72. Orthographic Software Modeling (OSM) with Single Underlying Model (SUM) - A 1-TS-Megamodel with Total Consistency

Prof. Dr. U. Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

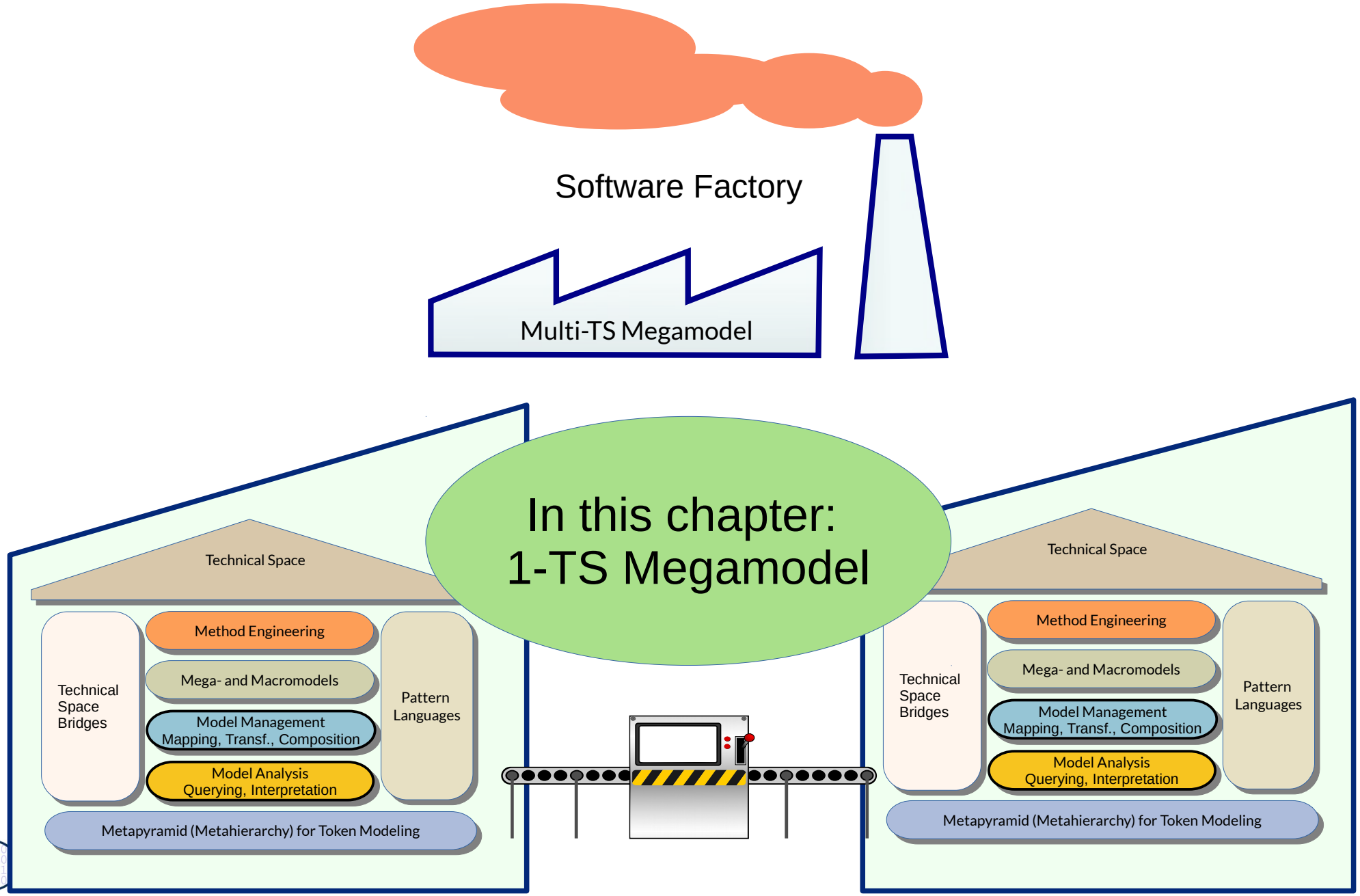
http://st.inf.tu-dresden.de/teaching/most

Version 17-0.1, 27.01.18

1) Orthographic Software Modeling (OSM) and Single Underlying Model (SUM)
2) Lenses

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Q11: A Software Factory's Heart: the Multi-TS Megamodel



References

- Zinovy Diskin and Yingfei Xiong and Krzysztof Czarnecki. From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case. Journal of Object Technology, 2011, vol. 10, 6, pp. 1-25,
  - http://dx.doi.org/10.5381/jot.2011.10.1.a6
- J. Nathan Foster and Michael B. Greenwald and Jonathan T. Moore and Benjamin C. Pierce and Alan Schmitt. Combinators for Bi-Directional Tree Transformations: A Linguistic Approach to the View Update Problem, ACM Transactions on Programming Languages and Systems, Vol 29(3), pp. 17, 2007
  - http://www.cis.upenn.edu/~bcpierce/papers/newlenses-popl.pdf

Synchronization of Projective Views on a Single Underlying Model (A Orthographic Macromodel)

