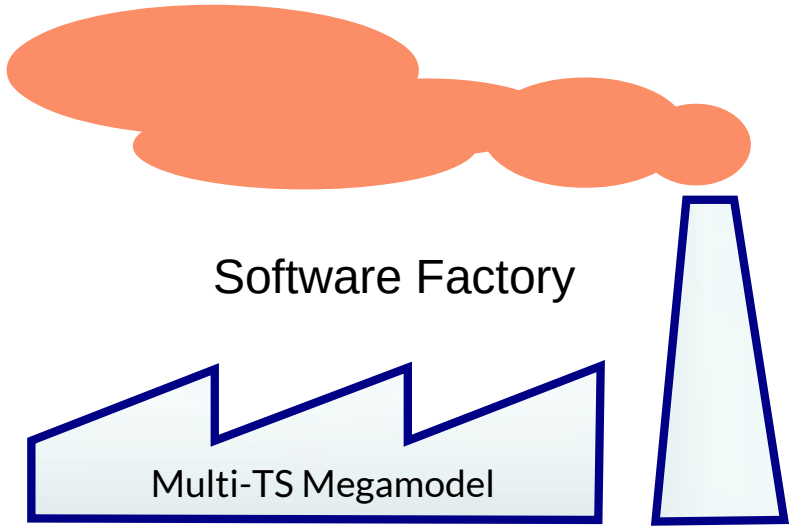
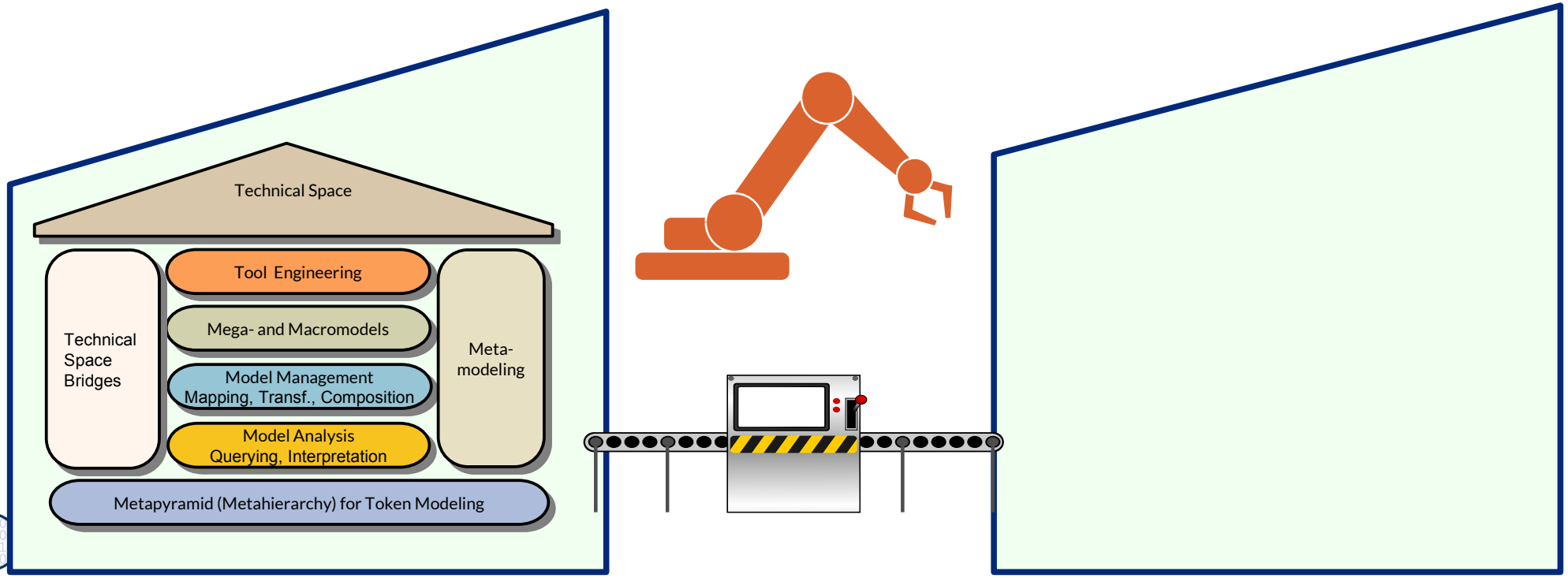
33. Macromodel Single Underlying Model (SUM)
with Orthographic Software Modeling (OSM) - A 1-TS-Megamodel with Total Consistency

Prof. Dr. U. Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
http://st.inf.tu-dresden.de/teaching/most
Version 21-1.1, 22.01.22

Software Factories with Only 1 Technical Space

In this chapter:
1-TS Megamodel
SUM





References

- [Atkinson19] Johannes Meier, Heiko Klare, Christian Tunjic, Colin Atkinson, Erik Burger, Ralf Reussner, and Andreas Winter. Single underlying models for projectional, multi-view environments. In Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD, pages 119--130. INSTICC, SciTePress, 2019.
- 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 Link-Tree Macromodels

The Link-Treeware TS is well apt for macromodel construction in a software factory

- A tree node abstracts a subtree (representant)
  - Attributes and attributions are *composable partial mappings* from treenodes
- RAGs are useful for all kinds of structure- and function-modeling in Link-Tree 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
- Relational RAGs (RelRAGs) are useful, because they have bidirectional constraints

	(Plain) MDA	General SUM	Skeleton SUM (partial function extension)
RAGs in Repositories	Markings		Repository-SUM: get/put as higher-order attributions of link trees
			• Javadoc-SUM
RAGs in Data-flow architectures	Needs trace models	get/put as model transformations (lenses)	Flow-SUM: Communicating link trees; In-place transformations of SUM
			• Google Docs, Stream-Based MDA

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

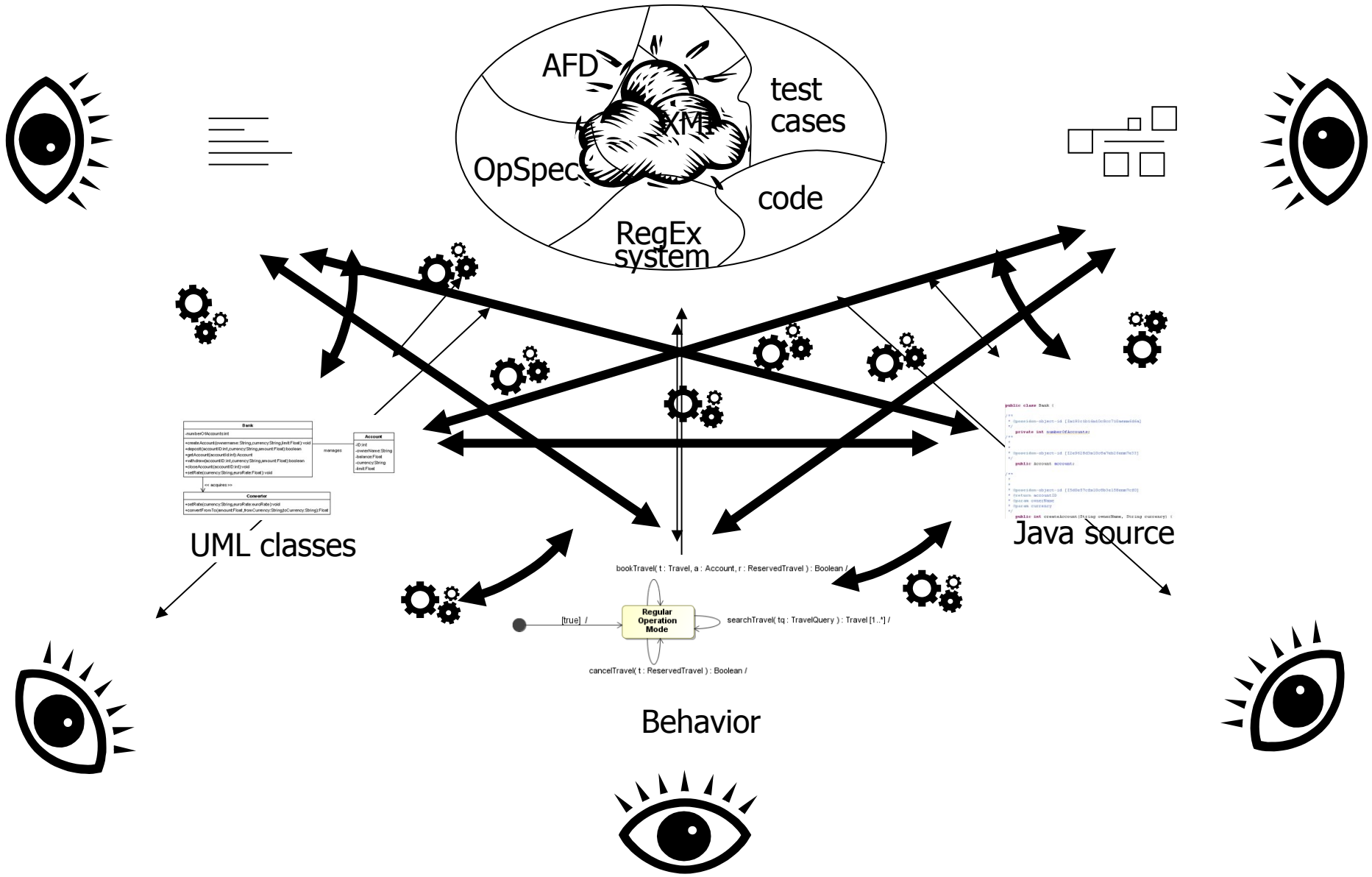
L'Aquila. Italy
21 July, 2015

UNIVERSITÄT MANNHEIM

33.1. The Macromodel “Single-Underlying Model (SUM)”

- is based on a ***repository (repository-based SUM)*** [Atkinson19]

Traditional View-based Software Engineering (VOSE)



