Orthographic Software Modeling –
A Practical Approach to View-Oriented, Component-Based Development

Colin Atkinson

Charles University, Prague
9th February 2010
Outline

- Motivation
- Review of KobrA
- Orthographic Software Modeling
  - View-based software engineering
  - Dynamic view generation
  - Dimension Based Navigation
- Case study
- Testing
- Conclusion
Modern Software Engineering Paradigms

- Model-driven Software Development
  - Levels of abstraction (PIMs, PSMs etc.)

- Component / Service - based Development
  - Composition (black box views, white box views etc.)

- Product-line Engineering
  - product families (common features, variable features etc.)

- Aspect-Orientation
  - aspect-orientation (core functionality, aspects etc.)

- Other trends
  - Models at run-time (intelligent, adaptive software systems)
  - Enterprise architecture (non-functional concerns)
Observations

- New software engineering paradigms dramatically increase number of views
  - especially when used together

- “Enterprise Architecture” adds more information and more views

- Executability is only one property of a system. Others include –
  - Checkability
  - Predictablity
  - Testability

- To support these effectively we need …
  - Inherently View-oriented Method(s)
  - Dynamic View Generation
  - Dimension Based Navigation

Orthographic System Modeling
other engineering disciplines have a long and successful tradition of technical drawing

- orthographic projection

- so why don't we do this in software engineering?
Basic Philosophy of our Approach

- Support for multi-views
  - Multiple inter-related diagrams and renderings

- Views as Projections
  - should be as “orthogonal” as possible

- Multi-Dimensional Navigation Space
  - every view exists at a particular set of coordinates

- Flexible infrastructure that can be easily extended
  - existing new tools can be easily integrated

- Single underlying representation of the software system
  - “code” is just a view
  - there is no code
  - everything is code
KobrA Modeling Principles

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

Parsimony
- minimal set of concepts (no redundancy)

Locality
- all models should be local to a component

Encapsulation
- component specifications (what) must be separated from component realizations (how)
Component Views in KobrA

**Specification**

Operational View
(operation specifications)

Operational View
(UML communication diagrams)

Behavioral View
(UML statechart diagram)

**Realization**

Structural View
(UML class/object diagrams)

Structural View
(UML class/object diagrams)
Component Nesting in KobrA

UML-based modeling of nested components

- attempts to reconcile MDD, CBD and PLE

Key ideas

- Three fundamental development dimensions
- Three fundamental viewpoints
- Strict separation of black box and white box views

Key benefits

- Full power of UML available to model components
- OCL available for more formality and precision
Separation of Development Dimensions

Abstraction (MDA)

Composition (CBD)

Genericity (PLE)

Application

Framework

instantiation

decomposition

refinement
View-Based Modeling Frameworks – Current Status

- UML classes
- Behavior
- Java source
- CD
- OpSpec
- RegEx
- XMI
- test cases
- code
- software component
- Java source
- UML classes
- Behavior
Goal – On-Demand View Generation
Views as Projections

Functional projection

Behavioral projection

Structural projection

system object
Separation of Specification and Realizations

- Functional projection
- Structural projection
- Behavioral projection

System object
Component model derived from KobrA (KobrA 2.0)

fully integrated into UML

formalized definition through OCL constraints

views generated on demand via ATL transformations
Single Underlying Model – Core Metamodel
The Navigation Jungle
Dimension-Based Navigation

- Core development concerns mapped to dimensions
- In every dimension, exactly one dimension element can be selected
- Once an element is selected in every dimension, an appropriate view is generated
- A transformation is launched and an editor is started

![Diagram of a system with a cell highlighted]
Navigation

- Dimension configuration example
  - Component (Composition)
  - Abstraction Level
  - Information Hiding / Encapsulation
  - Projection
  - Variant
  - Language
  - Notation
Software Modeling Environment Roles

- Methodologist
  - creates rules for dimensions and its contents
    - predefined elements
    - source of dimension elements (e.g. dynamically generated from the single underlying model)
    - changeability rights by developer

- Developer
  - uses dimension based navigation
  - manipulates software system through “views”
Case Study: Mobile Tourist Guide

Component
- ContextManager
- ContextMatcher
- GuiManager
- MobileTouristGuide
- TouristGuideService

