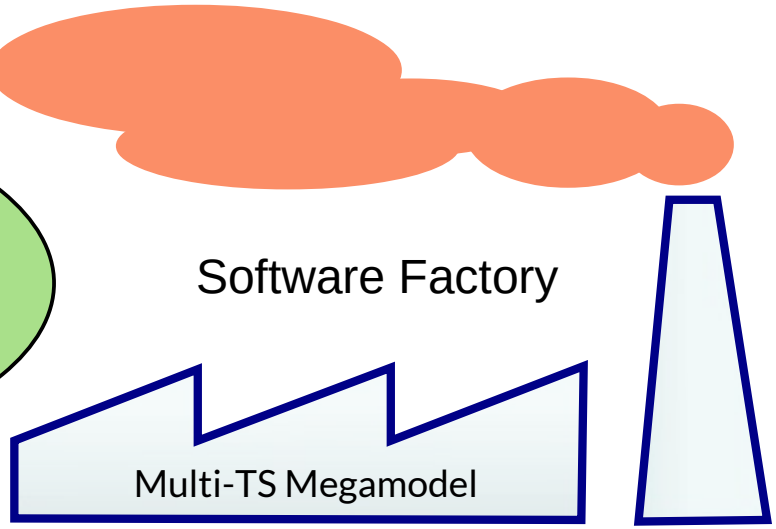
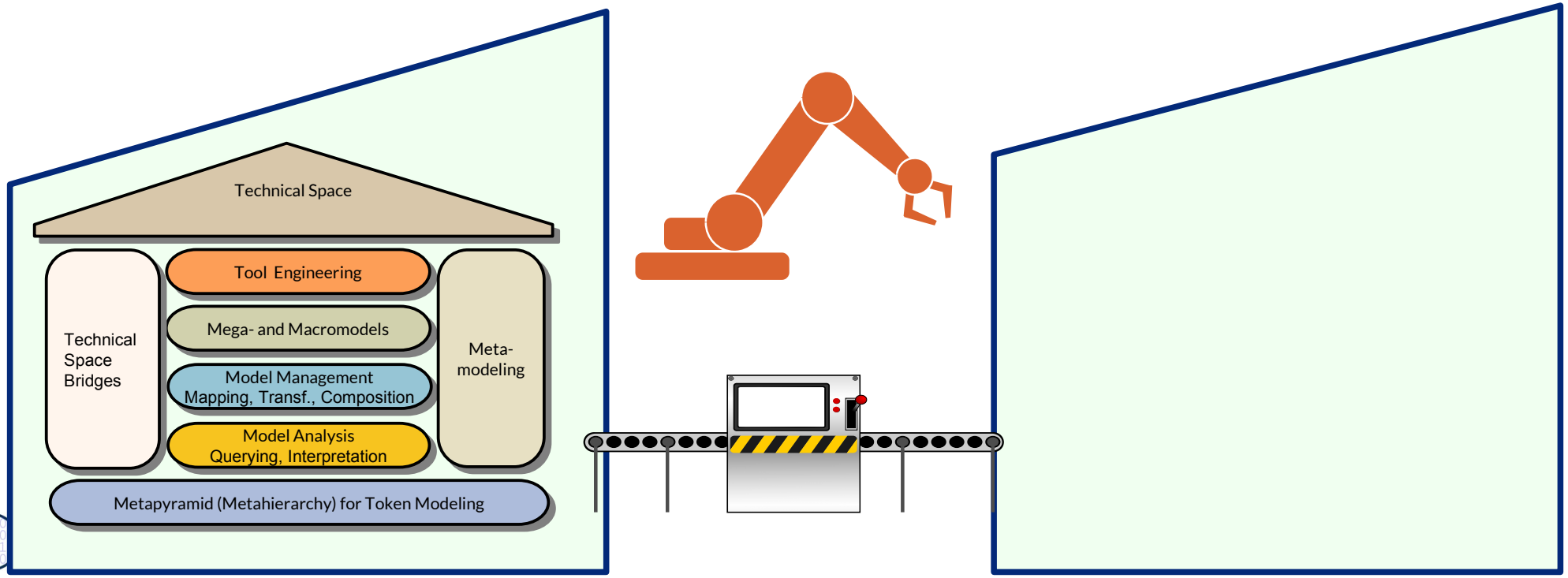
28. Megamodel Single Underlying Model (SUM)
with Orthographic Software Modeling (OSM) - A 1-TS-Megamodel with Total Consistency

Prof. Dr. U. Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
http://st.inf.tu-dresden.de/teaching/most
Version 19-1.1, 06.01.20

Software Factories with Only 1 Technical Space

In this chapter:
1-TS Megamodels
SUM





References

- Hettel, Thomas and Lawley, Michael J. and Raymond, Kerry (2008). Model Synchronisation: Definitions for Round-Trip Engineering. In Proceedings ICMT2008 - International Conference on Model Transformation: Theory and Practice of Model Transformations LNCS 5063/2008, pages pp. 31-45, Zurich, Switzerland.
- Thomas Hettel. Model Round-Trip Engineering. PhD Thesis. Queensland University of Technology, 2010
- 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

Overview Table for Macromodels

- RAGs are useful for all Macromodels, because they abbreviate dependencies in several models with cross-model relations.
  - In a macromodel under an artificial root (rooted macromodel), attributions can work on the SUM to ensure the constraints
- RelRAGs are useful, because they have bidirectional constraints

	MDA	Olympic (De)composition	Skeleton SUM (partial function extension)		General SUM	VSUM
				Orthographic Software Modeling (OSM)		
RAGs in Repositories	Markings	Marking of tree nodes	get/put as higher-order attributions			
RAGs in Data-flow architectures	Needs trace models	Slices	In-place transformations of SUM		get/put as model transformations	Works well to generate derived models

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

Many slides are courtesy to:
Christian Vjekoslav Tunjic,
Prof. Colin Atkinson

Used by permission

Presented at: VAO 2015

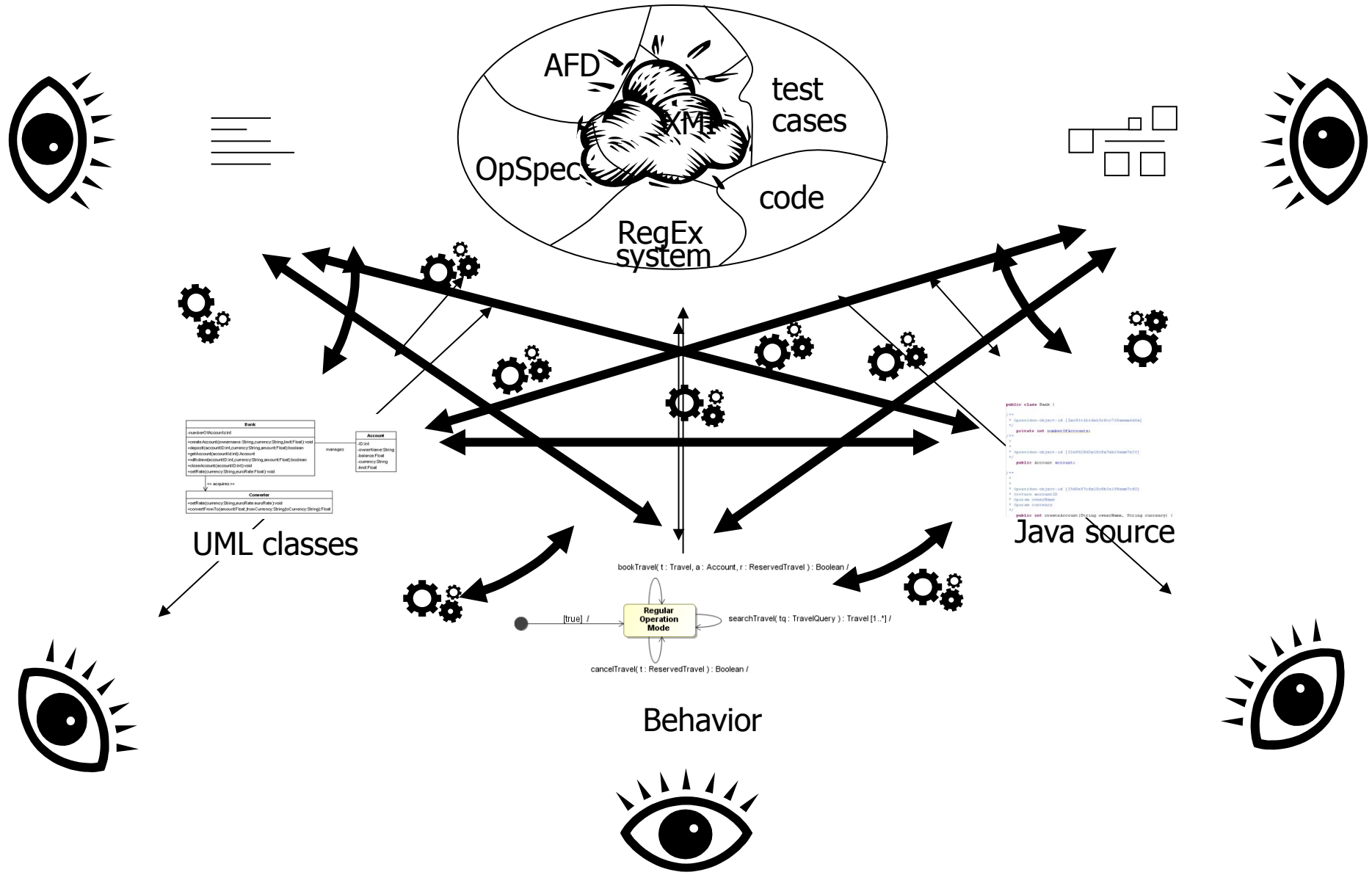
L'Aquila. Italy
21 July, 2015

UNIVERSITÄT MANNHEIM

28.1. The Megamodel “Single-Underlying Model (SUM)”

- [Atkinson]