These slides are courtesy to:
Christian Vjekoslav Tunjic
Colin Atkinson

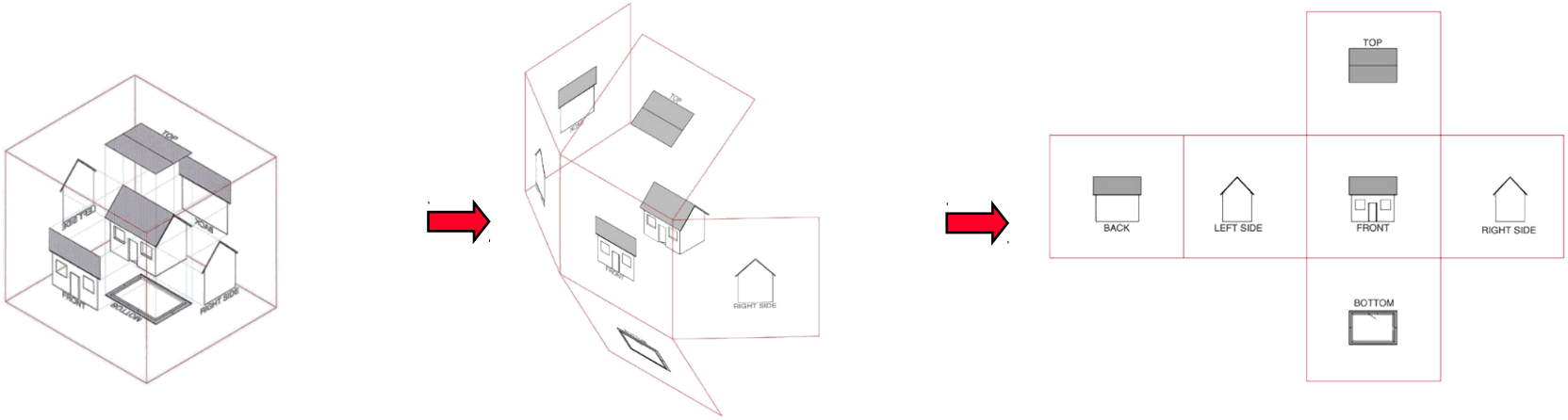
Presented at: VAO 2015

L'Aquila. Italy
21 July, 2015

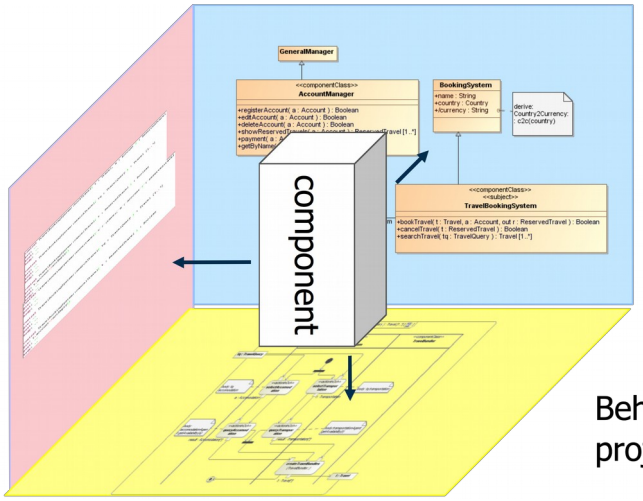
UNIVERSITÄT
MANNHEIM

72.1 Orthographic Software Modeling (OSM)

- other engineering disciplines have a long and successful tradition of technical drawing - orthographic projection



- so why don't we do this in software engineering?