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

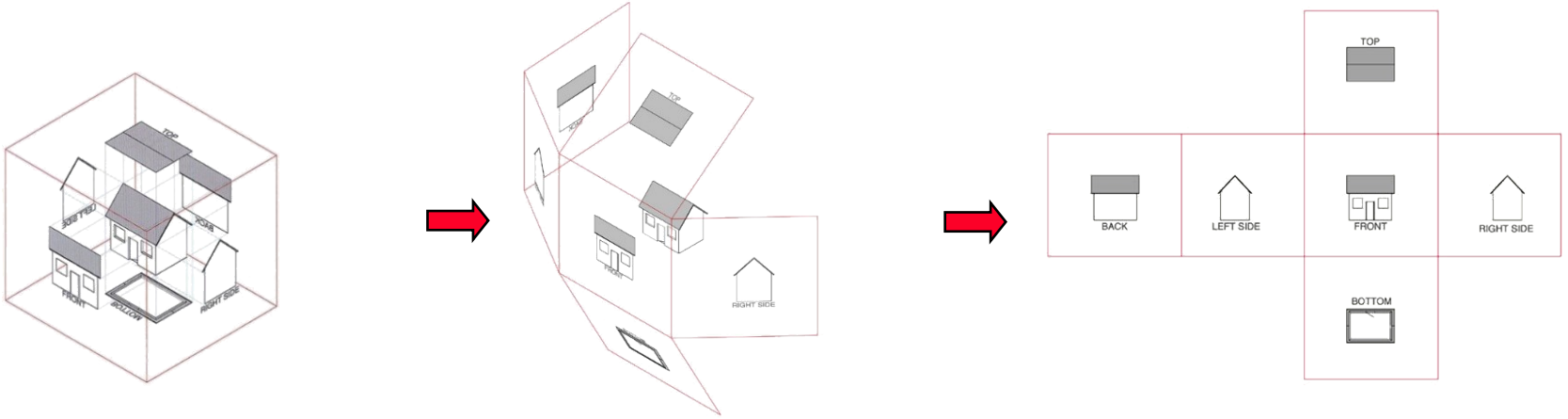
Context 2

Context 3

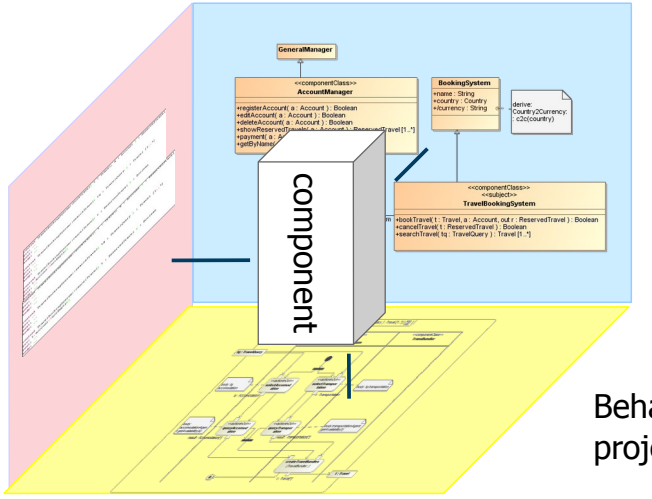
Context 4

Orthographic Software Modeling (OSM) as a 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?