Traditional View-based Development Environment



On-Demand View Generation in a SUM (Flat Contexts Correspond to Colors or Tags)

The SUM, if editable, provide a single-source view

Single Underlying Model (SUM) (all views merged)

Context 1

View 1

Requirements texts

View 5

Deployment diagrams

Context 5

View 2

UML classes

View 3

Behavior

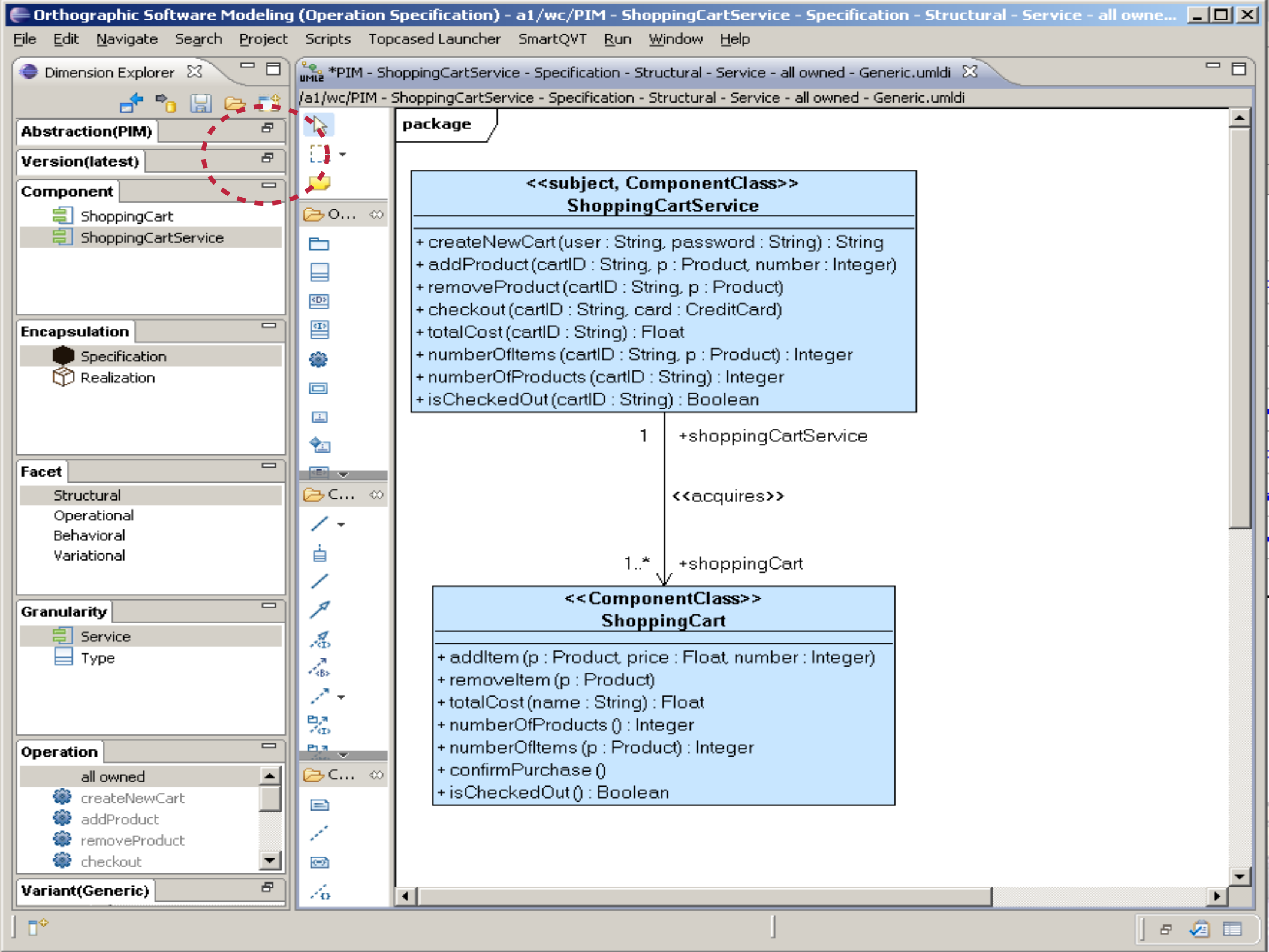
View 4

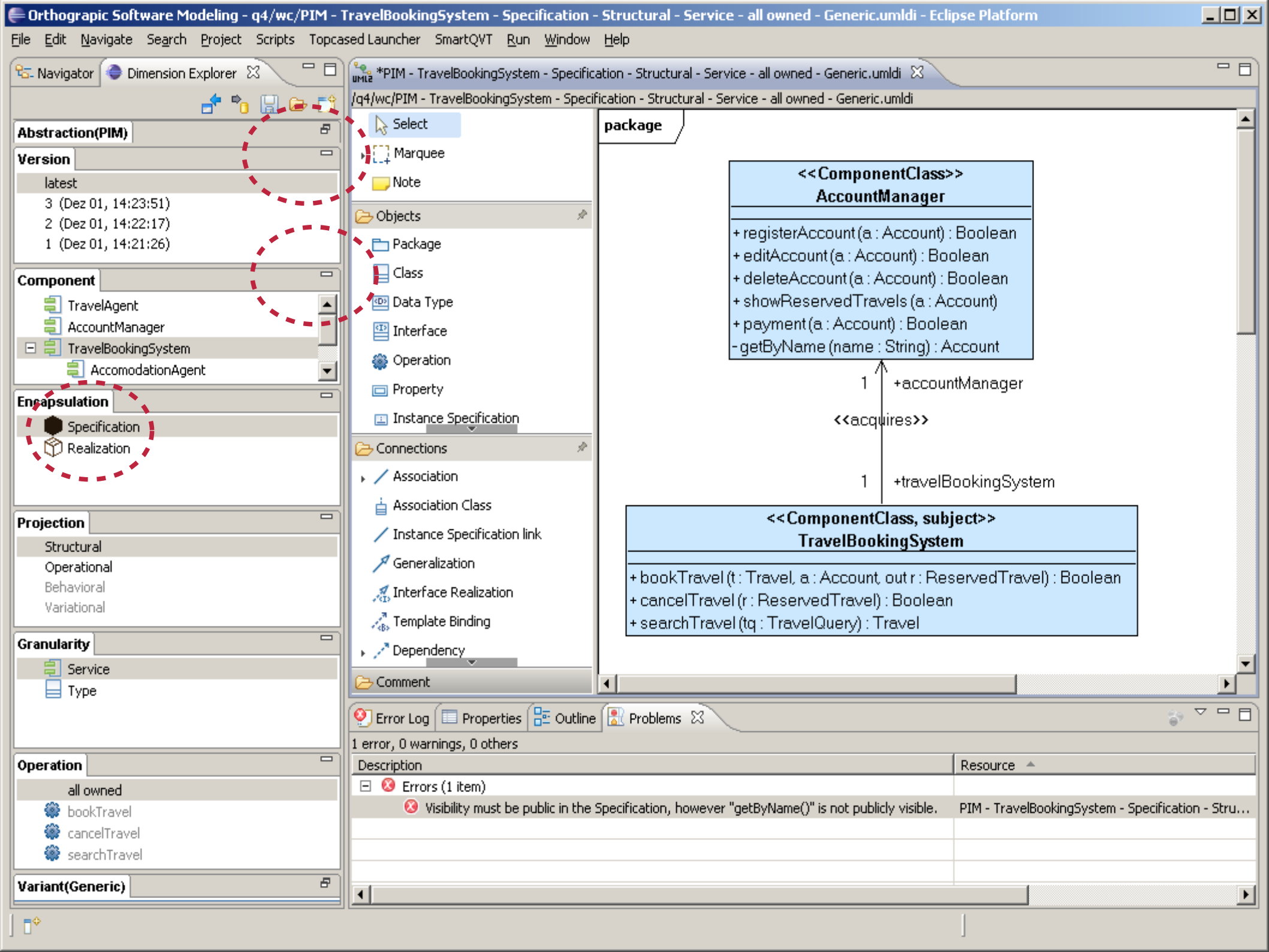
Java source

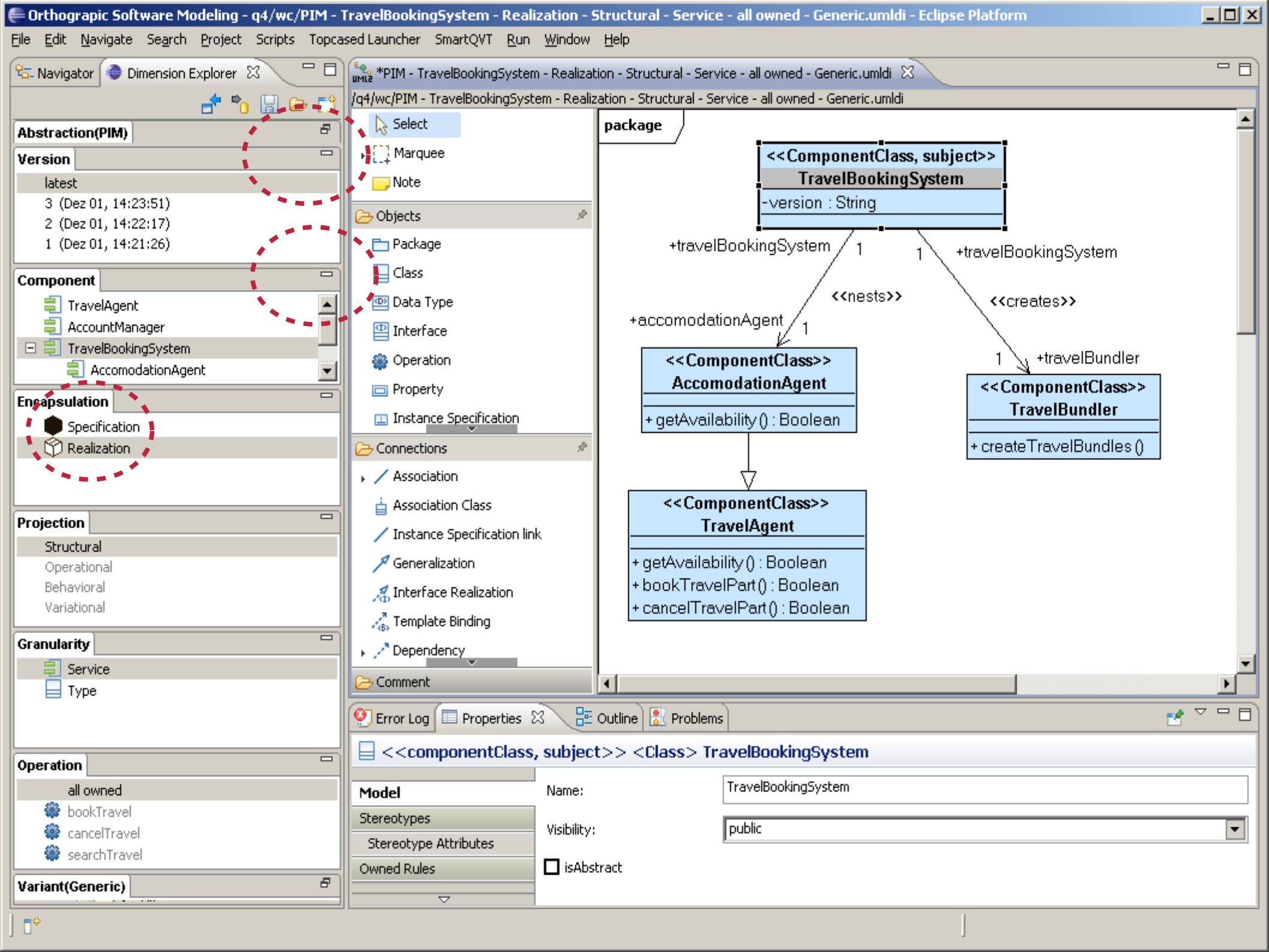
Context 2

Context 3

Context 4


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
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)
*PIM - ShoppingCartService - Specification - Structural - Service - all owned - Generic.umldi
/a1/wc/PIM - ShoppingCartService - Specification - Structural - Service - all owned - Generic.umldi
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...