Structural projection

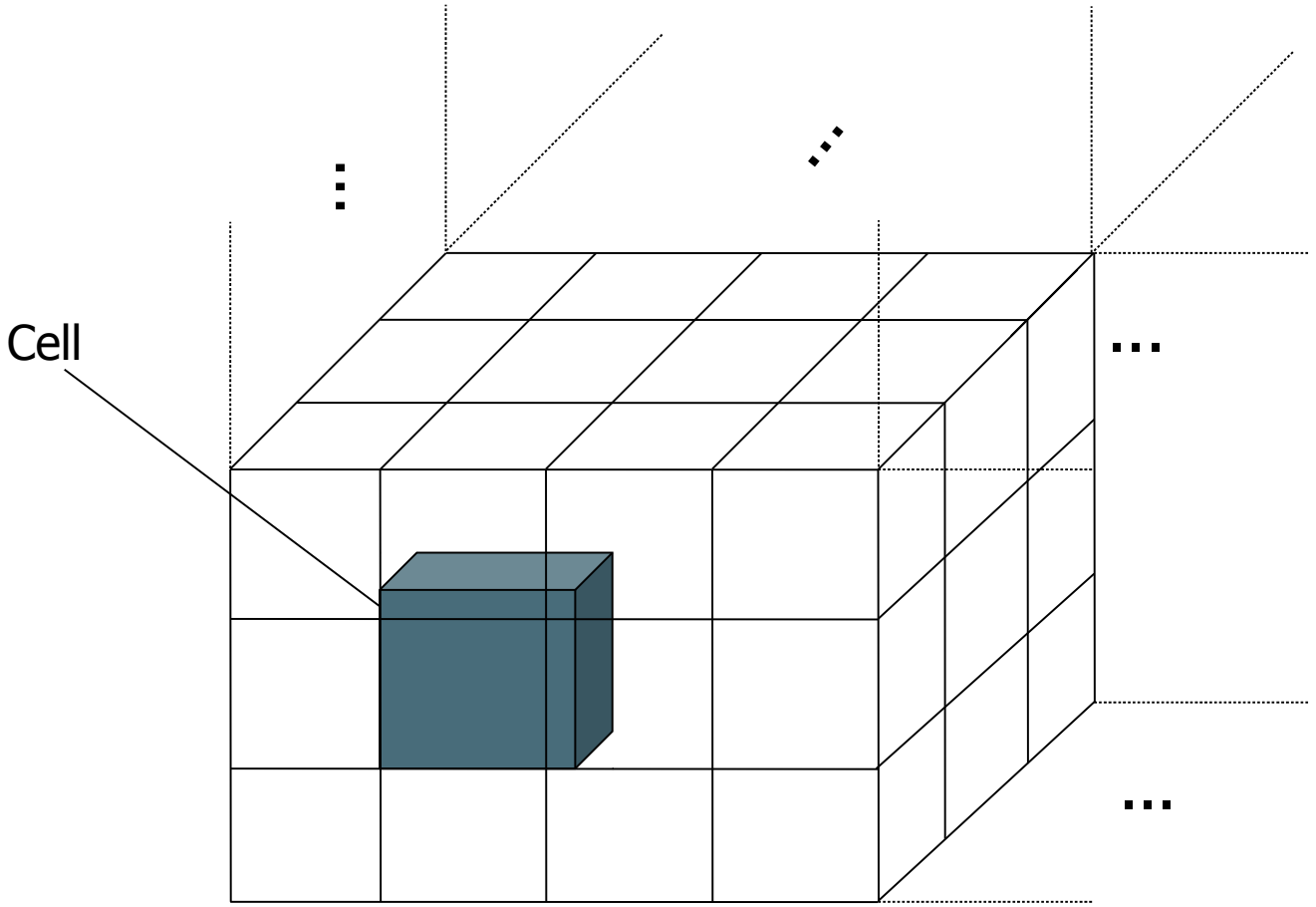
Operational projection

Behavioral projection

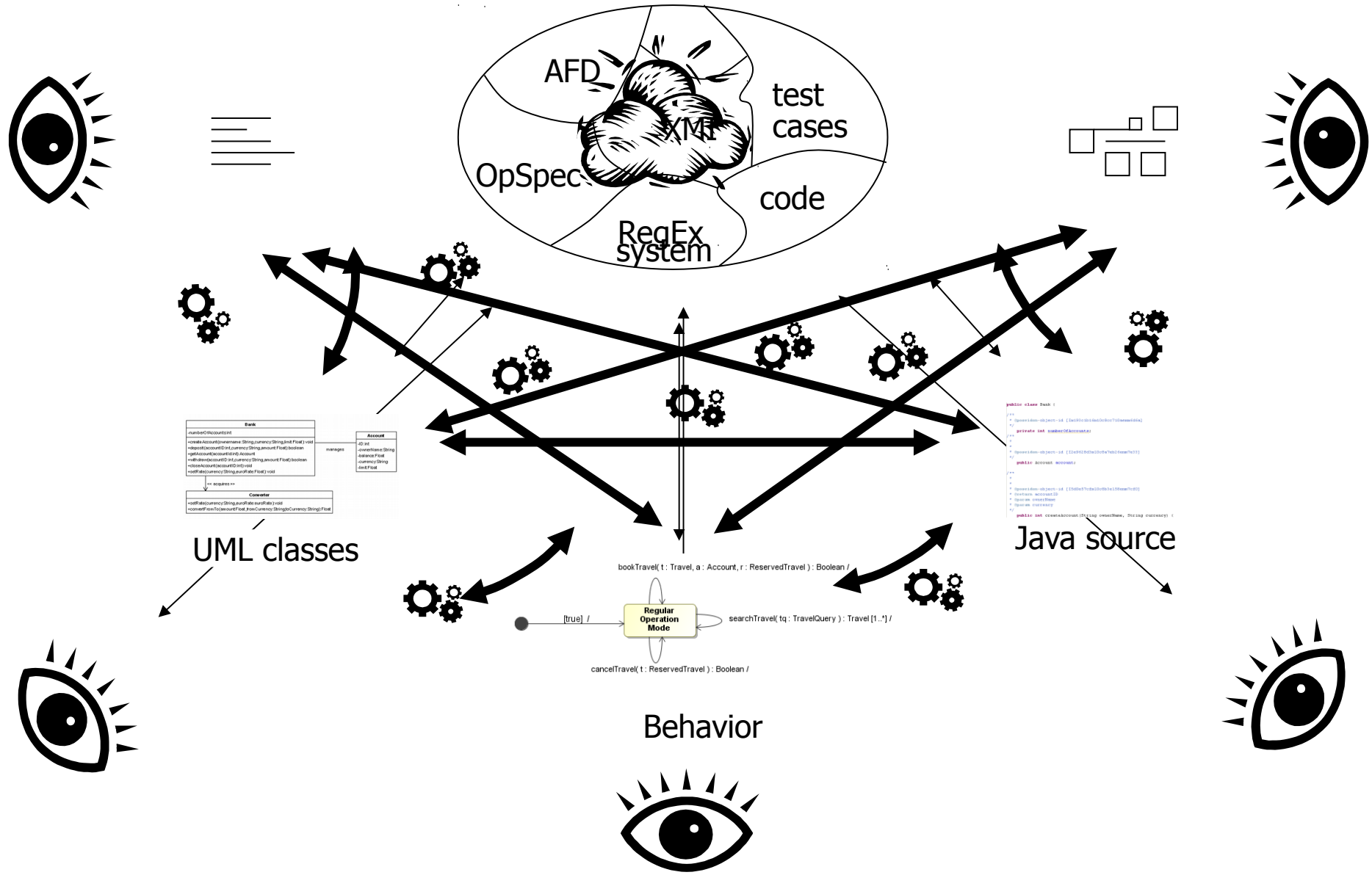
- On demand view generation (projective views)
- Dimension-based navigation
- View-based methodology

