Modeling the CoCoME with KobrA

Monika Juhasz August 1st, 2007

> Chair of Software Engineering University of Mannheim

> > 1



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

Component-based Product Line Engineering with UML

Colin Atkinson, Joachim Bayer, Christian Bunes, Erik Kamstles, Olivar Laitenberger, Roland Laque, Dirk Muthig, Barbara Paech, Jürgen Wust, Jörg Zettel



2002

Reuse technologies

Component-based Development (CBD)

Vision

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

Vision

- Development activities oriented around product families
- Manage commonalities
 and variabilities

Product-Line Engineering (PLE)



- Capture core software assets as platform-independent models (PIMs)
- Automatically map PIMS to platformspecific 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 Product and Process



Separation of Development Dimensions



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 Process Diagram (Functional view)



Computation Independent Model (CIM)

Enterprise Workflow Diagram (Behavioral view)



Context Realization - Structural View





Context Realization - Functional View

> 1: list products 2: list of Products

> 4: list of Products

5: orderProducts(ProductList)

8: shows list of order IDs

OrderProducts task

: Manager



- Context Realization Behavioral View
 - OrderProducts task







- TradingSystem Specification Functional View_
 - changePrice responsibility



Name	changePrice			
Description	This method changes the salesPrice of a product.			
Receives	<i>productID</i> : String – the identifier of the product <i>price</i> : Real – the new salesPrice of the product			
Returns	Success : Boolean – true, if the salesPrice is changed – false, otherwise			
Changes	The salesPrice of the product.			
Assumes	The <i>productID</i> must be valid and the <i>price</i> parameter has to be greater than zero.			
Result	The salesPrice of the product with the <i>productID</i> is set to <i>price</i> .			

TradingSystem Specification - Behavioral View_



TradingSystem Realization - Structural View





- TradingSystem Realization Behavioral View
 - rollInDelivery responsibility







- CashDesk Specification Functional View
 - CardPayment responsibility



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





- CashDesk Realization Behavioral View
 - IdentifyItems "operation"







Platform Specific Model (PSM)

CashDesk



32

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

Component Development IDE - TradingSystem - Specification - Structural - Class Diagram - Eclipse SDK						
<u>F</u> ile <u>E</u> dit Diagrams <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject Component Actions Layout Octopus <u>R</u> un MagicDraw <u>W</u> indow <u>H</u> elp						
🖸 • 🖫 👜 🖉 🖄 🗍 😢	🙆 🔟 🚈 💁 -	A Car + 2 + 7 → + → + E Component C	5 »			
Dimension Explorer	Admin-Tool 🛛 🔝 1	FradingSystem - Specification - Structural - Class Diagram 🗙	- 0			
[<mark></mark> * ▼	8	Å · [≫] ∕ [≫] ≜ · ∠ · ▲ ·				
 Component 	🛅 Common	package Data(🖳 TradingSystem - Specification - Structural - Class Diagram)				
Bank	🗈 Notiz 🔹 🔻					
	sbo Textbox 🔹 🔹					
TradingSystem	🔎 Anker zu Notiz 🔹 🔻	·····				
	_ [⊕] Containment					
Abstraction Lough	🏸 Abhängigkeit					
	Trennzeichen 🔹					
	🔏 Class Diagram					
Implementation	📕 Klasse 🔹 🔻					
	-• Schnittstelle	< <komponent>> TradingSystem</komponent>				
Projection	🛅 Paket 🔹 🔻	Bank ProcessSale()				
····· 005	🏸 Generalisierung 🔹	+validateCard() +dehitCard() < <acquires>> +changePrice() -id</acquires>				
	🗸 Assoziation 🔹 🔹	+listProducts() +listProductsRunningOutOfStock()				
Structural	🖌 Aggregation 🔹	+orderProducts() -purchasePrice				
Functional	🖌 Komposition 🔹	+snowStockreport() +showDeliveryReport()	1			
benavioural	🖉 Schnittstellenre 🔻	+rollnDelivery()				
	🔁 Verwendung 🔹 🔻	Stockmanager				
	📌 Abstraktion 🔹 🔹					
	🚍 Instanz					
	/ _L Link					
	💫 Use Case Diagram					
Create Perspective 🛛 🗖 🗖	🔙 Implementation Di					
	Composite Struct					
UML class diagram	Information Flows					
ComponentDescription	R Profiling Mechanism	·····				
	•					
🗖 Available Perspectives 🛿 📃 🗖						
UML class diagram 1						
			_			