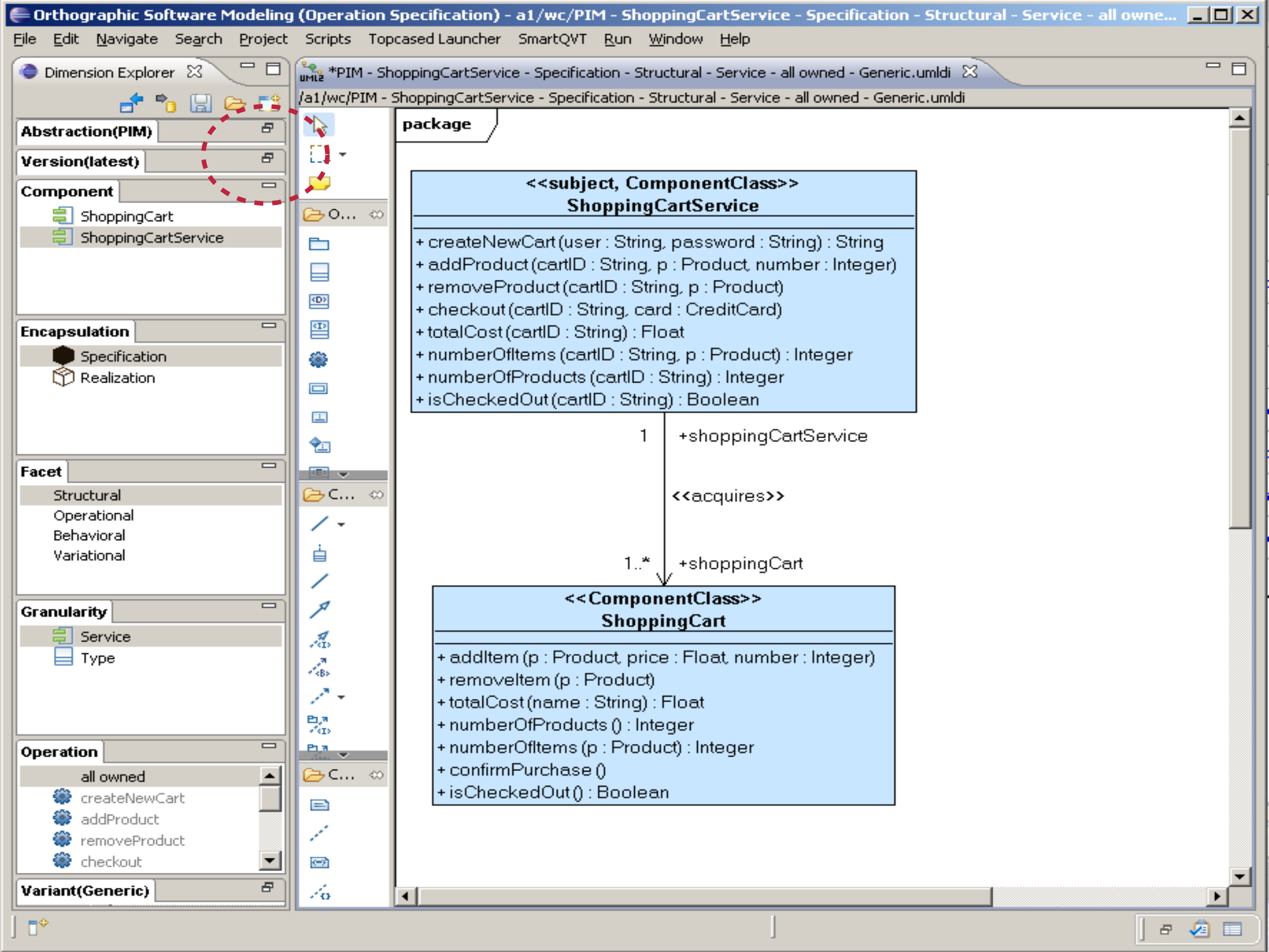


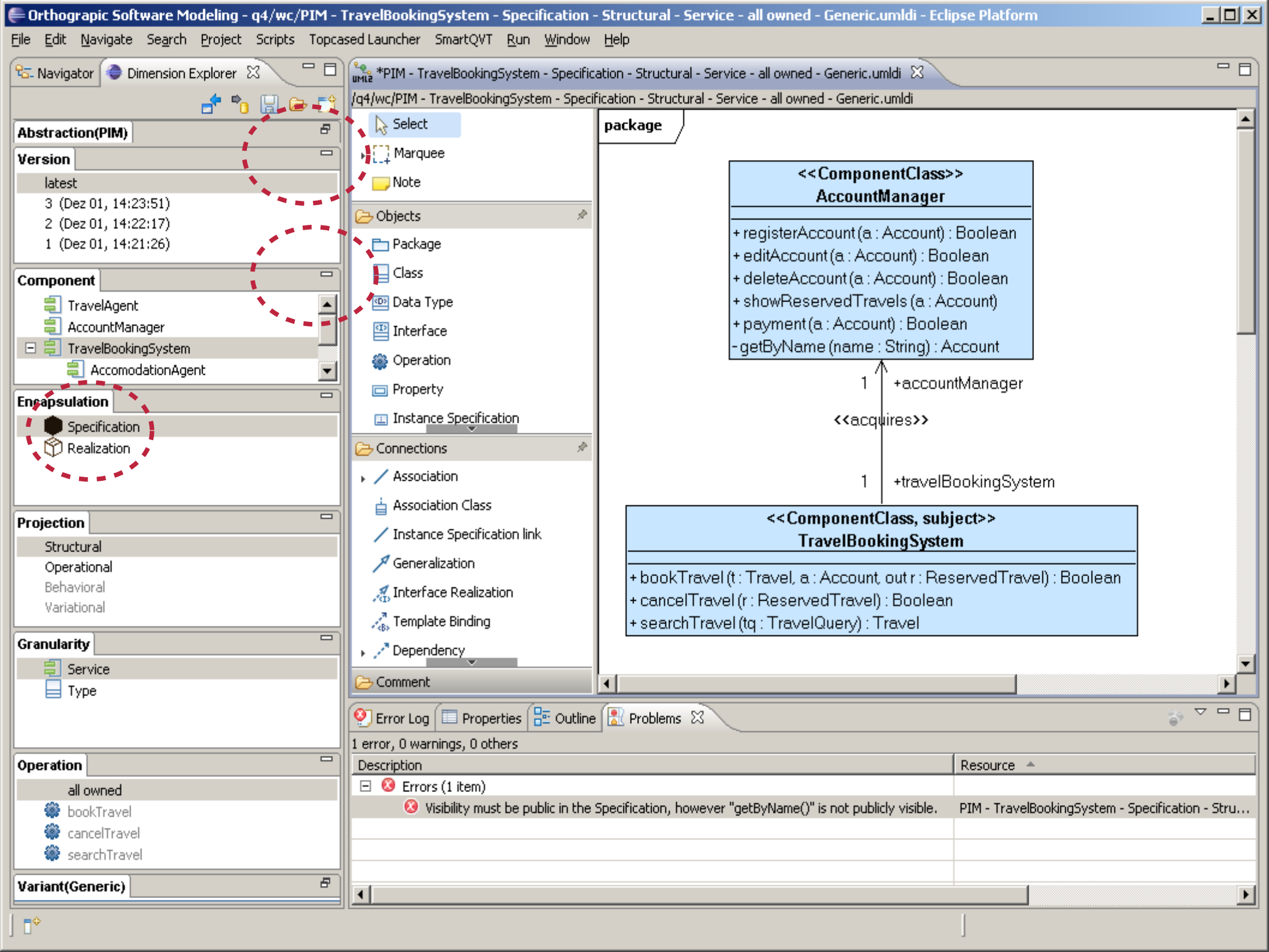
Structural projection

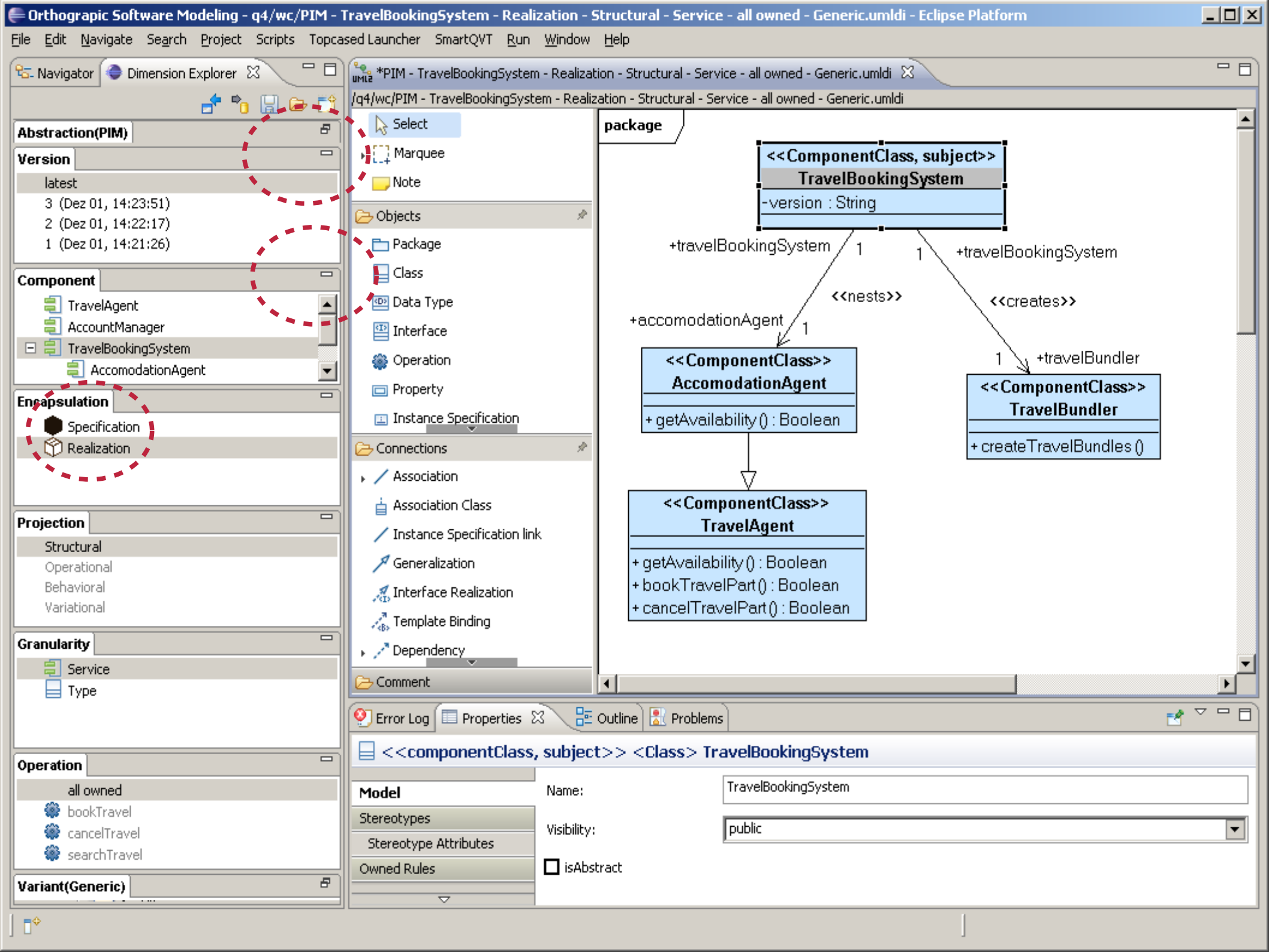
Operational projection

Behavioral projection

- On demand view generation (projective views)
- Dimension-based navigation
- View-based methodology
- Arrangement in a multidimentional SUM


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...

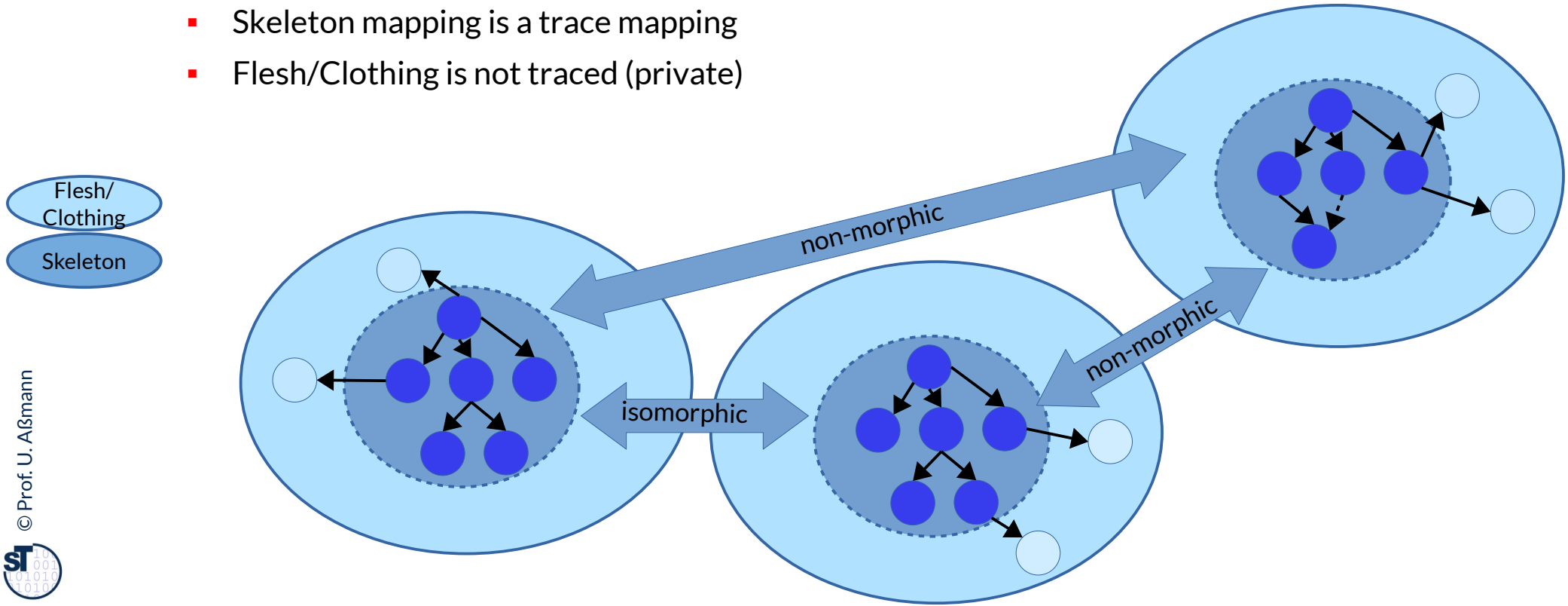

Orthograpic Software Modeling - q4/wc/PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi - Eclipse Platform
File Edit Navigate Search Project Scripts Topcased Launcher SmartQVT Run Window Help
Navigator
Dimension Explorer
Abstraction(PIM)
Version
latest
3 (Dez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

33.2. The Skeleton-SUM

[Hettel08]
[Seifert11]

