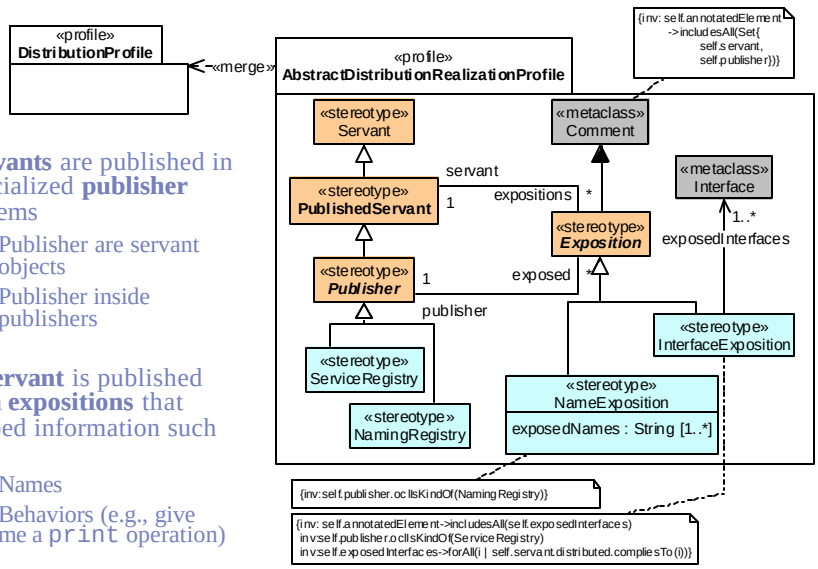
PSM-Level CORBA Distributed System

mtl-CORBA-Distribute (UML^i, UML^o)



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)

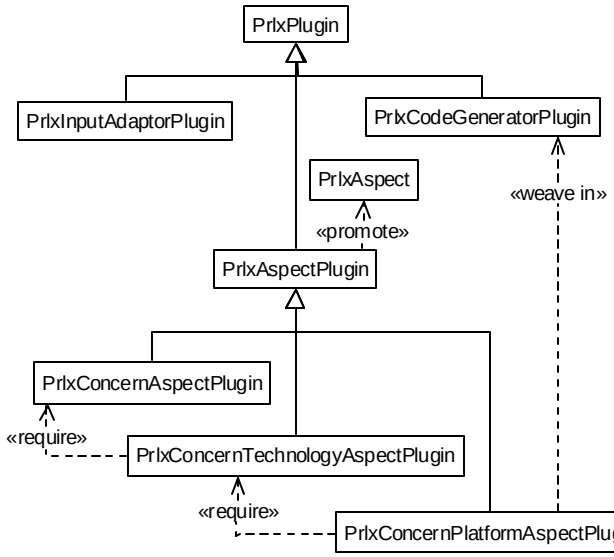


Parallax Tool Support

- 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



Dependency Dimensions of PrlxPlugins

- PrlxInputAdaptorPlugins (**2-DD**)
  - *UML-Specification*
  - *UML-Case-Tool (UML-Case-Tool-Exporter)*
  - *XMI-Specification* (???)
- PrlxCodeGeneratorPlugins (**1-DD**)
  - *Programming-Language*
- PrlxConcernPlatformAspectPlugins (**4-DD**)
  - *Middleware-Concern*
  - *Technology*
  - *Platform*
  - *Programming-Language*

Distribution – XMI & Parallax Support

- Stereotypes/TagValues → XMI
- Deployment Configuration → CORBA-XML-Config-File

- «Distributed»^{xmi} → CORBA IDLs
- «Servant»^{xmi} → ORB + Naming Service

Parallax Output

```
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);

nc = org.omg.CosNaming.NamingContextExt Helper.narrow(
          orb. resolve_initial_references("NameService"));
nc.rebind(nc.to_name("Bank"), so);

orb.run();
```

CORBA-XML-Config-File !

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 ! ?