Creating Applications Using Parameterized Frameworks

Quickly developed and highly customized

OMG’s Second Workshop on UML for Enterprise Applications:
Model Driven Solutions for the Enterprise
December 6, 2001
Burlingame, CA USA

Tomoo Yoda
Synergy Research Corporation
yoda@synergy-res.co.jp
Who am I?

- The president of Synergy Research Corporation—Tokyo, Japan
- Providing OO analysis/design and systems integration services to Japanese clients
- Focused on parameterized frameworks and their application for software productivity
- Recently developed a tool and a process to support MDA in a parameterized way
- Joined the initial submission to OMG EDOC RFP as a former member of CBOP
What I am presenting

- Our trial MDA process with parameterized frameworks
  - Framework
  - Process
  - Composition and Transformation
  - Tools
- Findings
The motivation

In many years of OO analysis/design and systems integration, I have repeatedly seen things from past experience reappear in different forms in new projects.

But the knowledge of these repeatedly reappearing things reside only in our heads and is hard to reuse effectively.

(Binary) Components have always been limited solutions, because they are a mixture of many, many specific things, e.g.:

- Client industry, Client business rule, Client’s customer’s preferences
- Operating System, Computer language, Distribute platform (EJB, proprietary O-R mapping, Client/Server, ...)

What we need to represent is this knowledge of proven best practice, not the things in specific forms!

To reuse this knowledge effectively, it must be represented in an organized, visible and easily manipulated way.
Knowledge: Right Structure

- Right partitioning
  - Business knowledge
    - Business concept
    - Relationship between business concepts
    - Best practice domain model
    - Standard domain model
  - Technology knowledge
    - Platform
    - Optimization
    - Deployment

- Right granularity
But How?

- Framework
- Behavior of framework
- Framework import mechanism
- Model transformation
Frameworks can be Confusing

- Traditionally
  - A collection of collaborating abstract classes
- In Catalysis context
  - A package designed to be imported with substitutions. It “unfolds” to provide a version of its contents that is specialized based on the substitutions made.
  
  (The Catalysis Approach, by Desmond F. D’Souza and Alan Cameron Wills)
- OMG EDOC Chapter 4 ‘The Patterns Profile’
- Simply called ‘Framework’
Substitution

[PartyRole | ShipTo]

```
< Party >

<PartyRole>Name:String
<PartyRole>Address:String
<PartyRole>ZipCode:String

get<PartyRole>Name():String
set<PartyRole>Name(name :String):void
get<PartyRole>Address():String
set<PartyRole>Address(address:String):void
get<PartyRole>ZipCode():String
set<PartyRole>ZipCode(zipcode:String):void

<Party>

shipToName:String
shipToAddress:String
shipToZipCode:String

getShipToName():String
setShipToName(name :String):void
getShipToAddress():String
setShipToAddress(address:String):void
```
Framework Import Mechanism

- Framework inheritance
  - Single
  - Multiple

- Inheritance can be controlled by substitutions on import
If the MDA process was to start with a large PIM, its power would be limited.
Framework Import Relationships: An example

PSMBookStoreOrder

PFWEJ BSession
PFWEJBEntity
PFWEJB

Platform Area

PIMBookStoreOrder

PI MBookStoreOrder

PIMCustomerOrder

PIMReferenceList

PIMList

PIMOnLineCustomer

PIMParty

PIMProductBook

PIM Area

<<import>>
<<import>>
<<import>>
<<import>>
<<import>>
<<import>>
<<import>>
<<import>>
<<import>>
<<import>>
<<import>>
<<import>>

is equivalent to
Synergy Exploder

Off-the-shelf CASE Tool such as Rational Rose

XML Loader/ Unloader

Automatic Namespace Maintenance Mechanism

UML Type System

UML Manipulation Interface

Exploder

XMI

XSLT

XML

Proprietary XML Formalism for Structural Elements in UML Foundation.Core Package
Working with a CASE Tool
Attribute-centric Framework

Parameter

<<concept>>

<<instanciate>>

create (partyRole)FirstName : String, (partyRole)LastName : String, (partyRole)Address : String, (partyRole)City : String, (partyRole)State : String, (partyRole)ZipCode : String)

<<concept>>

<<PartyComponent>>

(getPartyRole)FirstName() : String
(getPartyRole)LastName() : String
(getPartyRole)Address() : String
(getPartyRole)City() : String
(getPartyRole)State() : String
(getPartyRole)ZipCode() : String

(setPartyRole)FirstName(firstName : String) : Void
(setPartyRole)LastName(lastName : String) : Void
(setPartyRole)Address(address : String) : Void
(setPartyRole)City(city : String) : Void
(setPartyRole)State(state : String) : Void
(setPartyRole)ZipCode(zipCode : String) : Void
Operation-centric Framework

```
<ItemComponent>

<ListComponent>

<<concept>>
  <List>
  (<ListComponent> から)

add<Item>(<item> : <Item>) : Void
remove<Item>(<item> : <Item>) : Void
getNumOf<Item>s() : Integer
getNth<Item>(index : Integer) : <Item>
get<Item>() : Collection
```

Parameter
```
<List>
<ListComponent>
<Item>
<ItemComponent>
<List-has-Items>

{ordered}
```

JSR (Java Specification Request) #26 gives the UML profile to be used here.
Resolving Name Conflicts

Loser/Winner Report
Unfolding Yields PSM