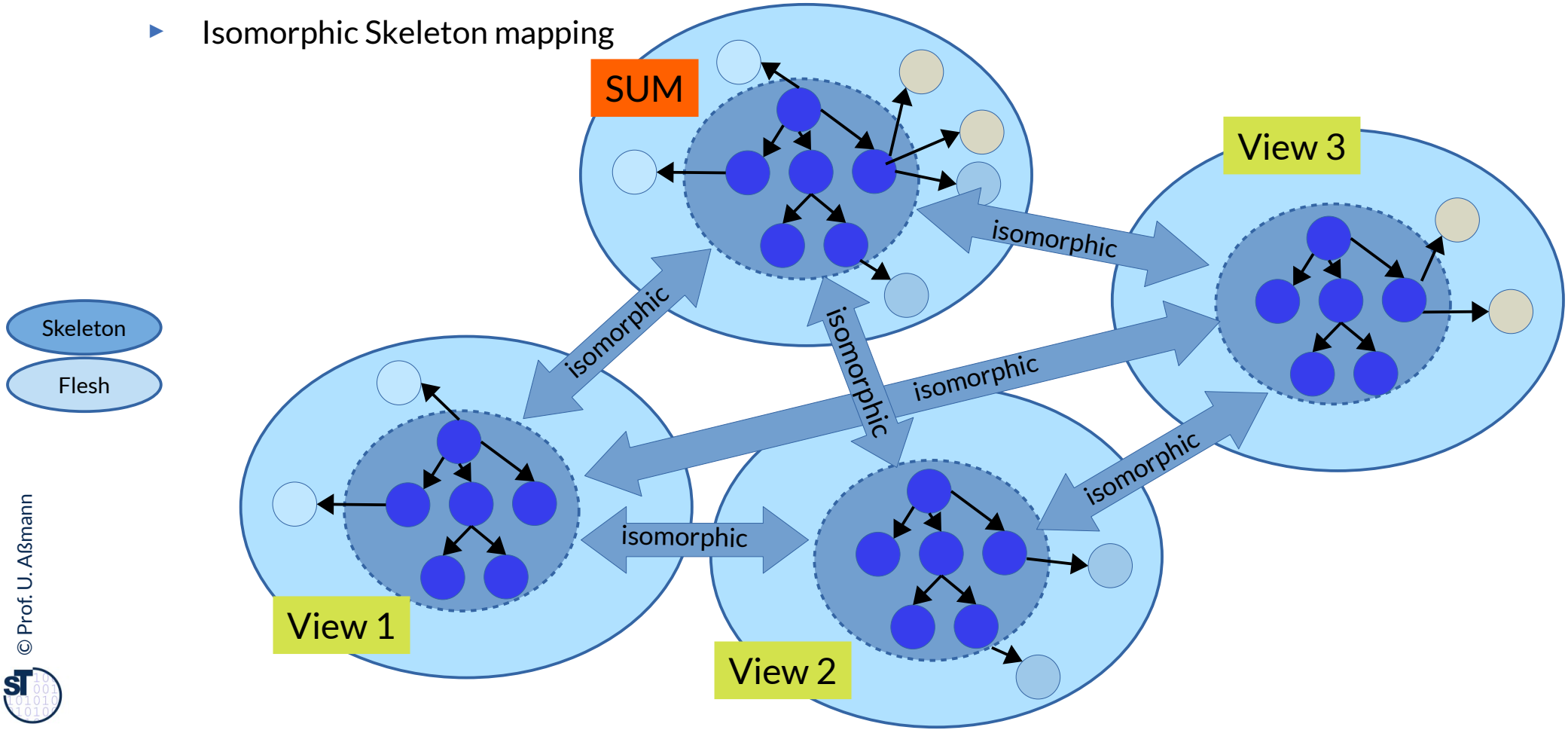
Skeletons and Flesh

- 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
  - Skeleton mapping is a trace mapping
  - Flesh/Clothing is not traced (private)



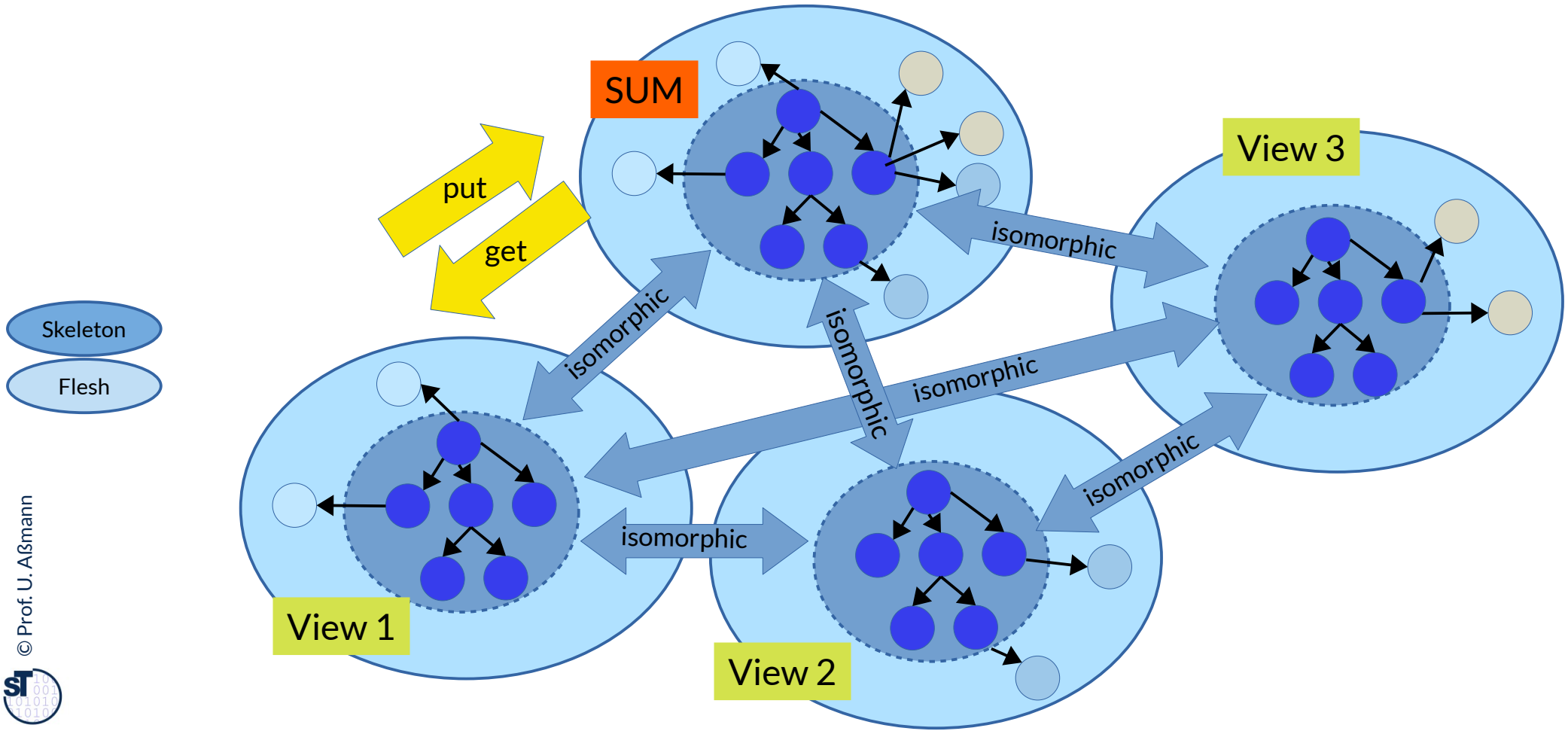
Mono-Skeleton-SUM

- Mono-Skeleton-SUM splits models into
  - **One common Skeleton** (redundant) (several contexts)
  - and **flesh (clothing)** (locally different stuff in views, mono-context) is stored in SUM together with skeleton
- Flesh must be non-overlapping, extending the skeleton
- Isomorphic Skeleton mapping



Get/Put in Mono-Skeleton-SUM

- From a Skeleton-SUM
  - get operation produces a view
  - put operation commits it into SUM