Abstraction Level
- PIM
  - PSM
  - Java

Information Hiding
- Public
- Private

Projection
- Structural
- Operational
- Behavioural
- Variational

Variant
- Generic
  - Variant A
  - Variant B

Language
- UML class diagram
- SDL class diagram
- Taxonomy

Notation
- Graphical
- Tabular
- Syntax Tree

Preferences
- indoor: Boolean
- landmarks: Boolean
- markets: Boolean
- museums: Boolean
- outdoor: Boolean
- recreation: Boolean

ContextSet
- contextItems: ContextItem[]

Video
- video: MPEG4Video

<<subject>>
MobileTouristGuide
+ searchTouristAttractions()
+ editProfile()
+ editPreferences()
+ getRouteMap()
<<variant>> + getVideo()

<<acquires>>

<<Komponent>>
TouristGuideService
+ getTouristAttraction( c: ContextSet, p: Preferences )
+ getRouteMap( c: ContextSet, touristAttractionID: long )
+ getVideo( c: ContextSet, touristAttractionID: long )
+ registerService( s: ServiceDescription, v: Video )

RouteMap
- map: JPEGImage

TouristAttractionList
- touristAttractions: TouristAttraction[]
Mobile Tourist Guide

<table>
<thead>
<tr>
<th>Name</th>
<th>searchTouristAttractions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Searches for tourist attractions depending on the user’s preferences and the current context (e.g. location, time, weather)</td>
</tr>
<tr>
<td>Receives</td>
<td>-</td>
</tr>
<tr>
<td>Returns</td>
<td>TouristAttractionList</td>
</tr>
<tr>
<td>Sends</td>
<td>TouristGuideService.getTouristAttraction()</td>
</tr>
<tr>
<td>Reads</td>
<td>ContextSet, Preferences</td>
</tr>
<tr>
<td>Changes</td>
<td>TouristAttractionList</td>
</tr>
<tr>
<td>Body</td>
<td>-</td>
</tr>
<tr>
<td>Precondition</td>
<td>Preferences have been set up, Context Sources are available</td>
</tr>
<tr>
<td>Postcondition</td>
<td>TouristAttractionList contains suitable attractions</td>
</tr>
</tbody>
</table>
Mobile Tourist Guide
## Mobile Tourist Guide

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
<th>What visibility should be assignable to context items on the mobile client? (Public context items are transmitted to the server, private context items not)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>MobileTouristGuide</td>
<td>MobileTouristGuide</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>Private</td>
<td>Private</td>
</tr>
<tr>
<td>Projection</td>
<td>Structural</td>
<td>Structural</td>
</tr>
<tr>
<td>Constraints</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>ResolutionSet</td>
<td>Boolean</td>
<td>ValueSet {Public, Private, PublicAndPrivate}</td>
</tr>
<tr>
<td>Effects</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ResolutionValue: <strong>True</strong></td>
<td>ResolutionValue: <strong>Public</strong></td>
</tr>
<tr>
<td></td>
<td>1. remove stereotype &lt;&lt;variant&gt;&gt; at Class Video</td>
<td>1. remove &lt;&lt;Komponent&gt;&gt; PrivateContextMatcher</td>
</tr>
<tr>
<td></td>
<td>2. remove stereotype &lt;&lt;variant&gt;&gt; on operation MobileTouristGuide.getVideo()</td>
<td>2. remove association PrivateContextMatcher - MobileTouristGuide</td>
</tr>
<tr>
<td>ResolutionValue: <strong>False</strong></td>
<td></td>
<td>ResolutionValue: <strong>Private</strong></td>
</tr>
<tr>
<td></td>
<td>1. remove Class Video</td>
<td>1. remove stereotype &lt;&lt;variant&gt;&gt; at Komponent PrivateContextMatcher</td>
</tr>
<tr>
<td></td>
<td>2. remove operation MobileTouristGuide.getVideo()</td>
<td>2. remove association PrivateContextMatcher - PrivateContextMatcher</td>
</tr>
<tr>
<td>ResolutionValue: <strong>PublicAndPrivate</strong></td>
<td></td>
<td>ResolutionValue: <strong>PublicAndPrivate</strong></td>
</tr>
<tr>
<td></td>
<td>1. remove stereotype &lt;&lt;variant&gt;&gt; at Komponent PrivateContextMatcher</td>
<td></td>
</tr>
</tbody>
</table>

