Modeling the CoCoME with KobrA

Monika Juhasz
August 1st, 2007

Chair of Software Engineering
University of Mannheim
Chair of Software Engineering

- Leader: Prof. Dr. Colin Atkinson

- Research focus
  - Component Development
  - Multi-Level Modeling
  - Component Discovery
  - Mobile Business

- Current projects
  - Aristaflow
    - component-oriented development of adaptive process-oriented enterprise software
  - Mobile Business
    - development of a generic software platform for mobile devices
  - ECOMODIS
    - efficient component-based development of dependable systems
Contents

- KobrA – an overview
  - Reuse technologies
  - Core concepts

- Modeling the CoCoME
  - Computation Independent Model
  - Platform Independent Model
  - Platform Specific Model

- Conclusion
  - Summary
  - Pros and Cons
  - Tool support
KobrA

- UML-based modeling method for component-based systems
- developed at Fraunhofer Institute for Experimental Software Engineering
- KobrA = „KoMponenten basierte Anwendungsentwicklung“ (German for component based application development)
- here: updated version of KobrA
Reuse technologies

Component-based Development (CBD)

Vision
- Assemble applications from prefabricated parts
- COTS component market
- Web Services

Development activities oriented around product families
- Manage commonalities and variabilities

Product-Line Engineering (PLE)

Model-Driven Architecture (MDA)

Vision
- Capture core software assets as platform-independent models (PIMs)
- Automatically map PIMs to platform-specific models (PSMs)
Contents

- KobrA – an overview
  - Reuse technologies
- Core concepts
  - Separation of Projections
  - Separation of Specification and Realization
  - Separation of Process and Products
  - Separation of Development Dimensions
  - Modeling Principles
- Modeling the CoCoME
- Conclusion
Separation of Projections

Functional projection

Behavioral projection

Structural projection

Software object
Separation of Specification and Realization

Functional projection

Structural projection

Behavioral projection

R

S

software object
Separation of Product and Process
Separation of Development Dimensions

Abstraction (MDA)

Composition (CBD)

Genericity (PLE)

Application

Framework

instantiation
decomposition
refinement
Modeling Principles

- Uniformity
  - all behavior rich elements should be viewed as components, including (sub)systems

- Locality
  - all models are views
  - all views should be local to a component

- Parsimony
  - minimal set of concepts (no redundancy)
Contents

- KobrA – an overview

- Modeling the CoCoME
  - Computation Independent Model
  - Platform Independent Model
  - Platform Specific Model

- Conclusion
Modeling the CoCoME
Computation Independent Model (CIM)

- Enterprise Concept Diagram (Structural view)
Computation Independent Model (CIM)

- Enterprise Process Diagram (Functional view)
Computation Independent Model (CIM)

- Enterprise Workflow Diagram (Behavioral view)
Platform Independent Model (PIM)

- Context Realization - Structural View

```
<<Component>>
Bank
+validateCard()
+debitCard()

<<Component>>
TradingSystem
+cardPayment()
+cashPayment()
+changePrice()
+enterAmount()
+finishSale()
+identifyItems()
+listProducts()
+listProductsRunningOutOfStock()
+orderProducts()
+rollInDelivery()
+showDeliveryReport()
+showStockReport()
+startNewSale()
```

Manager

Receipt *

StockManager

Cashier

Supplier 1..*

Order 1..*

Product 1..*

Customer 1..*
Platform Independent Model (PIM)

- Context Realization - Functional View
  - OrderProducts task

![Diagram showing interactions between Manager, TradingSystem, and Supplier, with sequence numbers 1 to 8 for interactions.]

1: list products
2: list of Products
3: list products running out of stock
4: list of Products
5: orderProducts(ProductList)
6: orderProducts(ProductList)
7: order ID
8: shows list of order IDs
Platform Independent Model (PIM)

- Context Realization - Behavioral View
  - OrderProducts task
Platform Independent Model (PIM)

- TradingSystem Specification
  - Structural View

```
<<subject>>
<<virtual>>
TradingSystem

+cardPayment
+cashPayment
+changePrice
+enterAmount
+finishSale
+identifyItems
+listProducts
+listProductsRunningOutOfStock
+orderProducts
+rollInDelivery
+showDeliveryReport
+showStockReport
+startNewSale

<<acquires>>

<<Component>>
Bank
+validateCard
+debitCard

<<role>>
Supplier

Order
-id

Product
-EAN
-id
-name
-purchasePrice
-salesPrice
```
Platform Independent Model (PIM)