Dimension Based Navigation

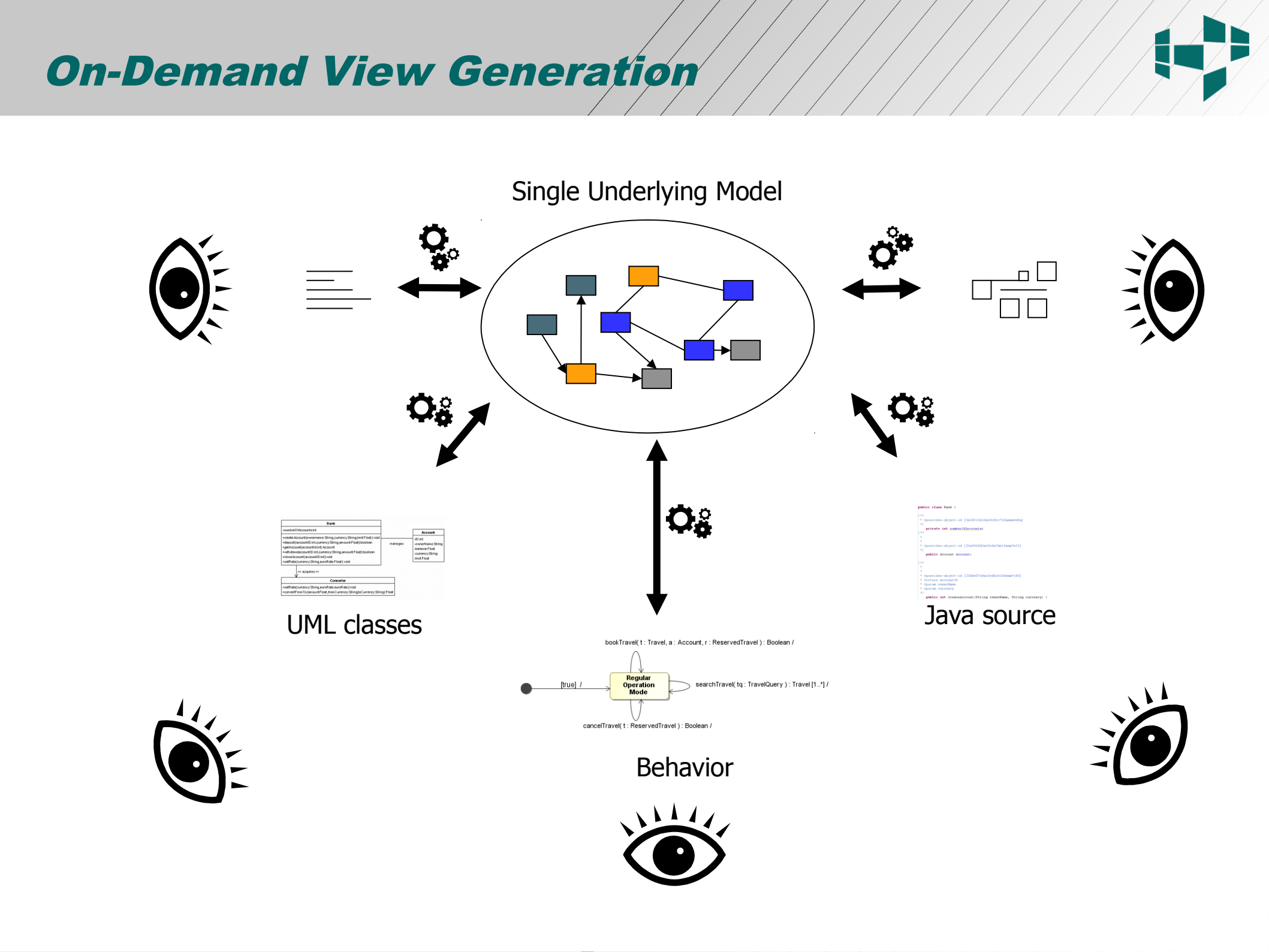
- views organized in a multi-dimensional cube
- one choice always “selected” from each dimension
- each cell represents a viewpoint