| Stakeholder | Application Engineer | Application Engineer |
Context Manager
<table>
<thead>
<tr>
<th><strong>ID</strong></th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Is the mobile client device capable of playing videos?</td>
</tr>
<tr>
<td><strong>Component</strong></td>
<td>ContextManager</td>
</tr>
<tr>
<td><strong>Abstraction</strong></td>
<td>Specification</td>
</tr>
<tr>
<td><strong>Projection</strong></td>
<td>Structural</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>--</td>
</tr>
<tr>
<td><strong>ResolutionSet</strong></td>
<td>Boolean</td>
</tr>
</tbody>
</table>
| **Effects** | ResolutionValue == True  
(1) remove stereotype <<variant>> at Komponent 3rdPartyContextServices  
ResolutionValue == False  
(1) remove Komponent 3rdPartyContextServices  
(2) remove association 3rdPartyContextServices-MobileTouristGuide |
| **Stakeholder** | Application Engineer |
Dimension relationships revisited

Variation
- family
- product 1
- product 2
- ...

Platform specificity
- platform independent
- platform specific 1
- platform specific 2
- ...

Component
- component 1
- component 2
- component 3
- ...

Version
- version 1
- version 2
- version 3
- ...

Visibility
- black box
- grey box
- white box

Projection
- structural
- operational
- dynamic
- behavioural
- variational
- non-functional

Operation
- operation 1
- operation 2
- operation 3

Perspective
- UML class diagram
- taxonomy
- ...

Notation (Concrete Syntax)
- graphical
- textual
- architecture
- design
Shopping Cart Example

```
ShoppingCartService

createNewCart (username : String, password : String)  :  String
addProduct (cartID : String,  p : Product,  number : Integer)
removeProduct (cartID : String,  p : Product)
checkout(cartID : String, card : CreditCard)

totalCost (cartID : String) : Integer
numberOfItems (cartID : String, p  : Product) : Integer
numberOfProducts (cartID : String) : Integer
isCheckedOut (cartID : String) : Boolean

ShoppingCart

ID : String
/totalCost: Integer
checkedOut : Boolean

addItem (p : Product, price : Integer, number : Integer)
removeItem (p : Product)
totalCost () : Integer
numberofProduct () : Integer
numberOfItems (p : Product)
confirmPurchase ()
isCheckedOut () : Boolean

Inv: limit >= 0

Product

ID : String
price : Integer
description : String
noOfItems : Integer
```
## Algebraic Specification

<table>
<thead>
<tr>
<th>Operation</th>
<th>totalCost</th>
<th>numberOfItems</th>
<th>numberOfProducts</th>
<th>isCheckedOut</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>createNewCart</strong></td>
<td>totalCost (createNewCart) = 0</td>
<td>totalNumberOfItems (createNewCart, P) = 0</td>
<td>numberProducts (createNewCart) = 0</td>
<td>isCheckedOut (createNewCart) = FALSE</td>
</tr>
<tr>
<td><strong>addProduct</strong></td>
<td>totalCost (addProduct (C, P, N)) = totalCost (C) + P.price * N</td>
<td>numberOfItems (addProduct (C, P, N), J) =</td>
<td>numberOfProducts (addProduct (C, P, N)) =</td>
<td>isCheckedOut (addProduct (C, P, N)) = isCheckedOut (C)</td>
</tr>
<tr>
<td><strong>removeProduct</strong></td>
<td>totalCost (removeProduct (C, I)) = totalCost (C) - numberOfItems (C, P) * P.price</td>
<td>numberOfItems (removeProduct (C, P, Q)) = if P = Q 0 else numberOfItems (J)</td>
<td>numberOfProducts (removeProduct (C, P)) = if noOfItems(C, P) = 0 noOfProducts (C) else numberOfProducts (C) - 1</td>
<td>isCheckedOut (removeProduct (C, P)) = isCheckedOut (C)</td>
</tr>
<tr>
<td><strong>checkout</strong></td>
<td>totalCost (checkout (C, CC)) = totalCost (C)</td>
<td>numberOfItems (checkout (C, CC)) = numberOfItems (C)</td>
<td>numberOfProducts (checkout (C, CC)) = numberOfProducts (C)</td>
<td>isCheckedOut (checkout (C, CC)) = if CC.limit &lt; totalCost (C) FALSE</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------</td>
<td>---------------------------</td>
<td>---------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>createNewCart</td>
<td>“testUser”</td>
<td>“testpwd”</td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>totalcost</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfItems</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfProducts</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>isCheckedOut</td>
<td>F1</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>Product</td>
<td>create</td>
<td>“768493”</td>
<td>250</td>
<td>“Harry Potter”</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>addProduct</td>
<td>F1</td>
<td>F6</td>
<td>3</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>totalcost</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfItems</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfProducts</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>isCheckedOut</td>
<td>F1</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>Product</td>
<td>create</td>
<td>“768493”</td>
<td>250</td>
<td>“Harry Potter”</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>totalcost</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfItems</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfProducts</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>isCheckedOut</td>
<td>F1</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>CreditCard</td>
<td>create</td>
<td>“TestUser”</td>
<td>“6GB38282”</td>
<td>500</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>checkOut</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>totalcost</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfItems</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>isCheckedOut</td>
<td>F1</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>removeProduct</td>
<td>F1</td>
<td>F6</td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>totalcost</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfItems</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfProducts</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>isCheckedOut</td>
<td>F1</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>Product</td>
<td>create</td>
<td>“784392”</td>
<td>34</td>
<td>“Star Wars”</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>removeProduct</td>
<td>F1</td>
<td>F28</td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>totalCost</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>NumberOfItems</td>
<td>F1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>numberOfProducts</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>isCheckedOut</td>
<td>F1</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>ShoppingCartService</td>
<td>isCheckedOut</td>
<td>F1</td>
<td></td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Auction House Example