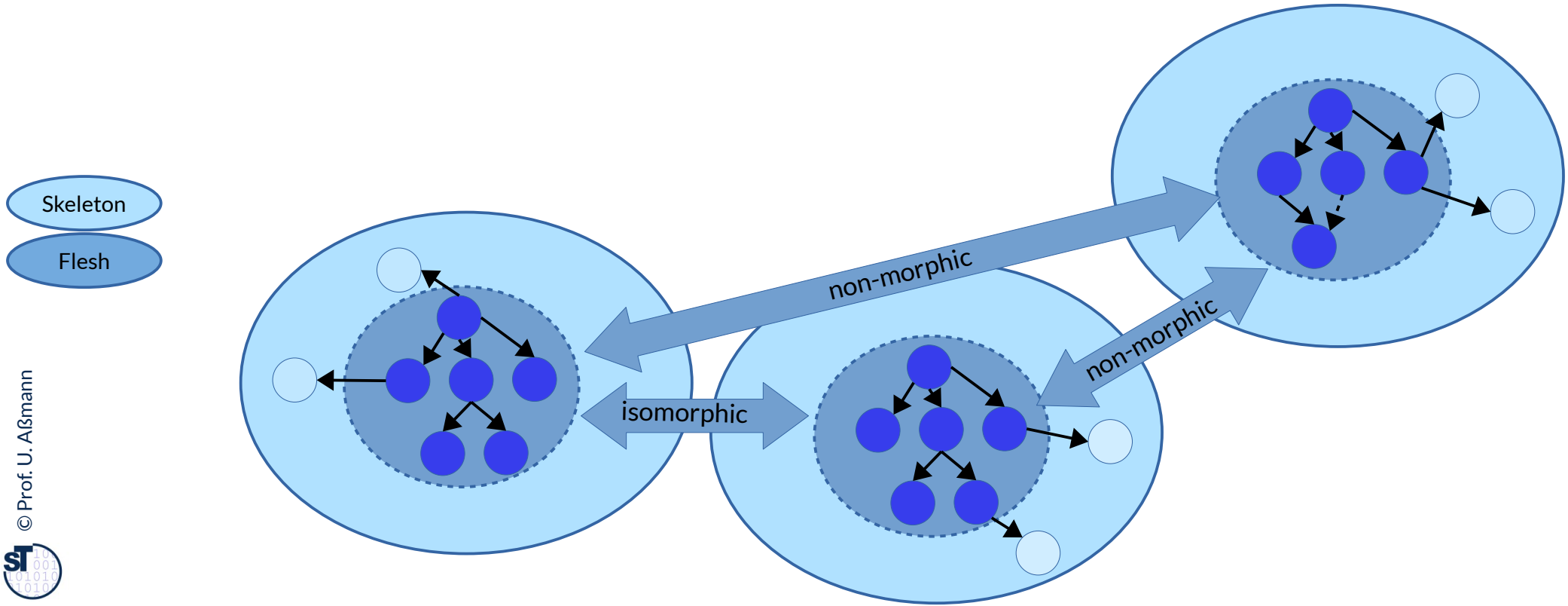

Orthograpic Software Modeling - q4/wc/PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi - Eclipse Platform
File Edit Navigate Search Project Scripts Topcased Launcher SmartQVT Run Window Help
Navigator
Dimension Explorer
Abstraction(PIM)
Version
latest
3 (Dez 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

28.2. The Skeleton-SUM

[Hettel08]
[Seifert11]

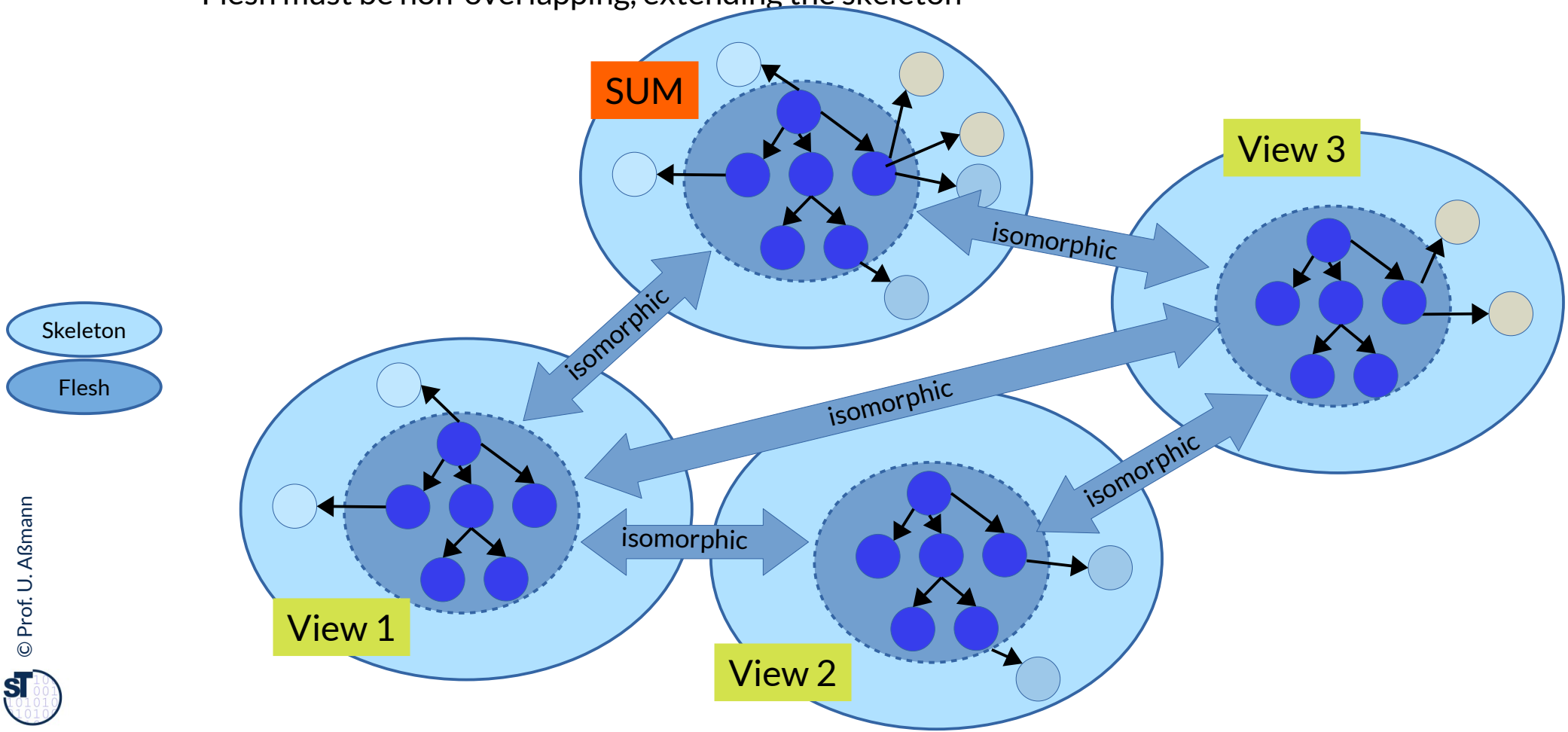
Skeletons and Flesh

- Skeleton splits models into
  - **Skeletons** (redundant) (several contexts)
  - and **flesh (clothing)** (locally different stuff in views, mono-context)
- Global invariants on skeletons vs. local „flesh“ variants
- Flesh must be non-overlapping, extending the skeleton
- Skeletons can have isomorphic, homomorphic, monotonically extended “skeleton” mappings,
  - or may be non-morphic