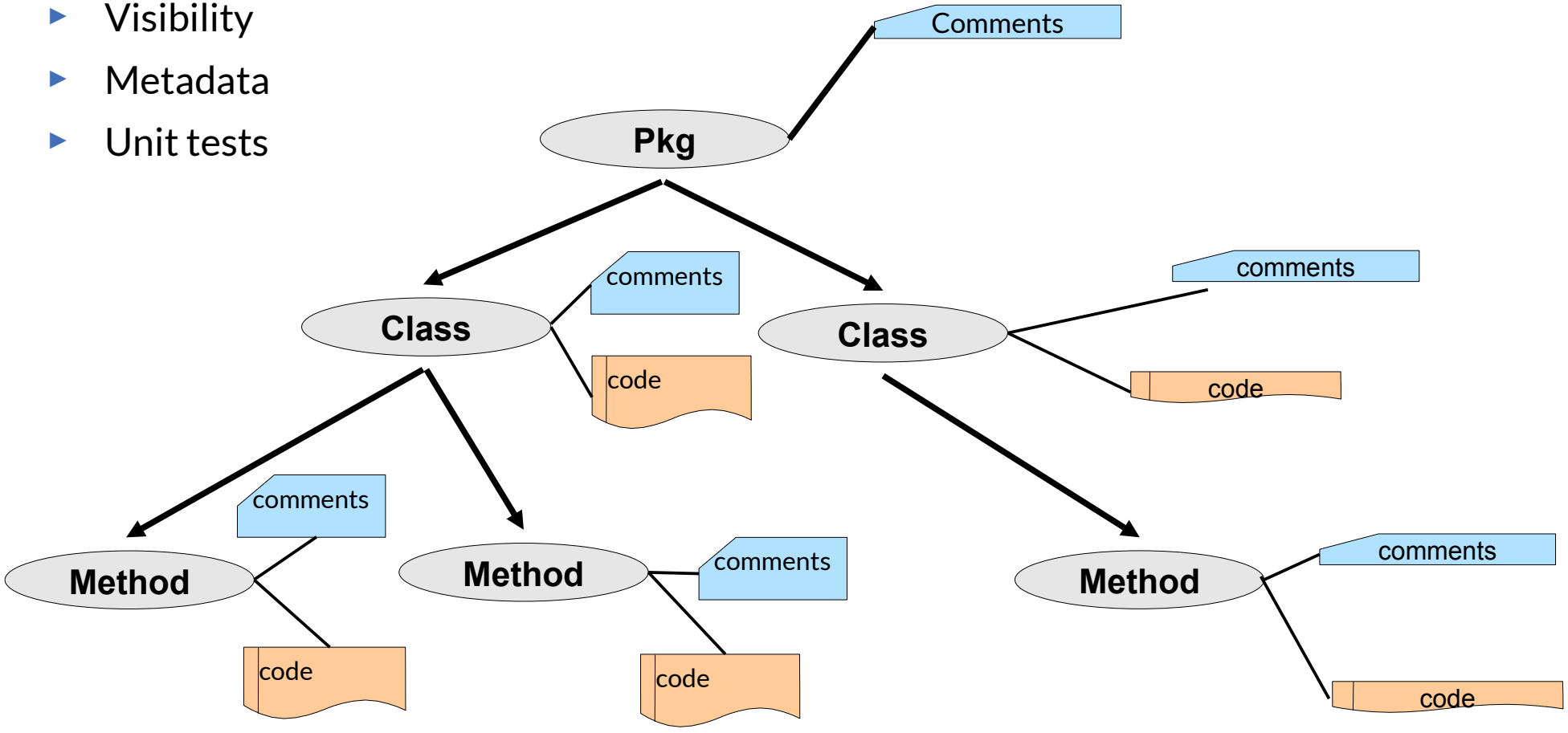


33.2.1 Javadoc-SUM, a Mono-Skeleton-SUM for Documentation

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

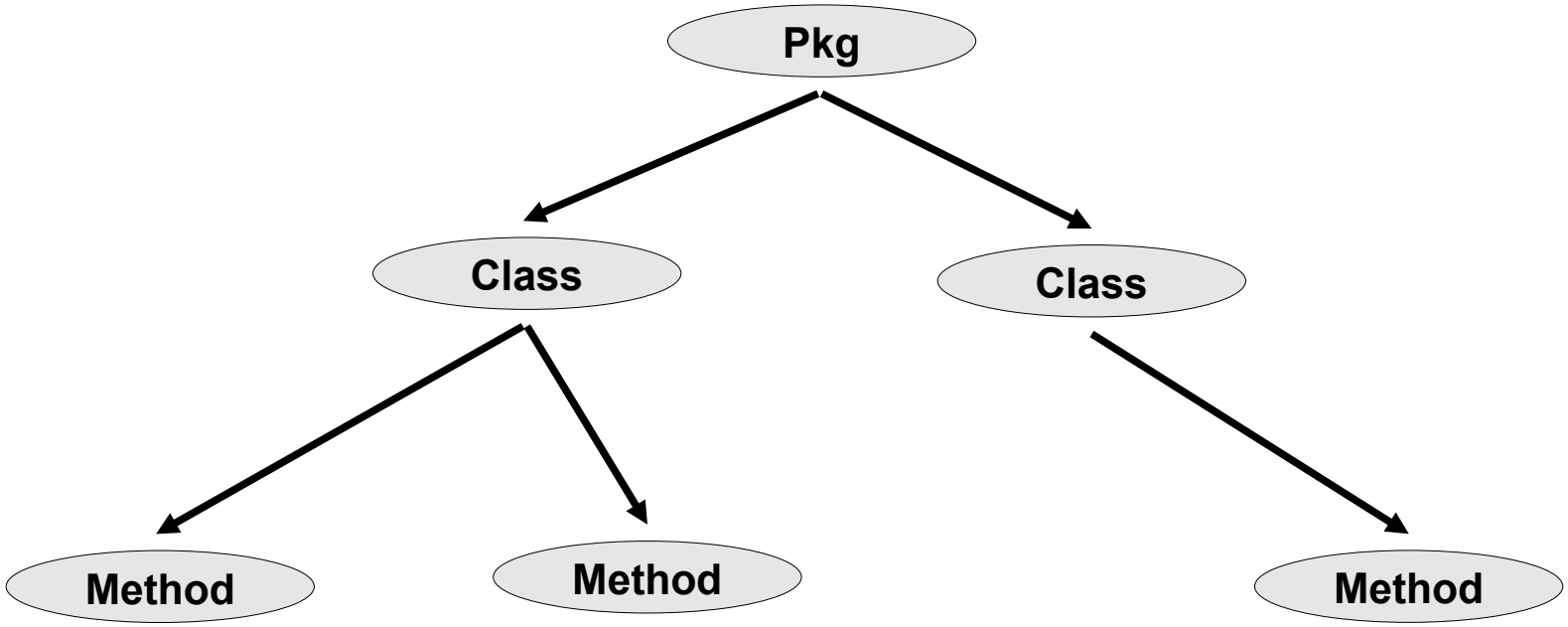
Javadoc comment relies on several attributes of nodes of the syntax tree:

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



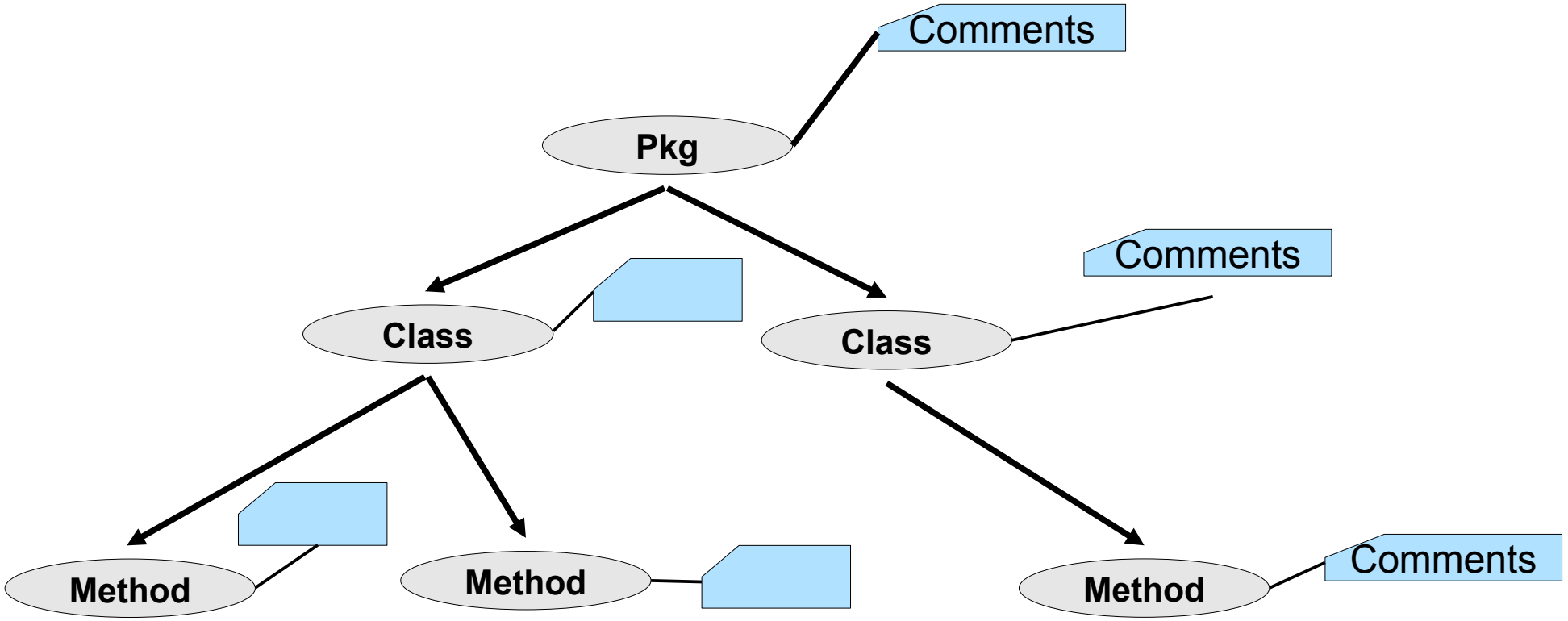
Projecting A Scope Tree for Skeleton

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



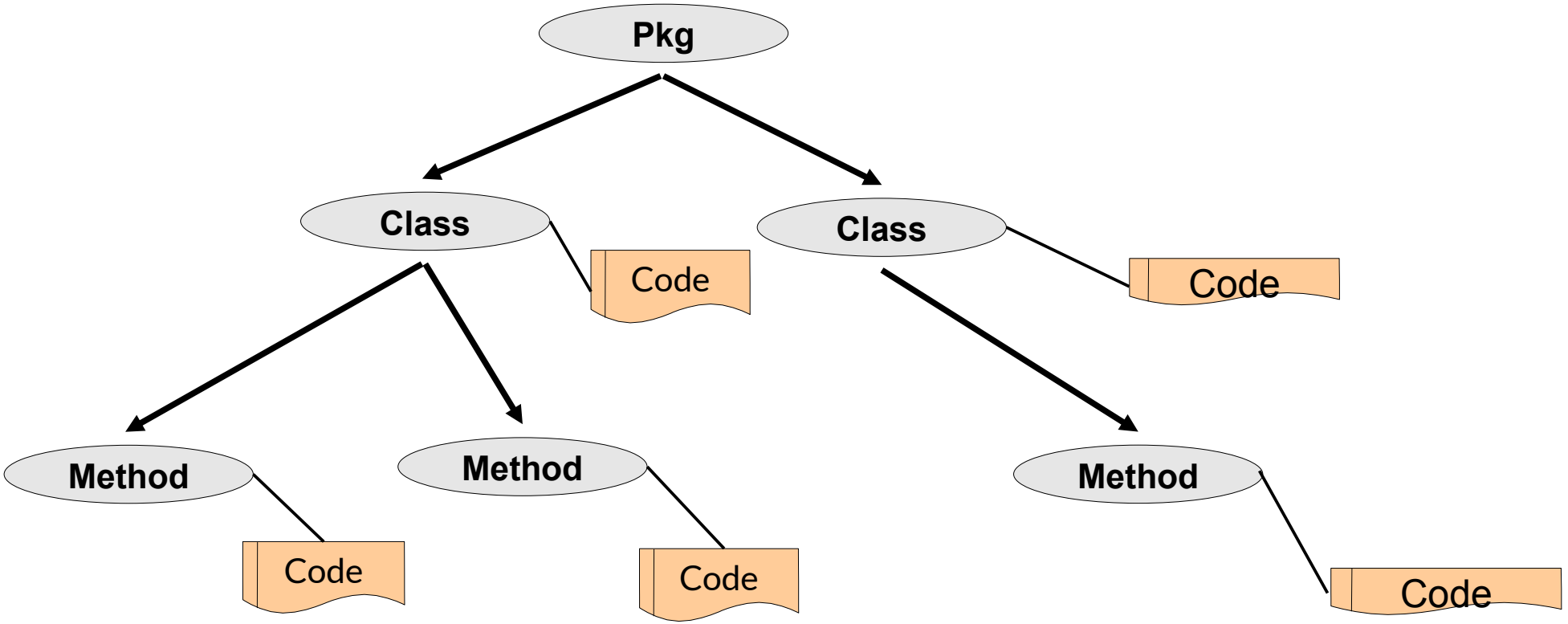
Projecting A Scope Tree for Skeleton

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



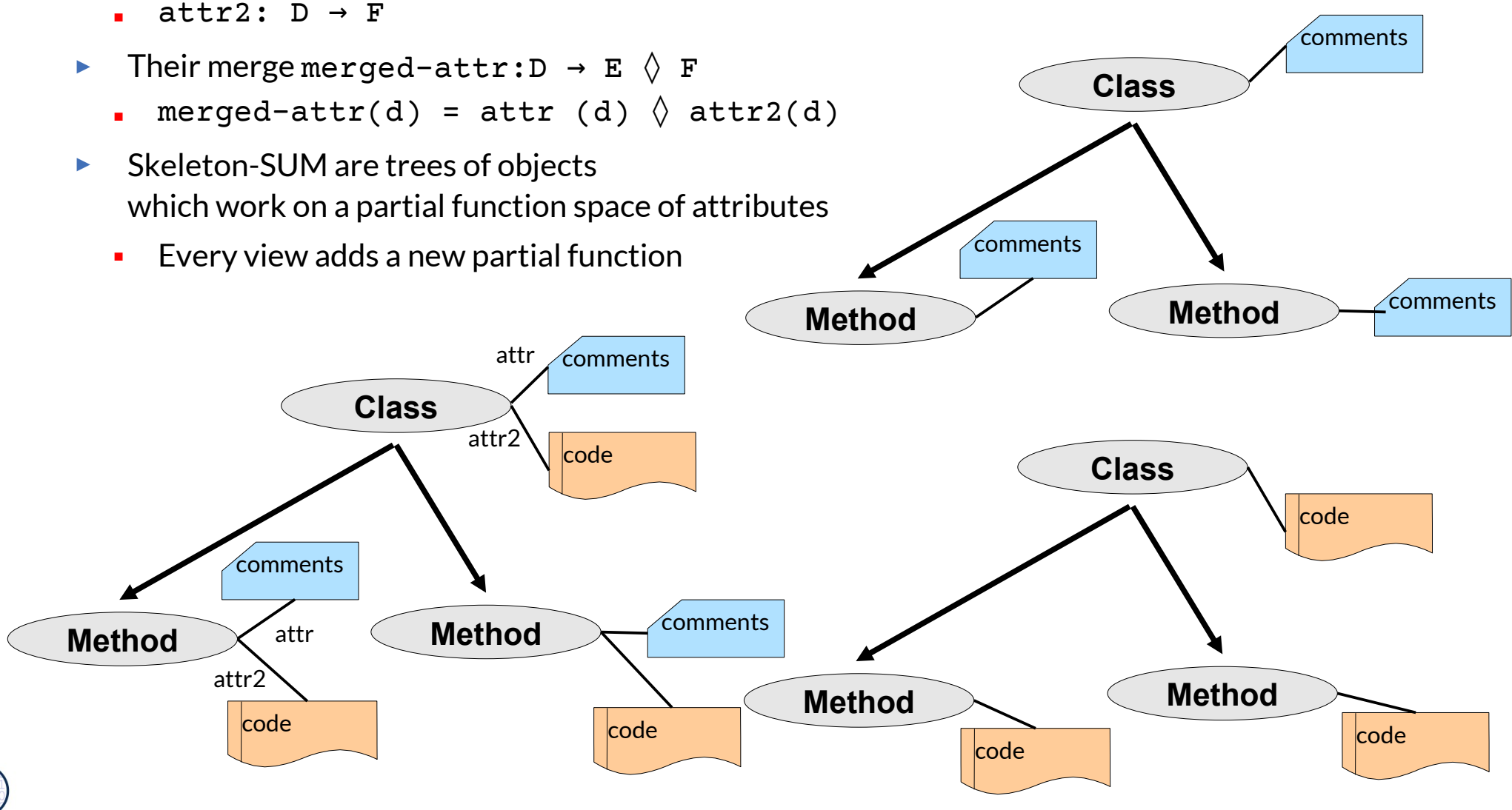
Projecting A Scope Tree for Skeleton

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

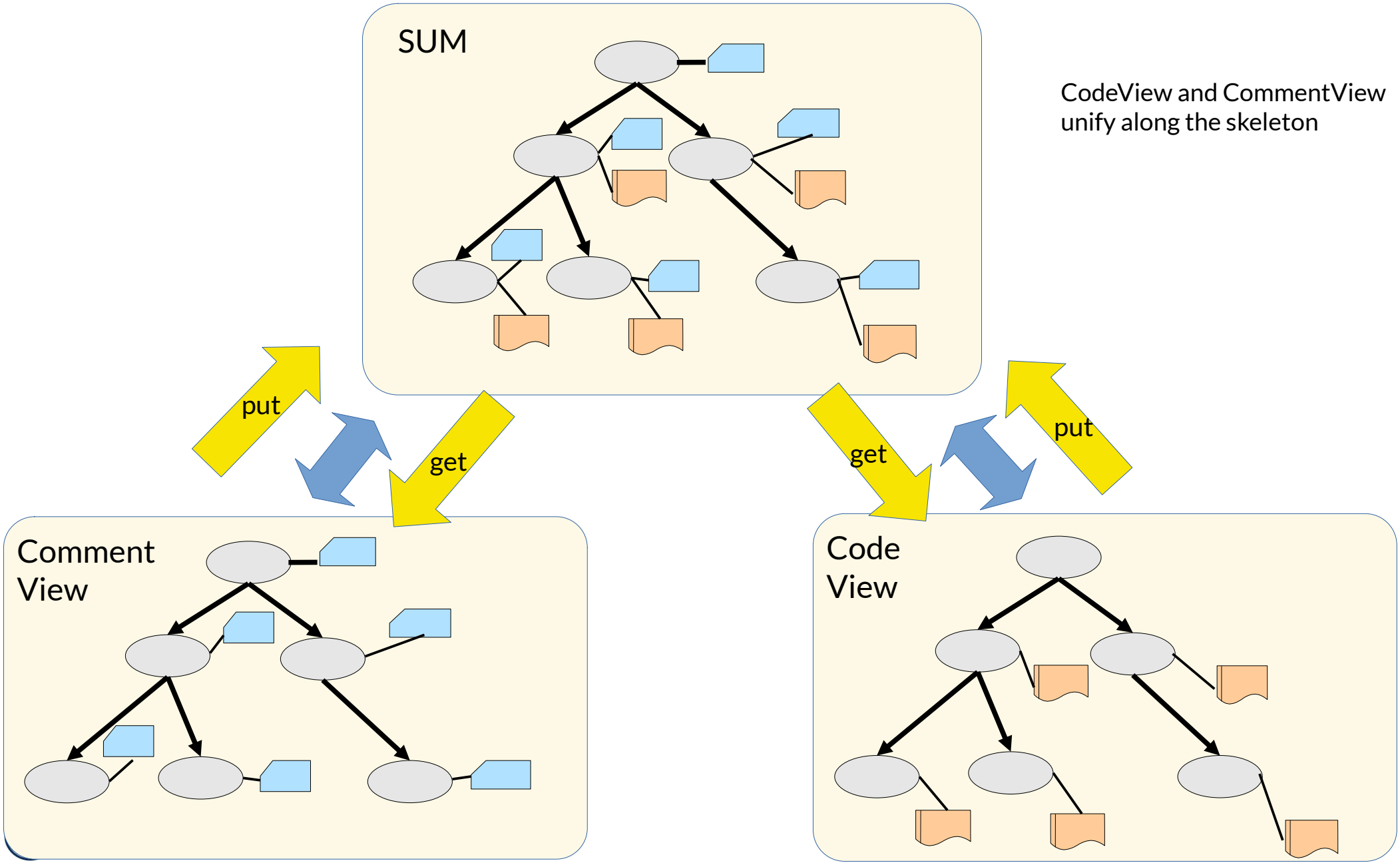


Merge of Partial Functions and Partial Trees in a Mono-Skeleton-SUM

- Given two partial functions on tree-node domain D and two domains E, F:
  - `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



Javadoc-SUM:
A Simple Metamodel-based Mono-Skeleton-SUM



Remarks on Mono-Skeleton-SUM

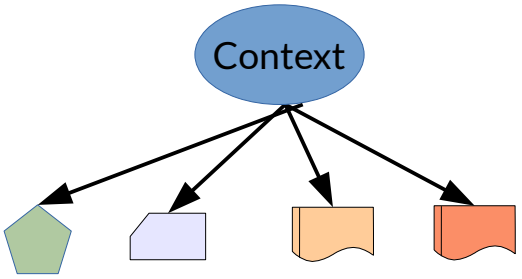
- **Generality:** The Skeleton need not be a link tree; it can be an arbitrary graph data structure
  - But RAGs can model Mono-Skeleton-SUMs very easily: inherit the flesh attributes to all nodes
- Between Skeleton and Flesh there holds a **key dependency**
  - A partial function describes the mapping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

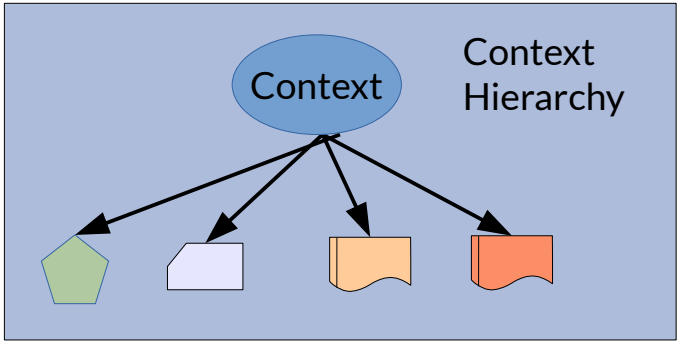
33.3. Context-Based Skeleton-SUM

[Hettel08]
[Seifert11]