- TradingSystem Specification - Functional View
  - changePrice responsibility

<table>
<thead>
<tr>
<th>Name</th>
<th>changePrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This method changes the salesPrice of a product.</td>
</tr>
<tr>
<td>Receives</td>
<td>productID : String – the identifier of the product</td>
</tr>
<tr>
<td></td>
<td>price : Real – the new salesPrice of the product</td>
</tr>
<tr>
<td>Returns</td>
<td>Success : Boolean</td>
</tr>
<tr>
<td></td>
<td>– true, if the salesPrice is changed</td>
</tr>
<tr>
<td></td>
<td>– false, otherwise</td>
</tr>
<tr>
<td>Changes</td>
<td>The salesPrice of the product.</td>
</tr>
<tr>
<td>Assumes</td>
<td>The productID must be valid and the price parameter has to be greater than zero.</td>
</tr>
<tr>
<td>Result</td>
<td>The salesPrice of the product with the productID is set to price.</td>
</tr>
</tbody>
</table>
Platform Independent Model (PIM)

- TradingSystem Specification - Behavioral View

TradingSystem has no externally visible states, as it serves multiple users simultaneously.
Platform Independent Model (PIM)

- TradingSystem Realization - Structural View
Platform Independent Model (PIM)

- TradingSystem Realization - Functional View
  - rollInDelivery responsibility
Platform Independent Model (PIM)

- TradingSystem Realization - Behavioral View
  - rollInDelivery responsibility
Platform Independent Model (PIM)

- CashDesk Specification - Structural View
### Platform Independent Model (PIM)

- **CashDesk Specification - Functional View**
  - **CardPayment responsibility**

<table>
<thead>
<tr>
<th>Name</th>
<th>cardPayment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This method realizes the payment via card.</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>For card payment the connection to the bank has to be established.</td>
</tr>
<tr>
<td><strong>Receives</strong></td>
<td>sum: Real – the sum to pay</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td>success: Boolean – true, if payment successful; false, otherwise</td>
</tr>
<tr>
<td><strong>Sends</strong></td>
<td>Bank::validateCard(cardInformation, pin)</td>
</tr>
<tr>
<td></td>
<td>Bank::debitCard(transactionID)</td>
</tr>
<tr>
<td><strong>Changes</strong></td>
<td>The card of the customer is debited.</td>
</tr>
<tr>
<td><strong>Rules</strong></td>
<td>Bank::debitCard(transactionID) can only be called, if the card is valid. To validate the card the customer has to enter his PIN. The cardInformation is read from the credit card of the customer.</td>
</tr>
<tr>
<td><strong>Assumes</strong></td>
<td>The sum of the prices of the goods must be greater than zero.</td>
</tr>
<tr>
<td><strong>Result</strong></td>
<td>The card of the customer is debited with the sum.</td>
</tr>
</tbody>
</table>
Platform Independent Model (PIM)

- CashDesk Specification - Behavioral View
Platform Independent Model (PIM)

- CashDesk Realization - Structural View
Platform Independent Model (PIM)

- CashDesk Realization - Behavioral View
  - IdentifyItems "operation"
Platform Independent Model (PIM)

- CashDesk Realization - Functional View
  - IdentifyItems “operation”
Platform Specific Model (PSM)

- CashDesk
Platform Specific Model (PSM)

- Deployment Diagram
Contents

- KobrA – an overview
- Modeling the CoCoME
- Conclusion
  - Summary
  - Pros and Cons
  - Tool support
Summary

- Uniform representation of components and composite systems in a hierarchical form

- Separation of concerns
  - Orthogonal Development Dimensions
  - Orthogonal Projections/Views
  - Separation of Specification and Realization
  - Separation of Product and Process
Pros and Cons of KobrA

- Simple and Systematic
  - strict separation of concerns

- Incremental introduction of components and product lines

- Uniform treatment of systems and components
  - component assembly = component creation
  - fractal-like product, recursive process

BUT

- Fairly complex and difficult to apply

Tool support needed
Tool Support

- Main objectives:
  - User-friendly framework for defining and navigating around the different views
  - Infrastructure for consistency checking and view-generating tools
  - Unifying meta-model allowing all views to be generated automatically from a single underlying representation of a component
Tool Support
Thank you.