Mono-Skeleton-SUM

- Mono-Skeleton-SUM splits models into
  - **One common Skeleton** (redundant) (several contexts)
  - and **flesh (clothing)** (locally different stuff in views, mono-context)
- Global invariants on the ONE skeleton vs. local „flesh“ variants
- Flesh must be non-overlapping, extending the skeleton

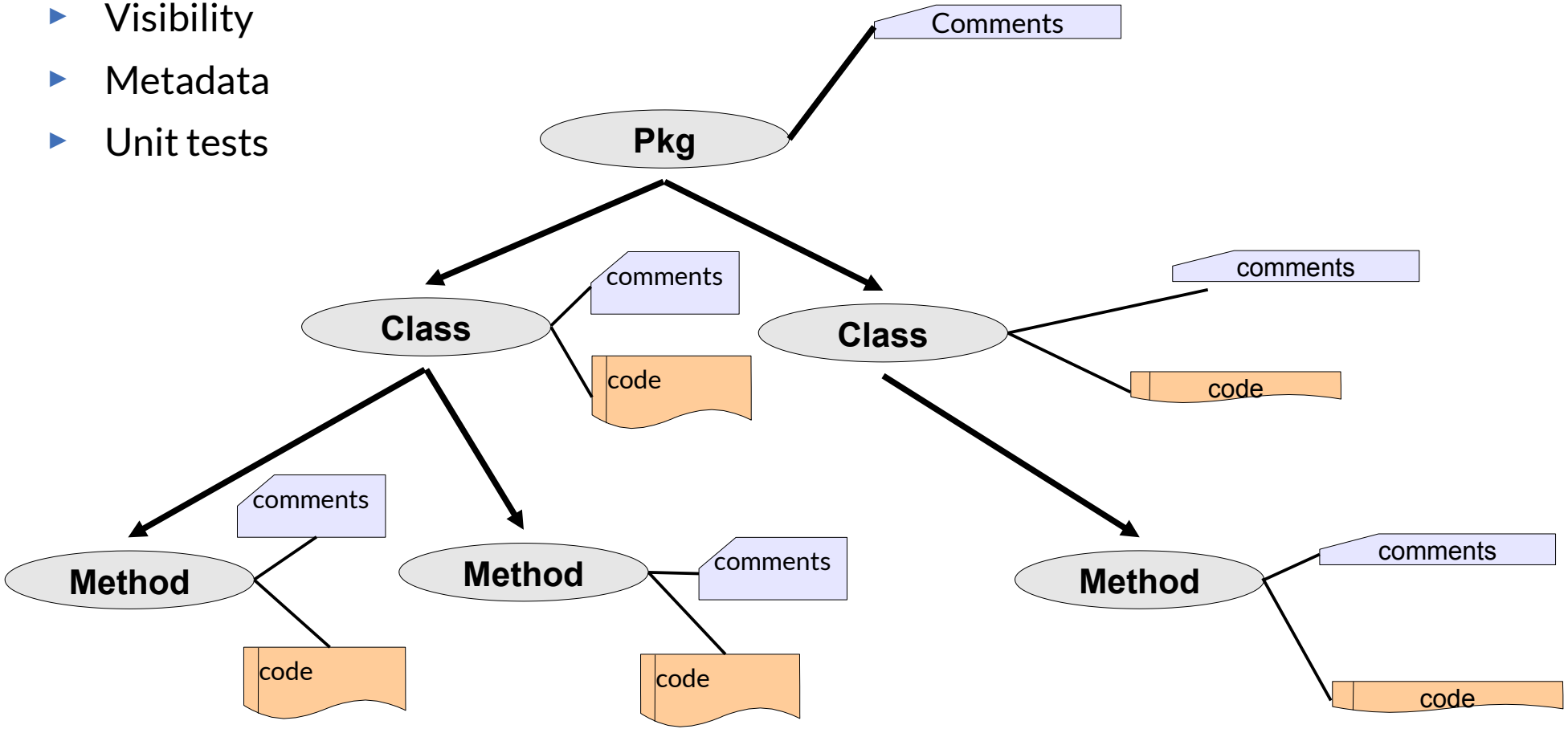


28.2.1 A Skeleton-SUM for Documentation

Example Skeleton-SUM: Scope tree of a program (static structuring)

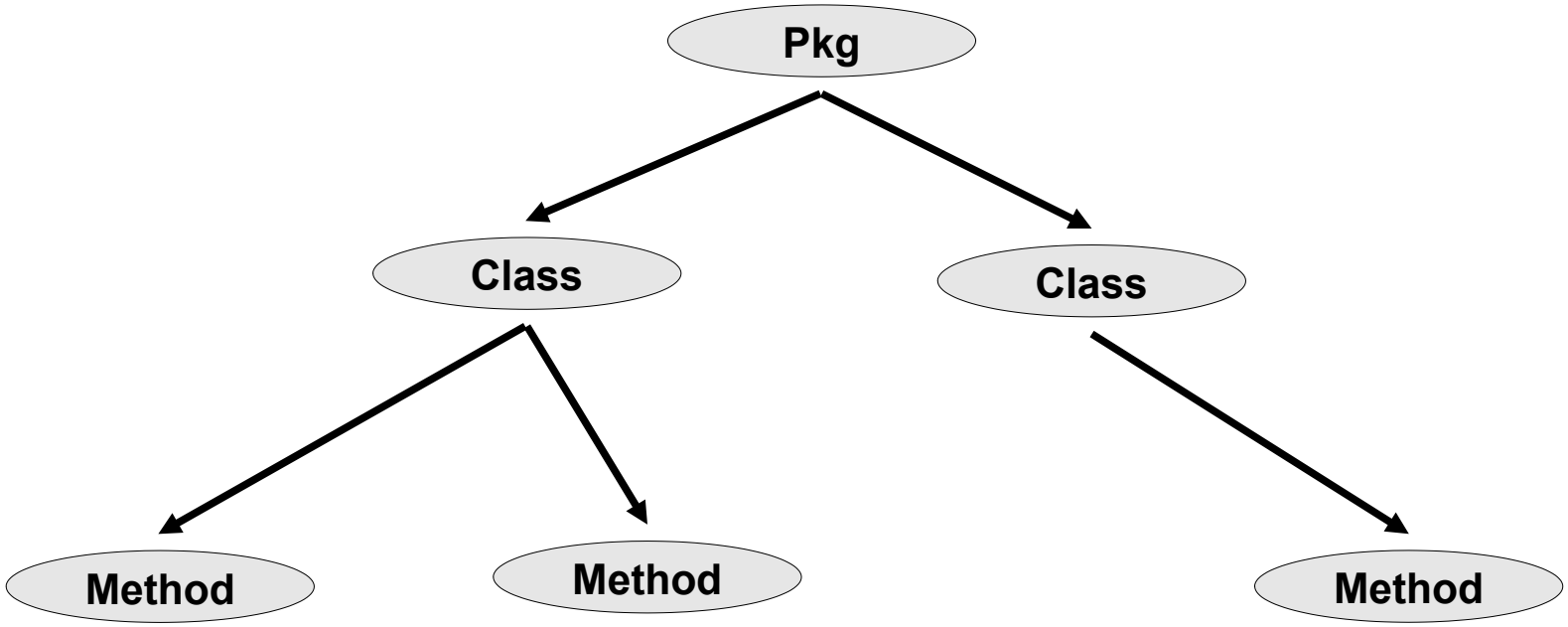
Attributes of Nodes:

- Comments (package, class, method, parameter)
- Code
- Visibility
- Metadata
- Unit tests



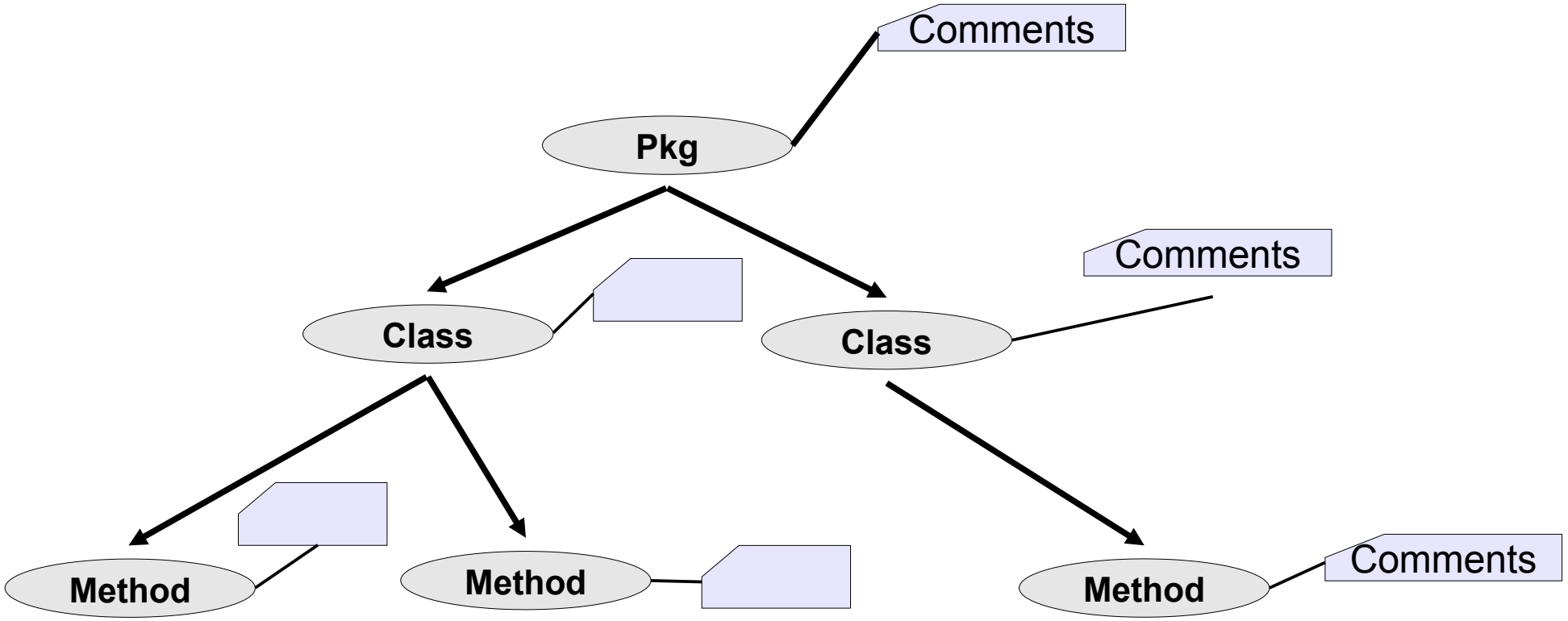
Projecting A Scope Tree for Skeleton

- put/get operations transform SU to views and back
- Get: partial function projection
- Put: partial function merge
- Ex: result of get operation for Scope Tree “Skeleton”