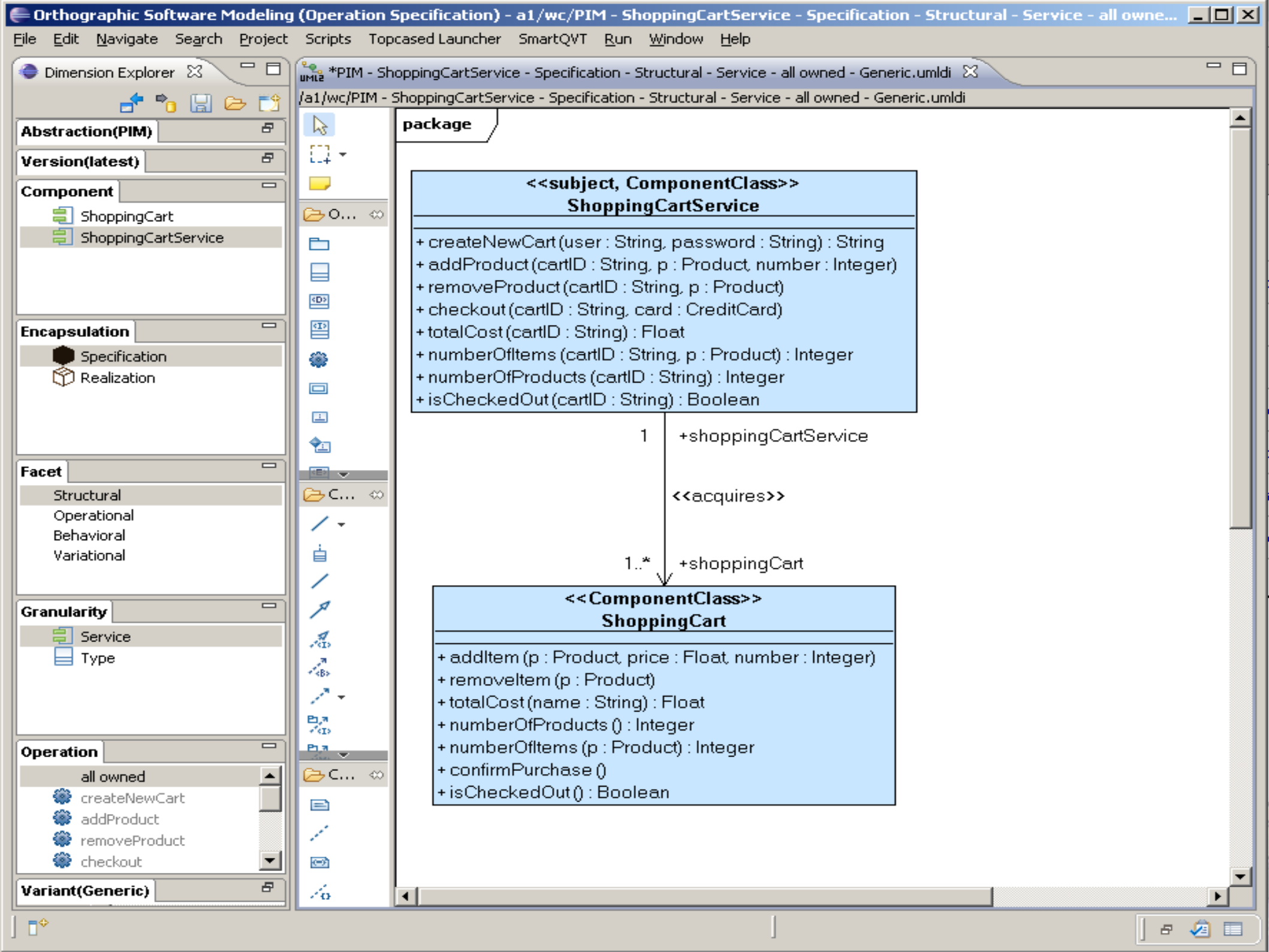


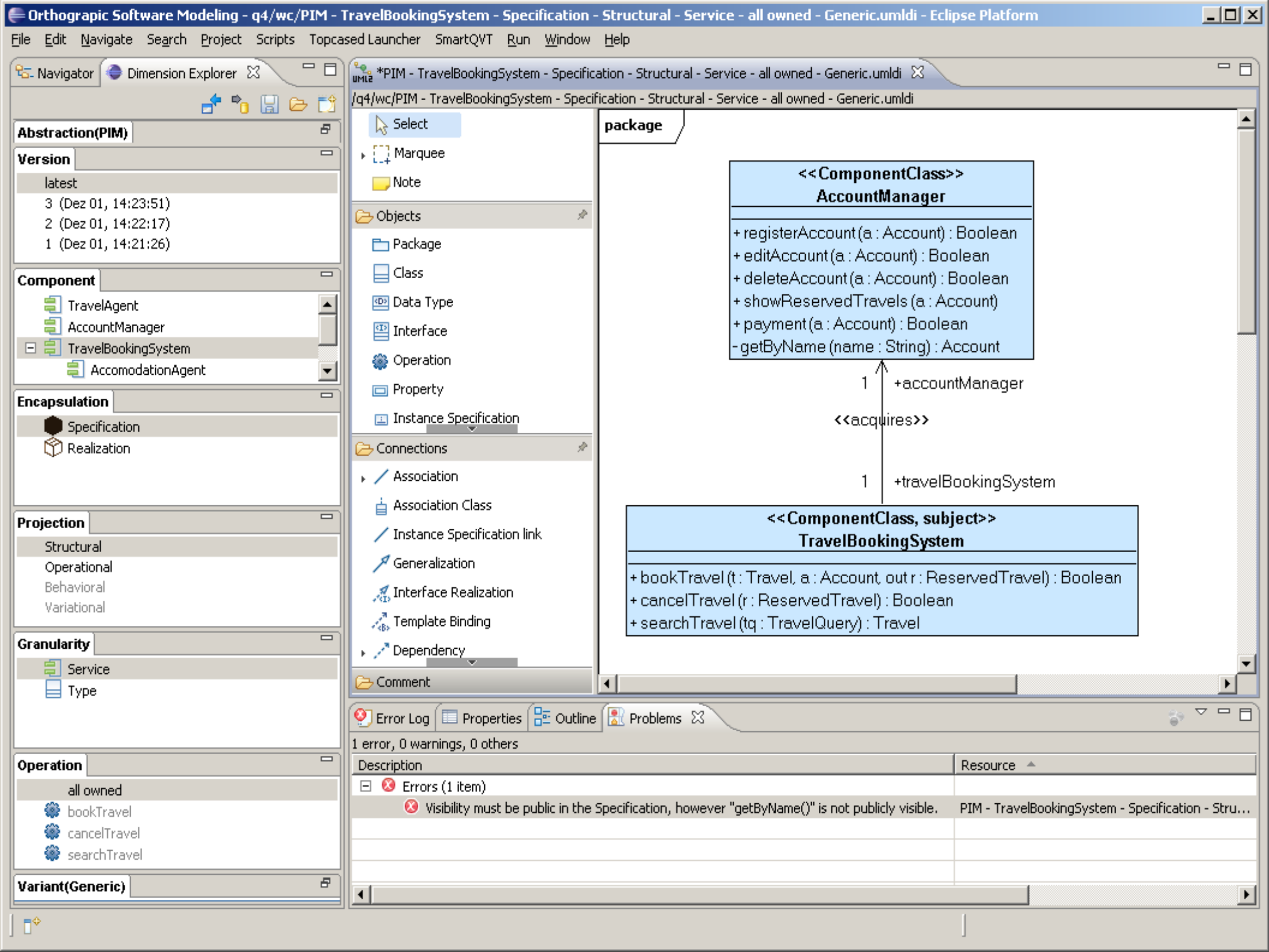
Traditional View-based Environment

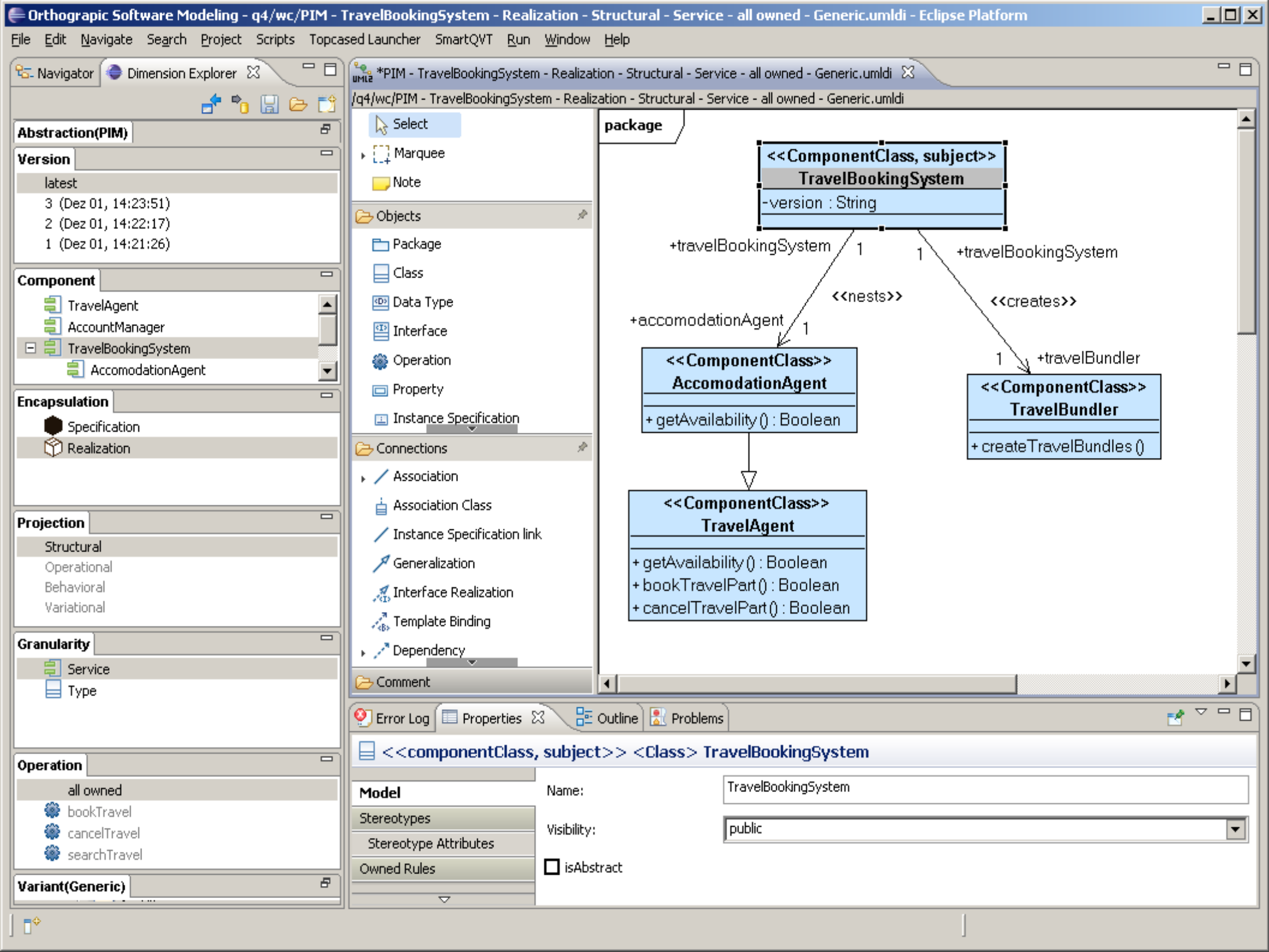


On-Demand View Generation




Orthographic Software Modeling (Operation Specification) - a1/wc/PIM - ShoppingCartService - Specification - Structural - Service - all owne...
File Edit Navigate Search Project Scripts Topcased Launcher SmartQVT Run Window Help
Dimension Explorer
*PIM - ShoppingCartService - Specification - Structural - Service - all owned - Generic.umldi
/a1/wc/PIM - ShoppingCartService - Specification - Structural - Service - all owned - Generic.umldi
Abstraction(PIM)
Version(latest)
Component
ShoppingCart
ShoppingCartService
Encapsulation
Specification
Realization
Facet
Structural
Operational
Behavioral
Variational
Granularity
Service
Type
Operation
all owned
createNewCart
addProduct
removeProduct
checkout
Variant(Generic)
package
<<subject, ComponentClass>>
ShoppingCartService
+ createNewCart (user : String, password : String) : String
+ addProduct (cartID : String, p : Product, number : Integer)
+ removeProduct (cartID : String, p : Product)
+ checkout (cartID : String, card : CreditCard)
+ totalCost (cartID : String) : Float
+ numberOfItems (cartID : String, p : Product) : Integer
+ numberOfProducts (cartID : String) : Integer
+ isCheckedOut (cartID : String) : Boolean
1
+shoppingCartService
<<acquires>>
1..*
+shoppingCart
<<ComponentClass>>
ShoppingCart
+ addItem (p : Product, price : Float, number : Integer)
+ removeItem (p : Product)
+ totalCost (name : String) : Float
+ numberOfProducts () : Integer
+ numberOfItems (p : Product) : Integer
+ confirmPurchase ()
+ isCheckedOut () : Boolean