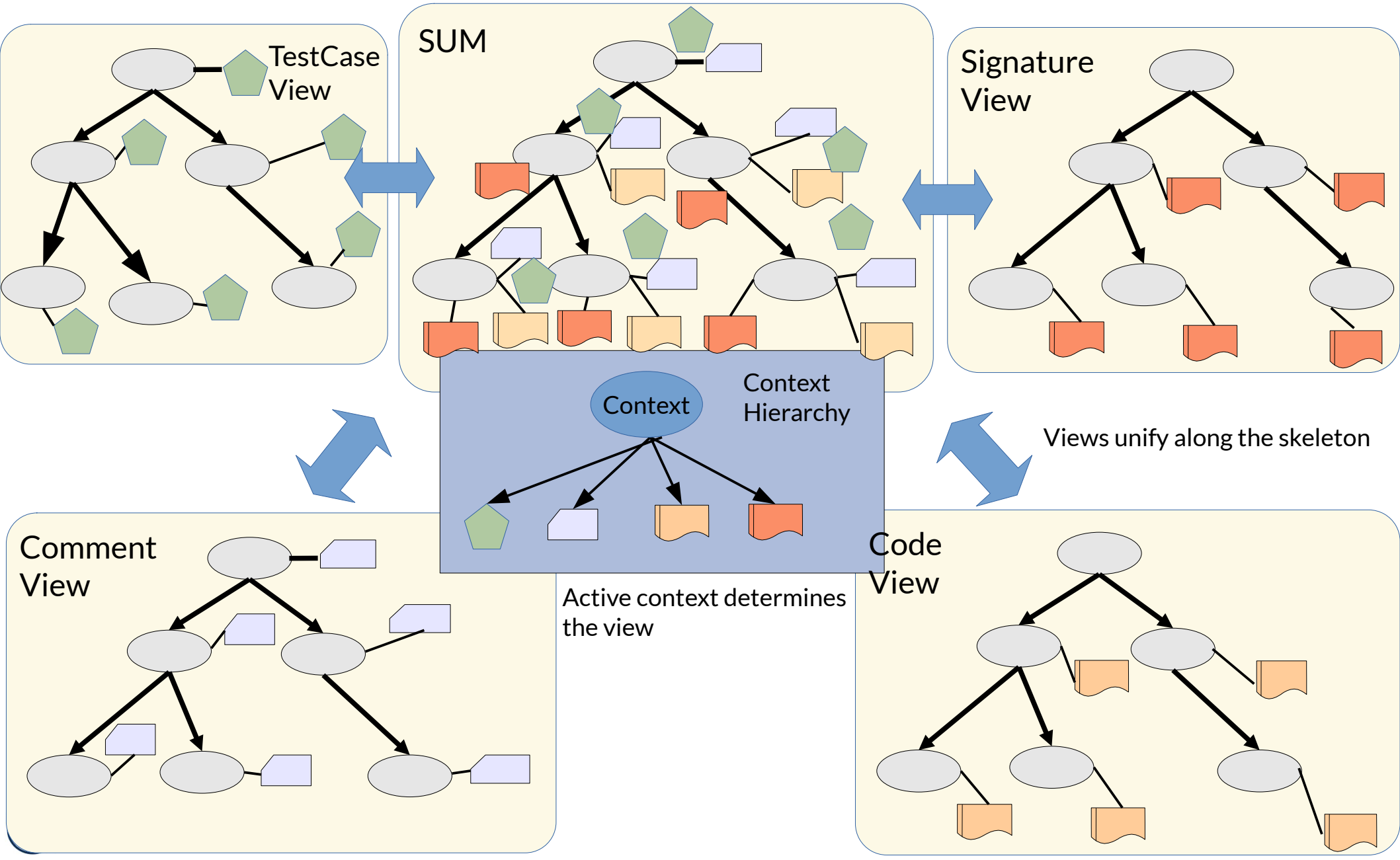
Skeleton-SUM

- 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



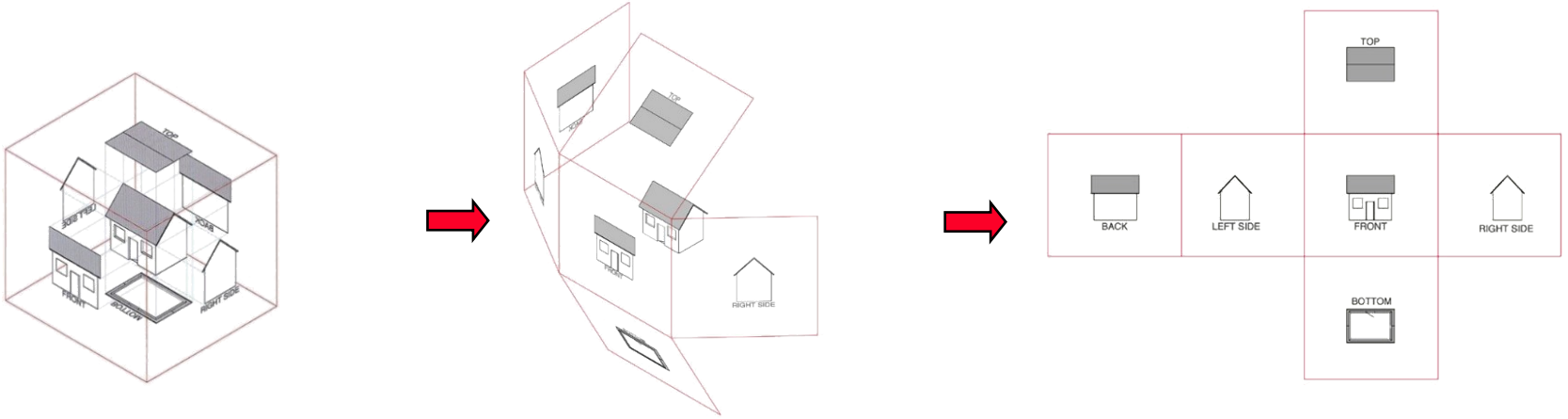
33.3.1. Orthographic Software Modeling (OSM) as a Dimensional, Context-Based Skeleton-SUM

[Hettel08]

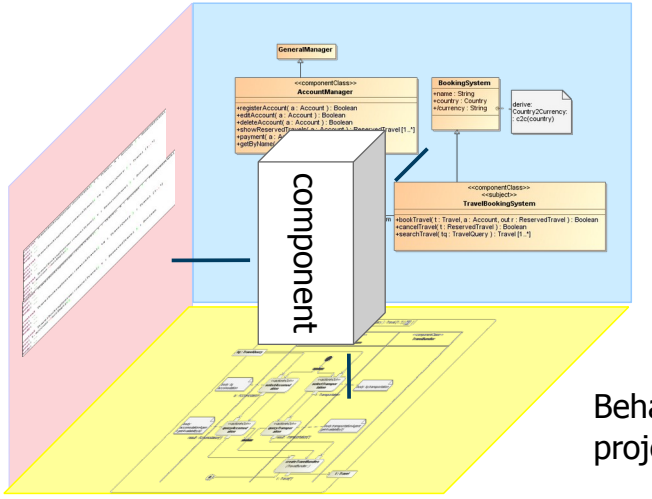
[Seifert11]

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

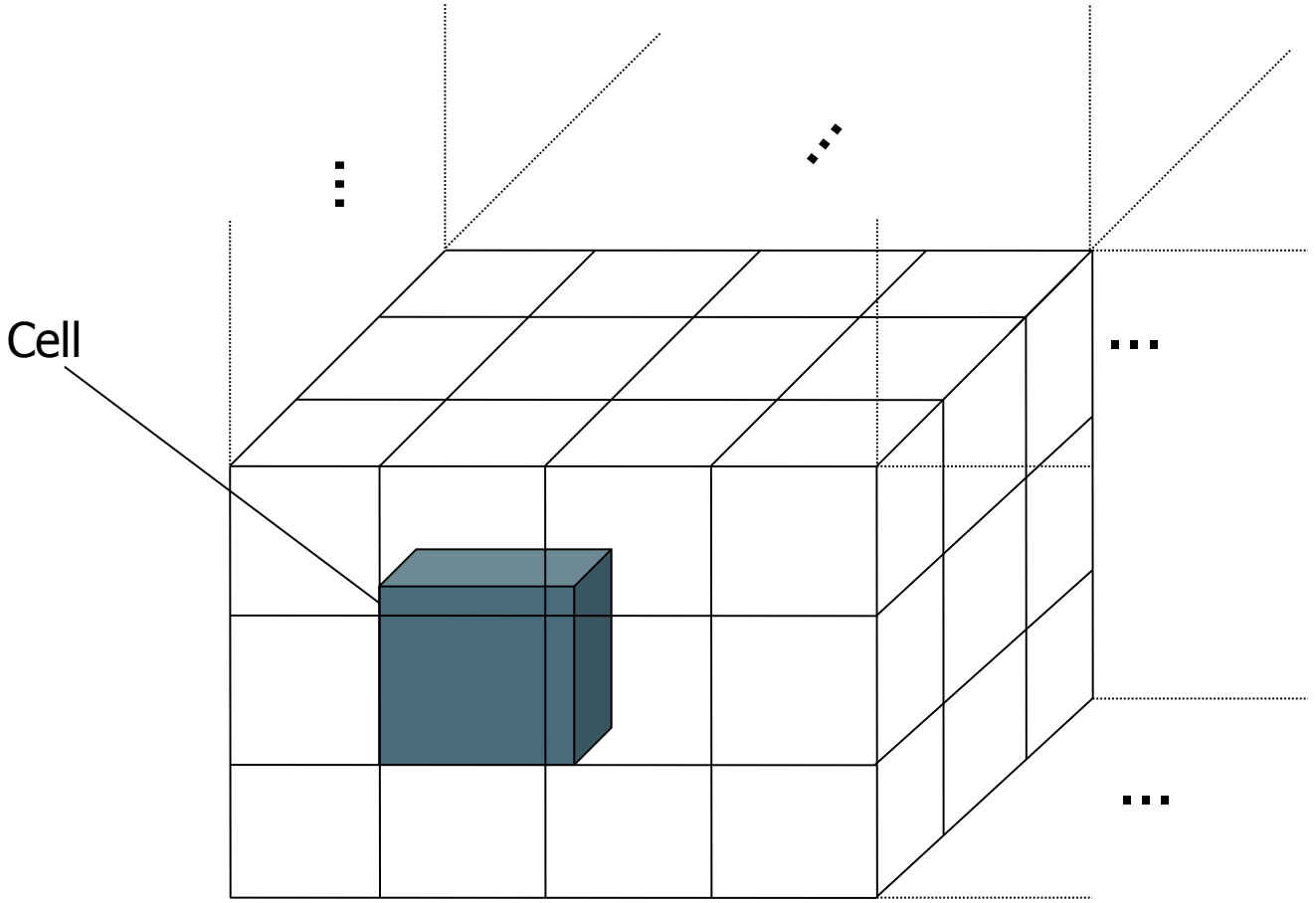
Operational projection

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 dimensions (contexts) are used for projection
- Instead of attributes, model elements have roles (CROM-Skeleton-SUM)