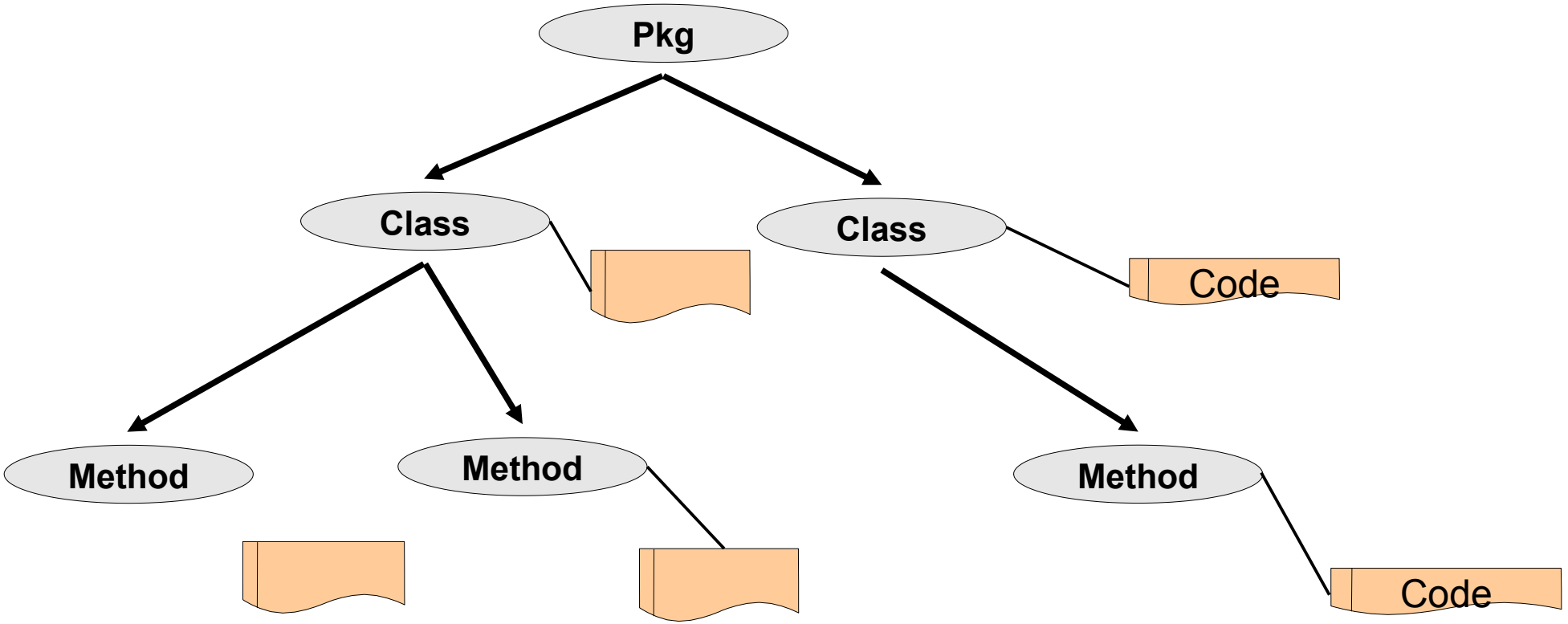
Projecting A Scope Tree for Skeleton

- Result of get operation for For Comment Context “Comment Flesh”



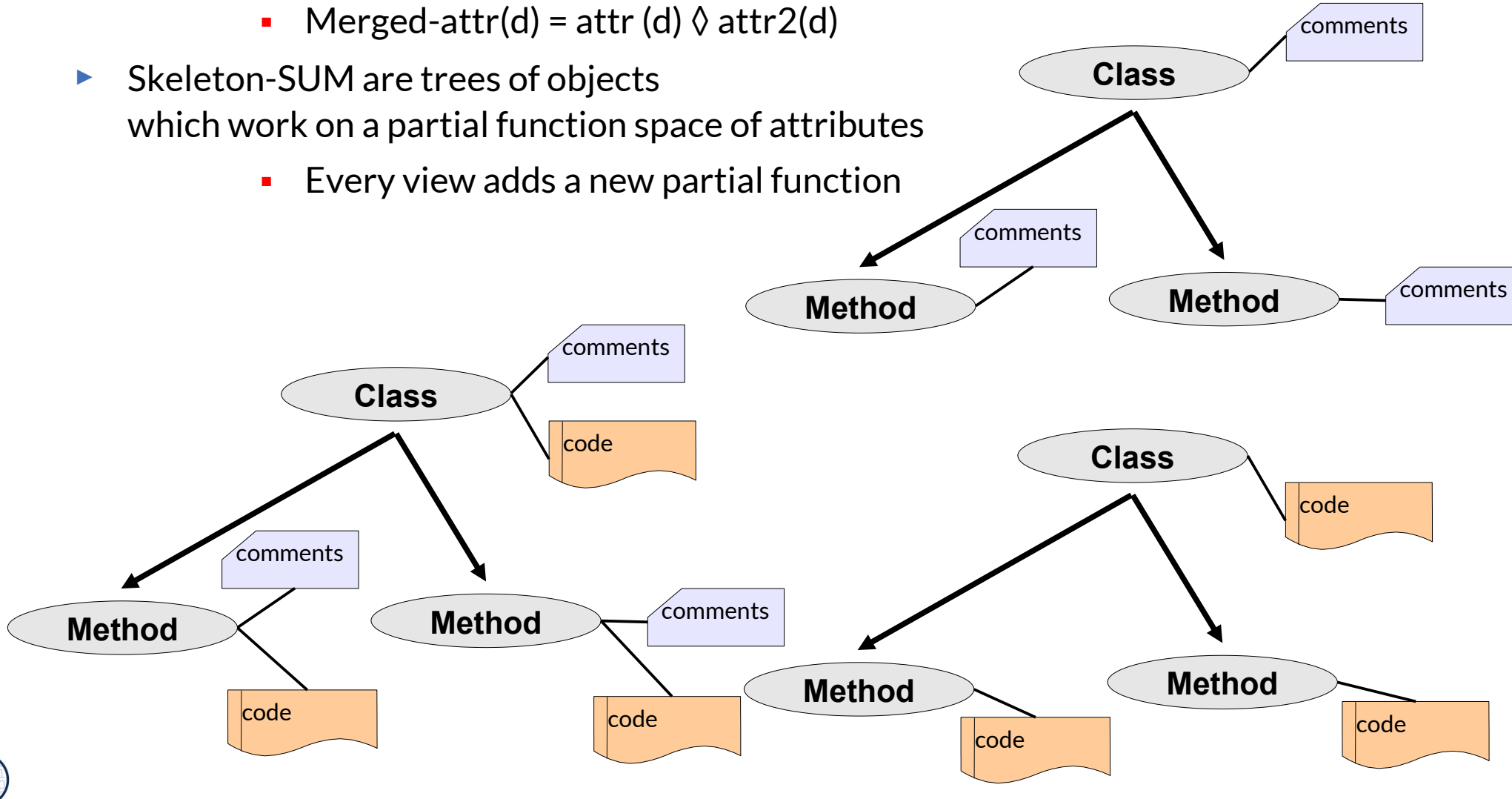
Projecting A Scope Tree for Skeleton

- Result of get operation for Code Context “Code Flesh”