Orthograpic Software Modeling - q4/wc/PIM - TravelBookingSystem - Specification - Structural - Service - all owned - Generic.umldi - Eclipse Platform
File Edit Navigate Search Project Scripts Topcased Launcher SmartQVT Run Window Help
Navigator
Dimension Explorer
Abstraction(PIM)
Version
latest
3 (Dez 01, 14:23:51)
2 (Dez 01, 14:22:17)
1 (Dez 01, 14:21:26)
Component
TravelAgent
AccountManager
TravelBookingSystem
AccomodationAgent
Encapsulation
Specification
Realization
Projection
Structural
Operational
Behavioral
Variational
Granularity
Service
Type
Operation
all owned
bookTravel
cancelTravel
searchTravel
Variant(Generic)
*PIM - TravelBookingSystem - Specification - Structural - Service - all owned - Generic.umldi
/q4/wc/PIM - TravelBookingSystem - Specification - Structural - Service - all owned - Generic.umldi
Select
Marquee
Note
Objects
Package
Class
Data Type
Interface
Operation
Property
Instance Specification
Connections
Association
Association Class
Instance Specification link
Generalization
Interface Realization
Template Binding
Dependency
Comment
package
<<ComponentClass>>
AccountManager
+ registerAccount (a : Account) : Boolean
+ editAccount (a : Account) : Boolean
+ deleteAccount (a : Account) : Boolean
+ showReservedTravels (a : Account)
+ payment (a : Account) : Boolean
- getByName (name : String) : Account
1
+accountManager
<<acquires>>
1
+travelBookingSystem
<<ComponentClass, subject>>
TravelBookingSystem
+ bookTravel (t : Travel, a : Account, out r : ReservedTravel) : Boolean
+ cancelTravel (r : ReservedTravel) : Boolean
+ searchTravel (tq : TravelQuery) : Travel
Error Log
Properties
Outline
Problems
1 error, 0 warnings, 0 others
Description
Resource
Errors (1 item)
Visibility must be public in the Specification, however "getByName()" is not publicly visible.
PIM - TravelBookingSystem - Specification - Stru...


Orthograpic Software Modeling - q4/wc/PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi - Eclipse Platform
File Edit Navigate Search Project Scripts Topcased Launcher SmartQVT Run Window Help
Navigator
Dimension Explorer
Abstraction(PIM)
Version
latest
3 (Dez 01, 14:23:51)
2 (Dez 01, 14:22:17)
1 (Dez 01, 14:21:26)
Component
TravelAgent
AccountManager
TravelBookingSystem
AccomodationAgent
Encapsulation
Specification
Realization
Projection
Structural
Operational
Behavioral
Variational
Granularity
Service
Type
Operation
all owned
bookTravel
cancelTravel
searchTravel
Variant(Generic)
*PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi
/q4/wc/PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi
Select
Marquee
Note
Objects
Package
Class
Data Type
Interface
Operation
Property
Instance Specification
Connections
Association
Association Class
Instance Specification link
Generalization
Interface Realization
Template Binding
Dependency
Comment
package
<<ComponentClass, subject>>
TravelBookingSystem
-version : String
+travelBookingSystem
1
1
+travelBookingSystem
<<nests>>
<<creates>>
+accomodationAgent
1
1
+travelBundler
<<ComponentClass>>
AccomodationAgent
+ getAvailability () : Boolean
<<ComponentClass>>
TravelBundler
+ createTravelBundles ()
<<ComponentClass>>
TravelAgent
+ getAvailability () : Boolean
+ bookTravelPart () : Boolean
+ cancelTravelPart () : Boolean
Error Log
Properties
Outline
Problems
<<componentClass, subject>> <Class> TravelBookingSystem
Model
Stereotypes
Stereotype Attributes
Owned Rules
Name:
TravelBookingSystem
Visibility:
public
isAbstract

Scalability and Applicability

- An approach needs to be applicable to more than just a toy example
- An approach must scalable for the chosen field of applicabilty
- Simple minded implementation approach –
  - uni-directional transformations (SUM-to-view, view-to-SUM)
  - create a new (version of the) view whenever there is a change in the SUM
  - create a new (version of the) SUM whenever there is a change in a view
- Would work but -
  - not scalable (inefficient)
  - transformation more complex than necessary
  - too large grained

⇒ Delta-based bidirectional lenses

72.2 Delta-Based Lenses

- Lenses (Pierce et al. 2007) are bidirectional transformations based on get (projectiion, checkout) and put (integration, checkin) operations
  - axioms for *well-behaved lenses*

v: View; s:SUM
get(put(v, s)) = v // *PUTGET invariant rule*
put(get(s), s) = s // *GETPUT invariant rule*

  - axiom for *very well behaved lenses*

put(v', put(v, s)) = put(v', s) // *PUTPUT invariant rule*

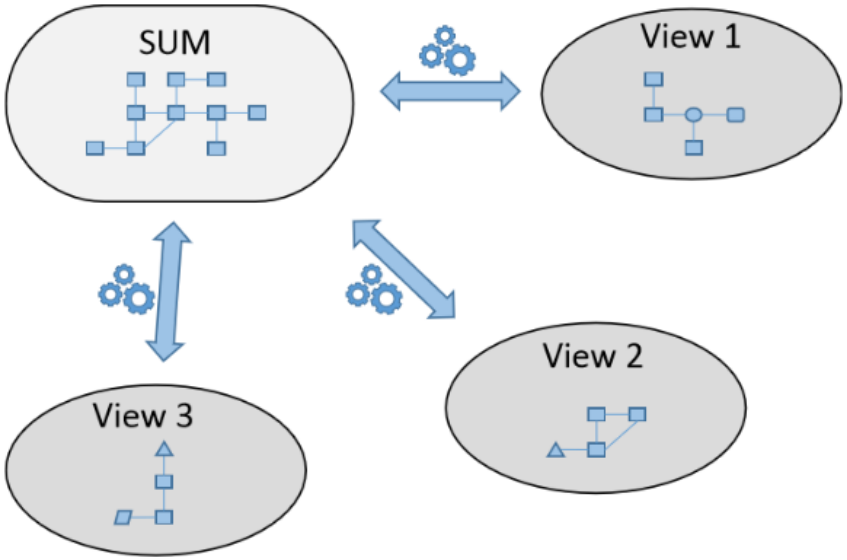
- Delta-based Lenses optimize the checkin/checkout (Diskin et al. 2011)
  - dput and dget operations driven by the changes to the views
  - avoids problems with the *PUTPUT* rule

if $\Delta s = dput(\Delta v, s)$, then $dget(\Delta s) = \Delta v$ ≡ *DeltaPUTPUT rule*