<<subject>> AuctionHouse

- activeAuctions
- maxAuctions
- register()
- login()
- logout()
- joinAuction()
- leaveAuction()
- bid()
- offerItem()
- authorizePayment()
- setMaxAuctions()

<<configuration>>

<<subject>>

<<component>> AuctionManager

- joinAuction()
- leaveAuction()
- bid()

<<component>>

<<subject>>

<<component>>

<<component>>

<<component>>

<<component>>

<<subject>>

<<component>>

<<component>>

<<component>>

<<component>>

logBid()
logLogin()
logRegister()

logBid()
logLogin()
logRegister()

logBid()
logLogin()
logRegister()

logBid()
logLogin()
logRegister()

logBid()
logLogin()
logRegister()

logBid()
logLogin()
logRegister()

logBid()
logLogin()
logRegister()

logBid()
logLogin()
logRegister()

logBid()
logLogin()
logRegister()
Auction House Specification Behavioral Model

uninitialized \[\text{initialized and tested}\] \rightarrow \text{inService}

\begin{itemize}
  \item \text{idle}
    \begin{itemize}
      \item register()
      \item login()
      \item logout()
    \end{itemize}
  \item \text{active}
    \begin{itemize}
      \item \text{offerItem()}\newline\text{[activeAuctions = 1]}
      \item \text{authorizePayment()}
      \item \text{offerItem()}\newline\text{[activeAuctions + 1 < maxAuctions]}
      \item \text{authorizePayment()}
      \item register()
      \item login()
      \item logout()
      \item joinAuction()
      \item leaveAuction()
      \item bid()
    \end{itemize}
\end{itemize}
Making the Interface Testable

- “functionallize” all semantic information in the interface
  - define methods to make it accessible by clients

- four main sources
  - logical attributes of the service
  - logical states of the service
  - non-functional requirements
  - configuration settings

- combined functional and testable interface is known as the “extended interface”
Auction House Interface Enhancements

**Logical Attributes**

- setActiveAuctions()
- getActiveAuctions()

**Logical State**

- setIdle()
- isIdle()
- setActive()
- isActive()
- setIsInService()
- isInService()
Orthographic modeling offers a simple but powerful paradigm for supporting the view based development of software systems.

\[
\text{Orthographic System Modeling} = \quad \text{On-demand generation from single underlying model} \\
\text{view generation mechanism} + \quad \text{Orthographic (dimension-based)} \\
\text{navigation paradigm} + \quad \text{KobrA 2.0} \\
\text{View-based method}
\]

- an effective way of supporting “living models”
  - when accessible at run-time, the SUM provides the basis for “semantics driven” applications
  - when enhanced with non-functional information, the approach provides the basis for quality engineering (Enterprise Architecture)
Questions