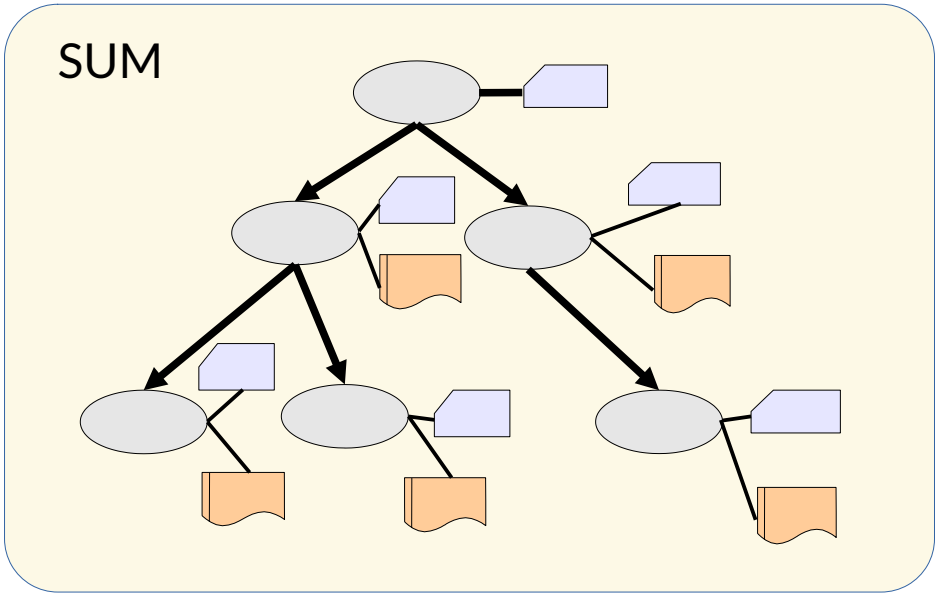


Merge of Partial Functions and Partial Trees in a Mono-Skeleton-SUM

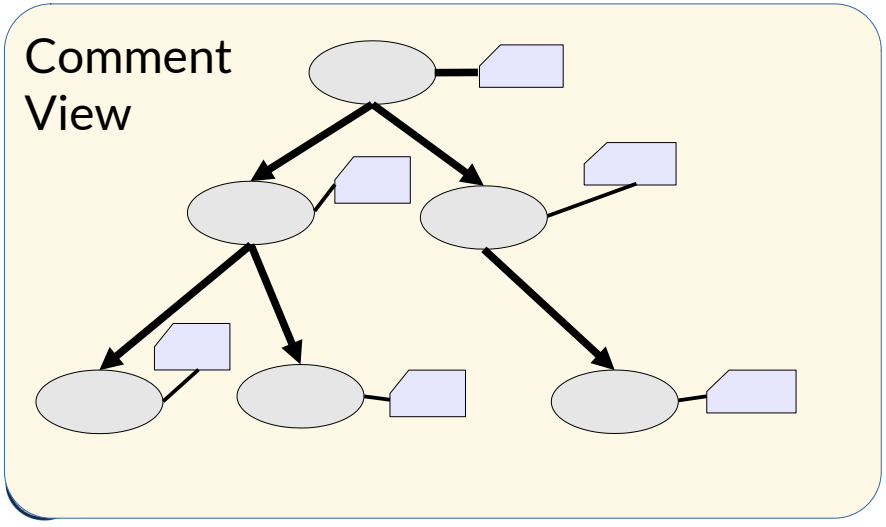
- Given two partial functions: attr: D → E and attr2: D → F
- Their merge merged-attr:D → E ◊ F
  - Merged-attr(d) = attr (d) ◊ attr2(d)
- Skeleton-SUM are trees of objects
  which work on a partial function space of attributes
  - Every view adds a new partial function

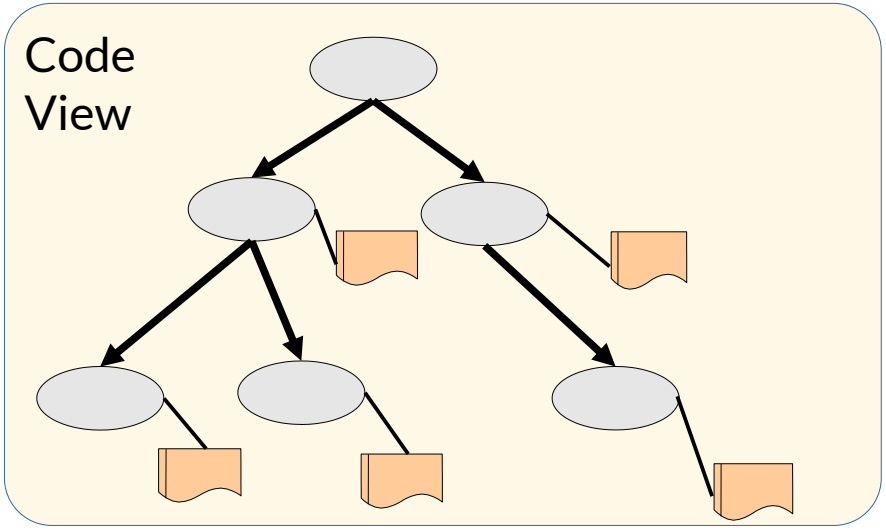


A Simple Metamodel-based Mono-Skeleton-SUM



CodeView and CommentView unify along the skeleton





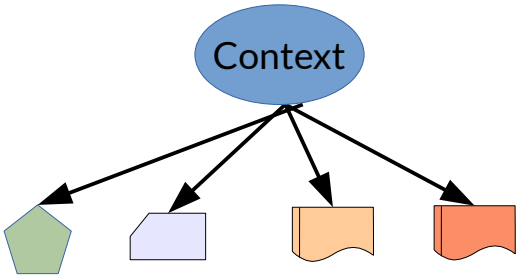
- The Skeleton need not be a link tree; it can be an arbitrary graph data structure
  - But RAGs can model Mono-Skeleton-SUMs very easily: inherit the flesh attributes to all nodes
- Between Skeleton and Flesh there holds a **key dependency**
  - A partial function describes the mapping between skeleton and flesh
  - Different partial functions exist for every view
  - Flesh-skeleton unification employs partial function merge (feature term unification)

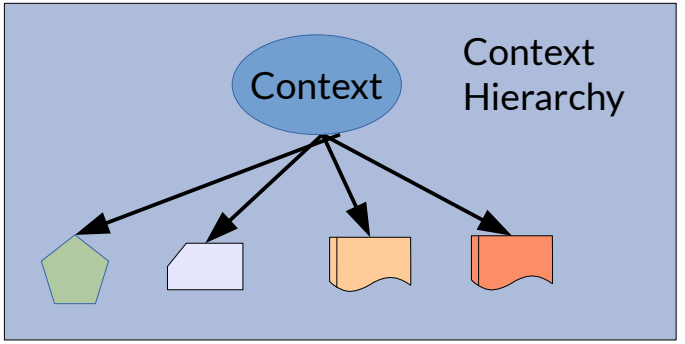
28.3. Context-Based Skeleton-SUM

[Hettel08]
[Seifert11]

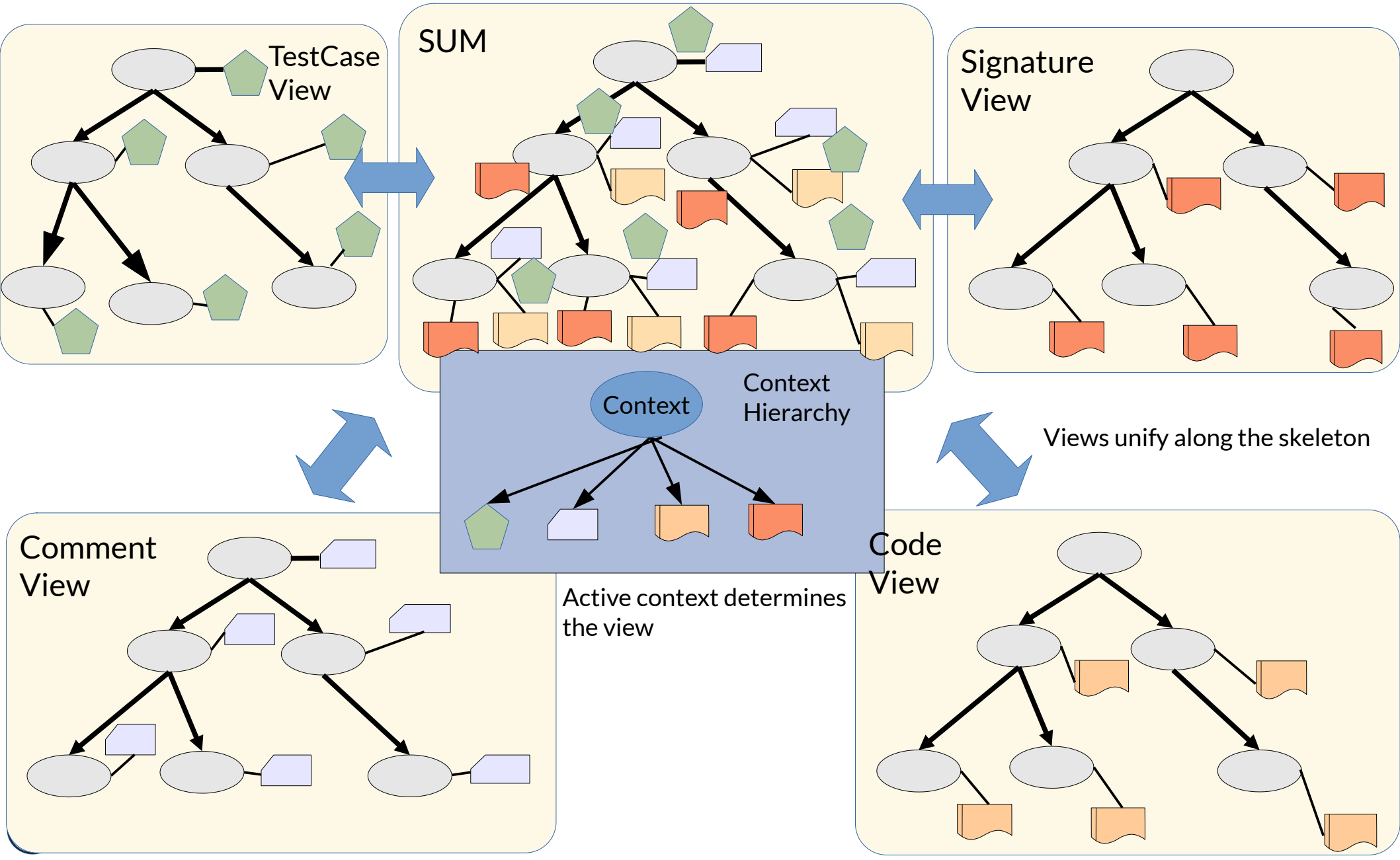
Skeleton-SUM

- Clothing can be associated to context (context-aware clothing)
  - Code context
  - Comment context
- If all clothings have mono-context, the SUM is called ***flat contextual SUM.***