  - much more fine-grained and scalable

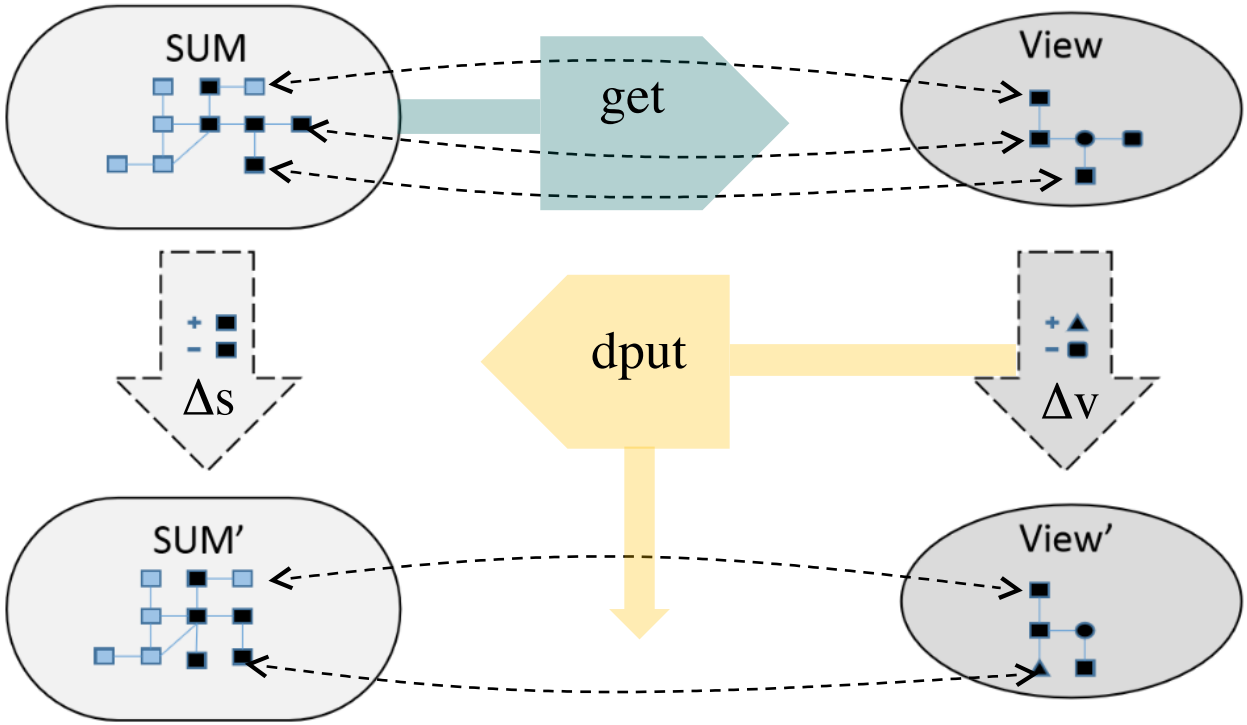
OSM Context

- The SUM is much larger than the views
  - the views are relatively small and compact
- Views can be updated concurrently
  - axioms only applicable locally (i.e. to one view at a time)
- Usually have one-to-one correspondences between view elements and SUM elements
  - changes can conveniently be traced to the affected element
- View elements cannot be changed just locally
  - for example, cannot delete an element from just the view, but not the SUM



Hybrid Approach

- use ***get*** to create views from the SUM
- use ***dput*** to update the SUM when a view is changed



Pros and Cons

- **TGG** can be used to specify put/get and dput/dget combinators
- **Traces** allow affected SUM elements to be efficiently identified
  - can be generated most mainstream transformation engines
- Traces also allow the open views impacted by a change to be identified
  - must be updated dynamically a la MVC pattern
- Use of ***get*** to create views reduces the complexity of the transformation with little extra overhead
  - no need to update trace information
- Use of ***dput*** to update the SUM greatly enhances the efficiency of updating SUM
  - the SUM is only ever updated via changes to views
- However, it increases the amount of information that needs to be stored on the server
  - part of the SUM?

SUM on RoSI CROM

- The SUM principle can be played on all metalanguages, e.g., CROM
- Contexts provide *viewpoints*
- Roles provide *views*

- **Theorem:** If in a CROM-based SUM, a change is local to a context, then the change fulfils the delta-putput invariant

Conclusion

- Work in progress…. !
- Related work
  - Inclusion of correspondences suggests connection to Triple Graph Grammars (definition of completeness, correctness etc.)
  - Vitruvius (change objects, projectional scope …)
- Challenges
  - determine appropriate laws in a multi-view context -
    - e.g. when does PUTPUT make sense?
  - accommodate many-to-many correspondences
- Possible enhancements
  - extend correspondence information with layout information to allow retainment of layout between view updates
  - allow local editing and manipulation of views
    - e.g. domain specific rendering