A Metamodel-based Skeleton-SUM with Flat Context Hierarchy



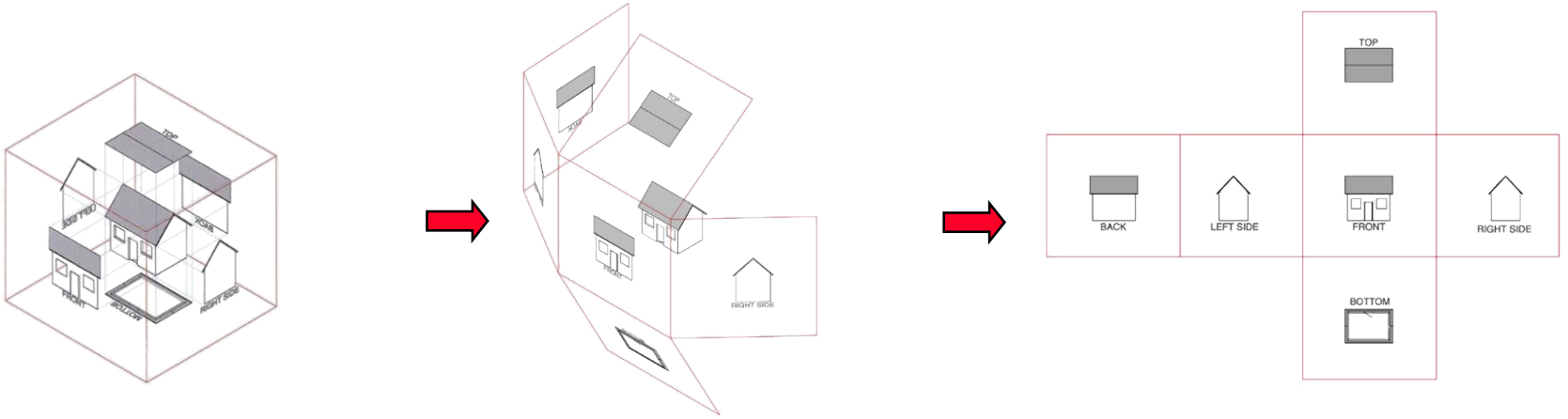
28.3.1. Orthographic Software Modeling (OSM) as a Dimensional, Context-Based Skeleton-SUM

[Hettel08]

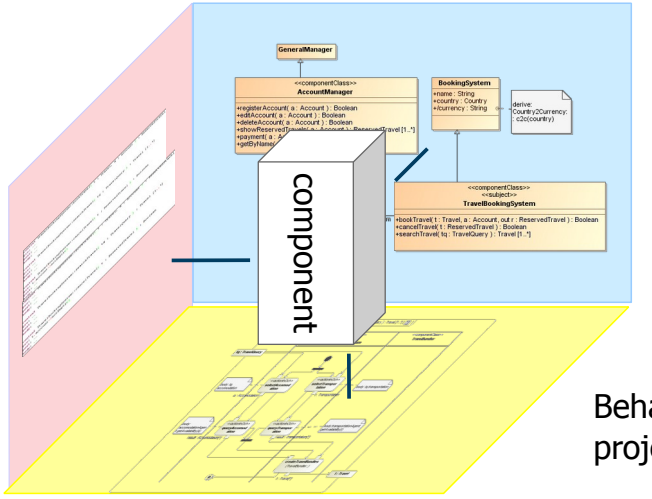
[Seifert11]

Orthographic Software Modeling (OSM) as a Dimensional Skeleton-SUM

- Many 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

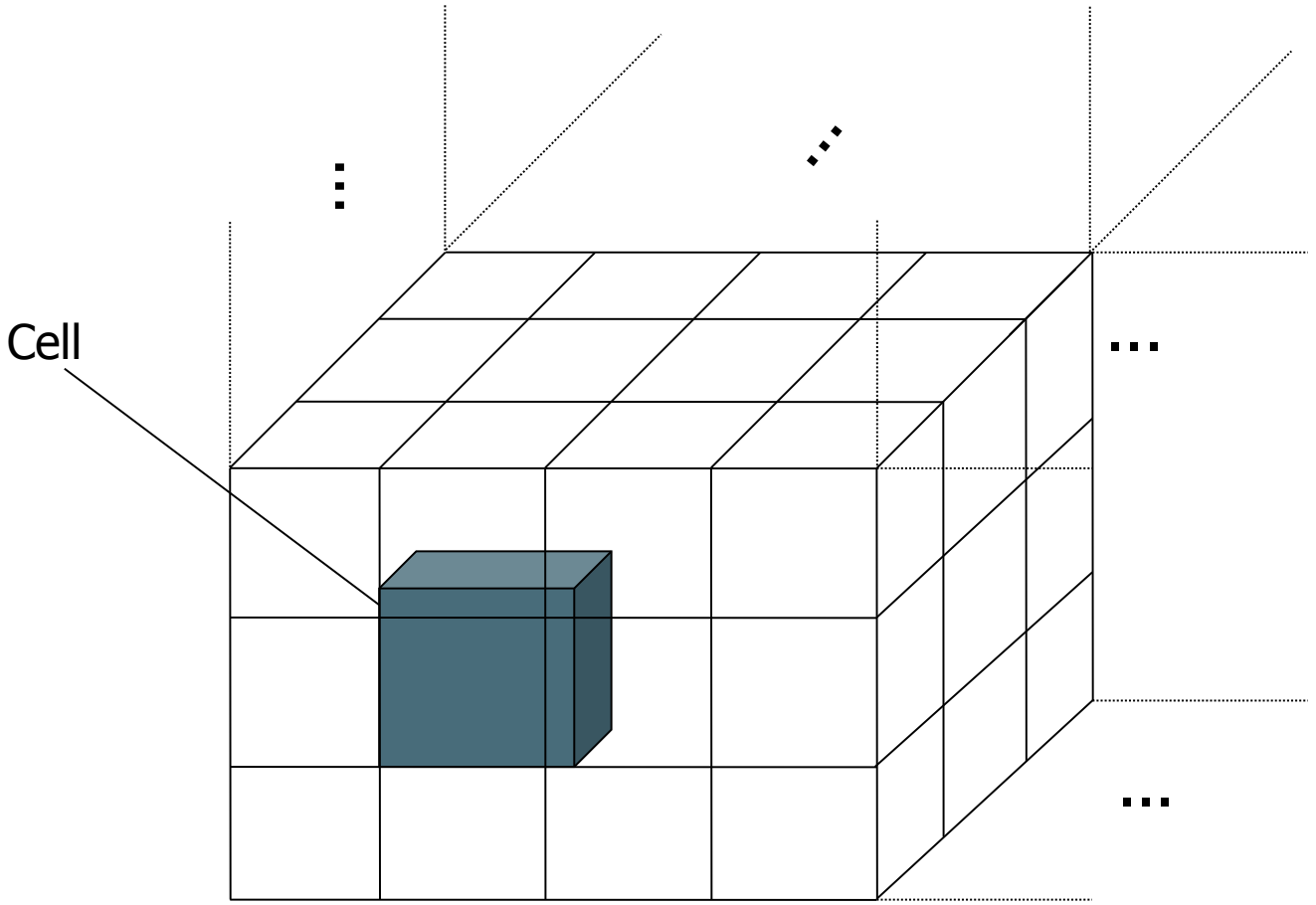
Operational projection

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



OSM is a Flat Contextual Skeleton-SUM

- OSM defines *n-dimensional contexts*, i.e., every model element is related to n contexts.
- OSM can be realized by a Skeleton-SUM providing n mono-contextual clothings
  - i.e., n mono-contextual attributes for every model element (link tree node).
- The n Contexts are used for projection
- Instead of attributes, model elements have roles (CROM-Skeleton-SUM)

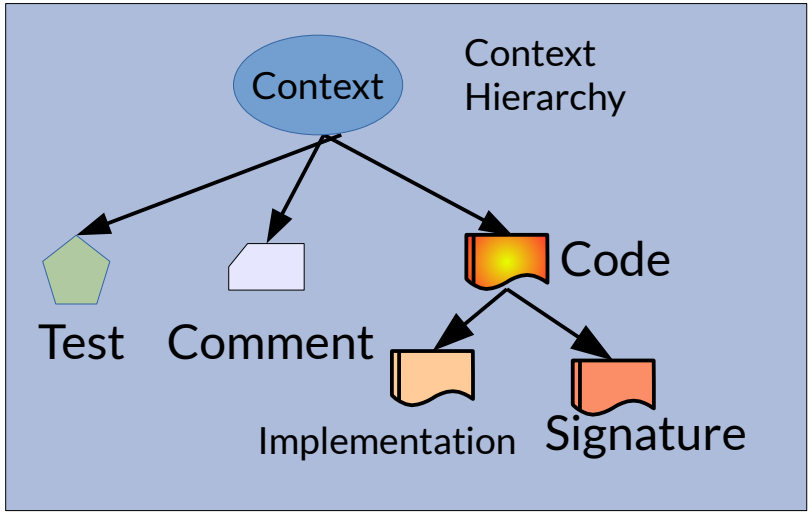
- ROSIMA is a CROM-Skeleton-SUM

28.4. Hierarchic Context-Based Skeleton-SUM

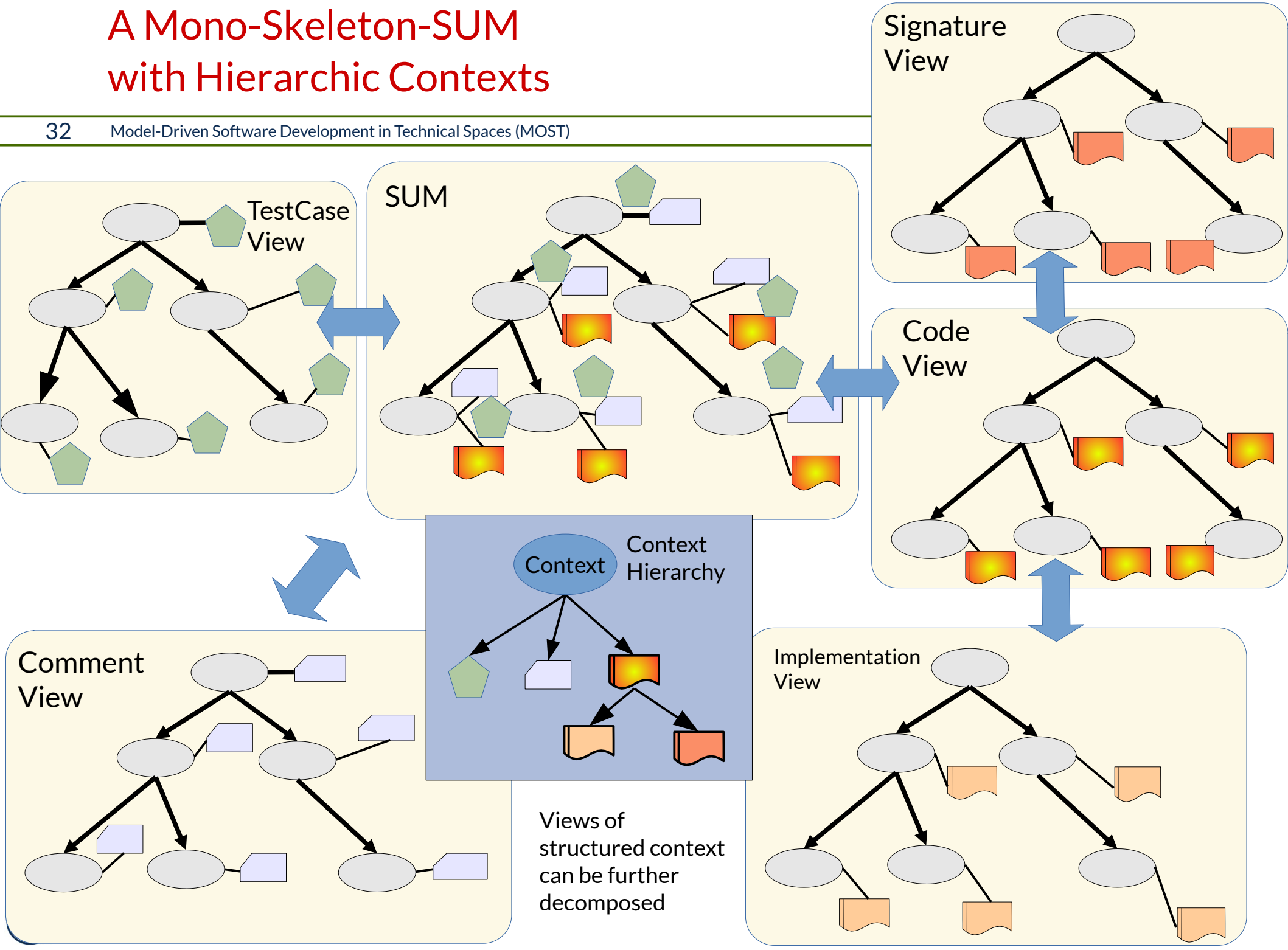
[Hettel08]
[Seifert11]

Hierarchic Skeleton-SUM

- Clothing can be associated to structured context
  - Code context
    - Signatures
    - Implementation
  - Comment context
- If som clothings have an inner (structured) context, the SUM is called **hierarchic contextual SUM.**



A Mono-Skeleton-SUM with Hierarchic Contexts



28.5 Delta-Based Lenses for Incremental Modifications for Scalability and Applicability of Skeleton-SUMs

[Diskin]

Delta-Based Lenses for Scalability and Applicability

- A technical approach must scalable for the chosen field of applicabilty
- Simple minded implementation approach –
  - uni-directional *exhaustive* 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
  - No incrementality
- Would work but -
  - not scalable (inefficient)
  - transformation more complex than necessary
  - too large grained

⇒ Delta-based bidirectional lenses

Delta-Based Lenses and Skeleton SUMs

- Lenses (Pierce et al. 2007) are bidirectional transformations based on get (exhaustive projection, decomposition, checkout) and put (exhaustive integration, checkin) operations on models
  - 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
```

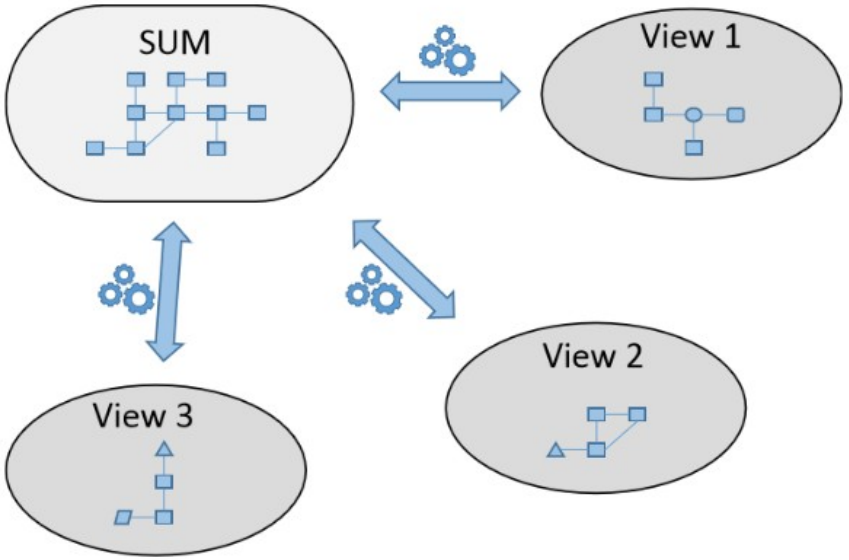
- Delta-based Lenses optimize the checkin/checkout (Diskin et al. 2011)
  - Incremental delta operations: dput and dget operations driven by the changes to the views

```
if Δs = dput(Δv, s), then dget(Δs) = Δv      // DeltaPUTPUT rule
```

  - much more fine-grained and scalable
  - Skeleton-SUMs fulfill the DeltaPUTPUT rule

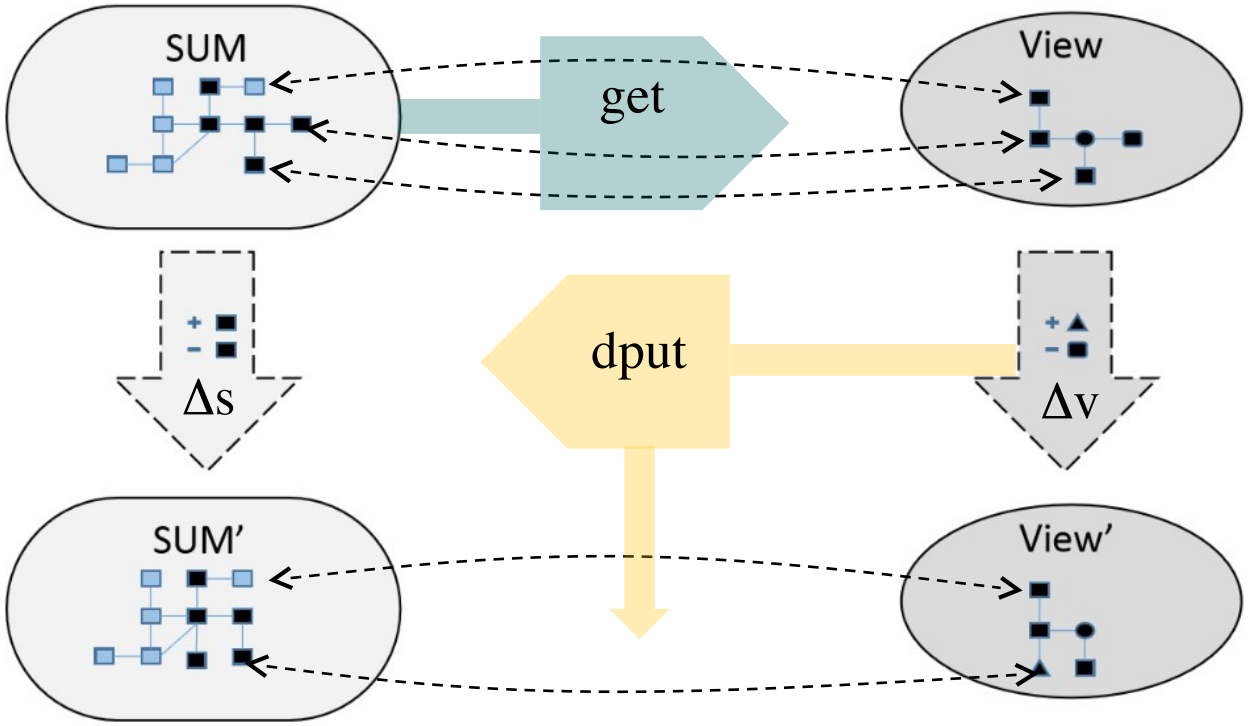
OSM Context

- In OSM, 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
- Skeleton-SUM (and therefore OSM9) fulfill the DeltaPutPut rule



if Δs = dput(Δv, s), then dget(Δs) = Δv // *DeltaPUTPUT rule*

Pros and Cons

- **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?

28.6 Skeleton-SUM on RoSI CROM

28.6 Skeleton-SUM on RoSI CROM

- The SUM principle can be played on all metalanguages, e.g., CROM
- CROM supports Mono-Skeleton-SUM for all
  - Contexts provide *viewpoints*
  - Cores provide *Skeleton*, Roles provide *flesh/clothing*
  - Role-play provides *partial functions from objects to roles* for a SkeletonSUM over cores and roles

Theorem: A CROM-based Skeleton-SUM fulfils the delta-putput invariant.

The End

- Explain, how partial functions between objects and attributes enable the projections (get) and the merge functions (put) of a Skeleton-SUM
- Why are contexts important for views?
- Which are the contexts of OSM?
- Why does ROSI-CROM enable Skeleton-SUM?