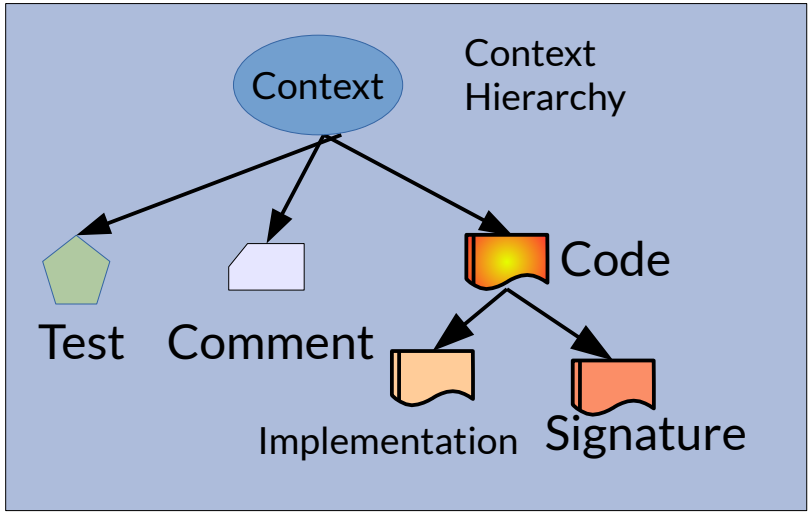
- ROSIMA is a CROM-Skeleton-SUM

33.4. Hierarchic Context-Based Skeleton-SUM

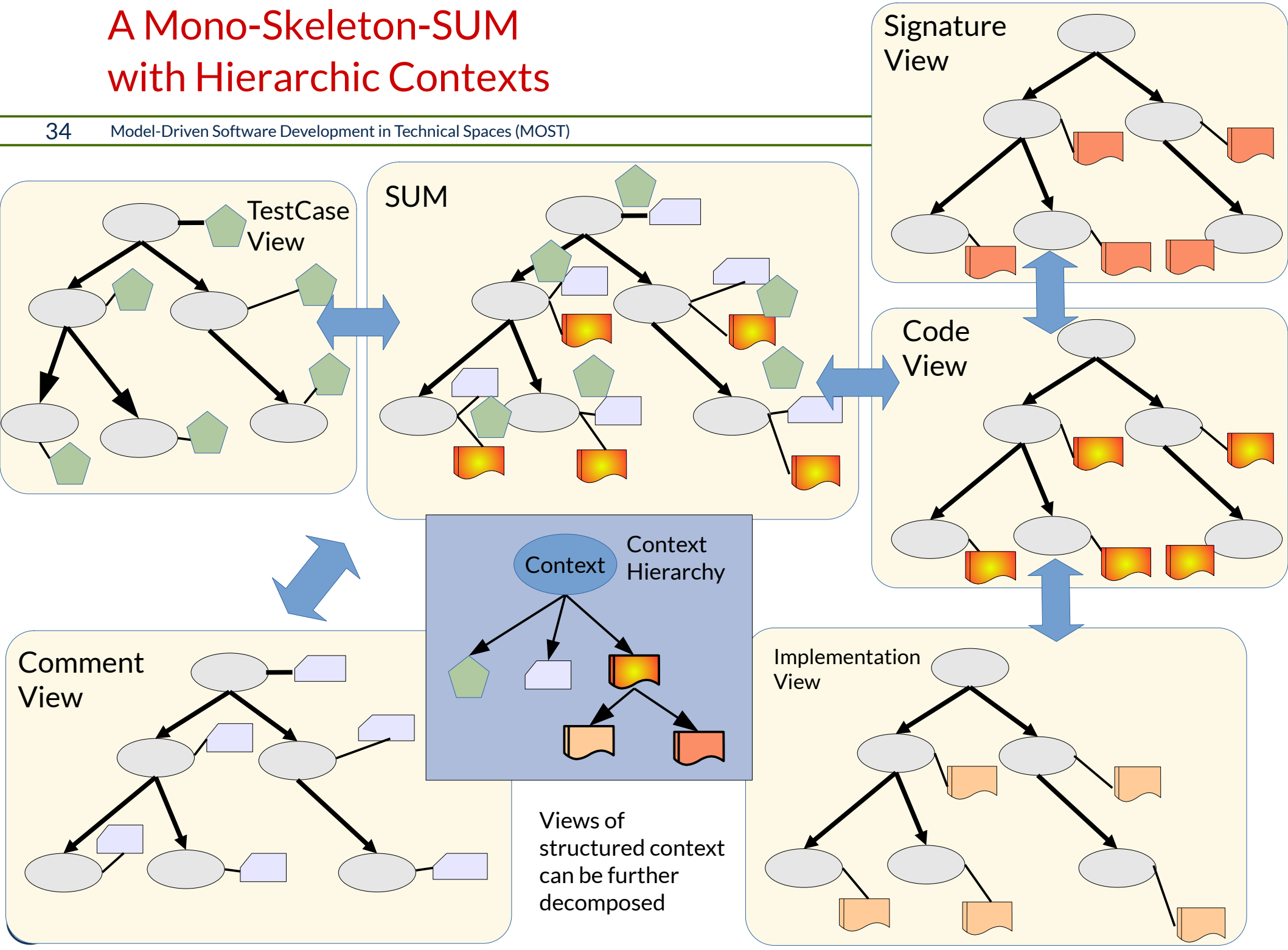
[Hettel08]
[Seifert11]

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

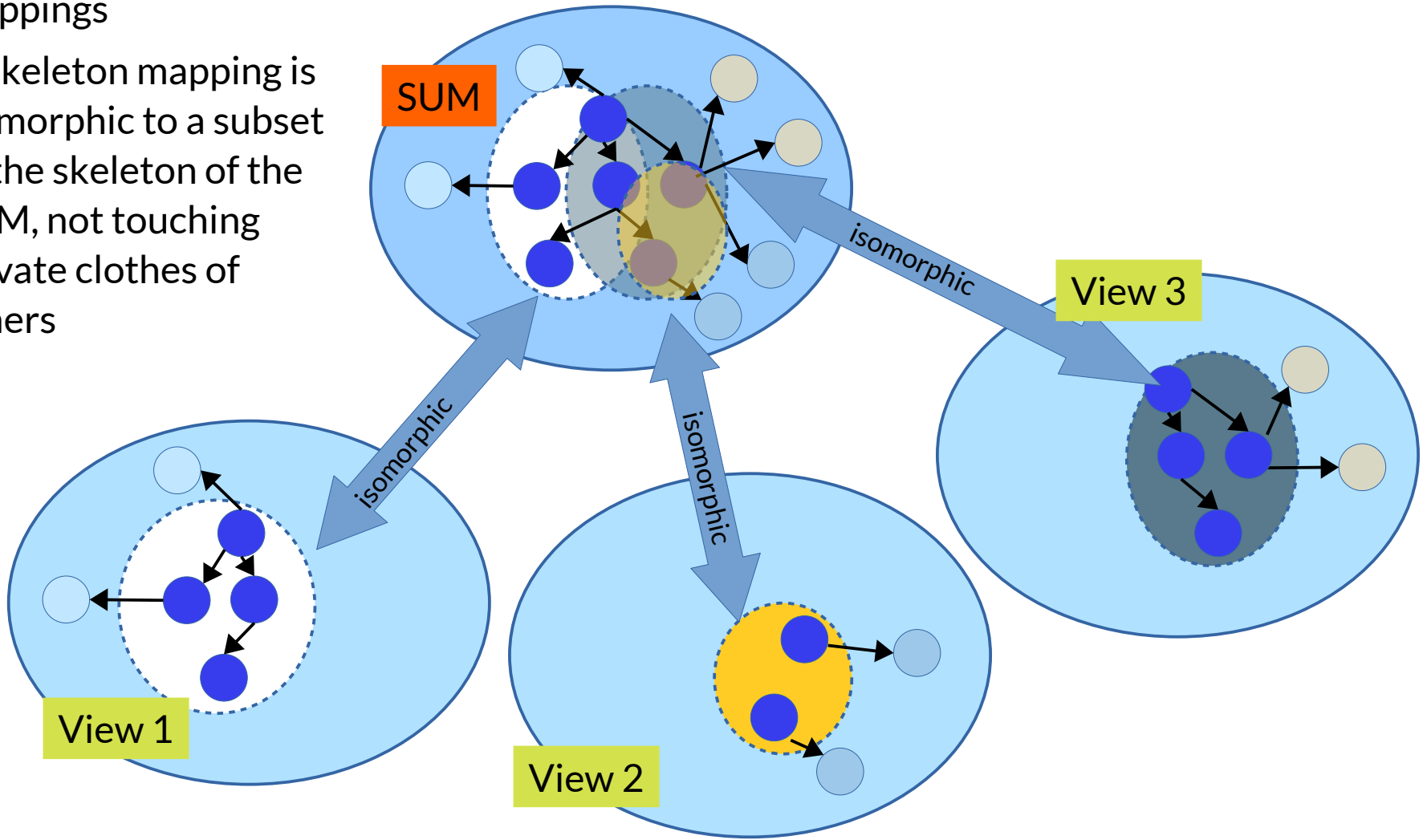


33.5. Multi-Skeleton-SUM

[Seifert11]

Multi-Skeleton-SUM

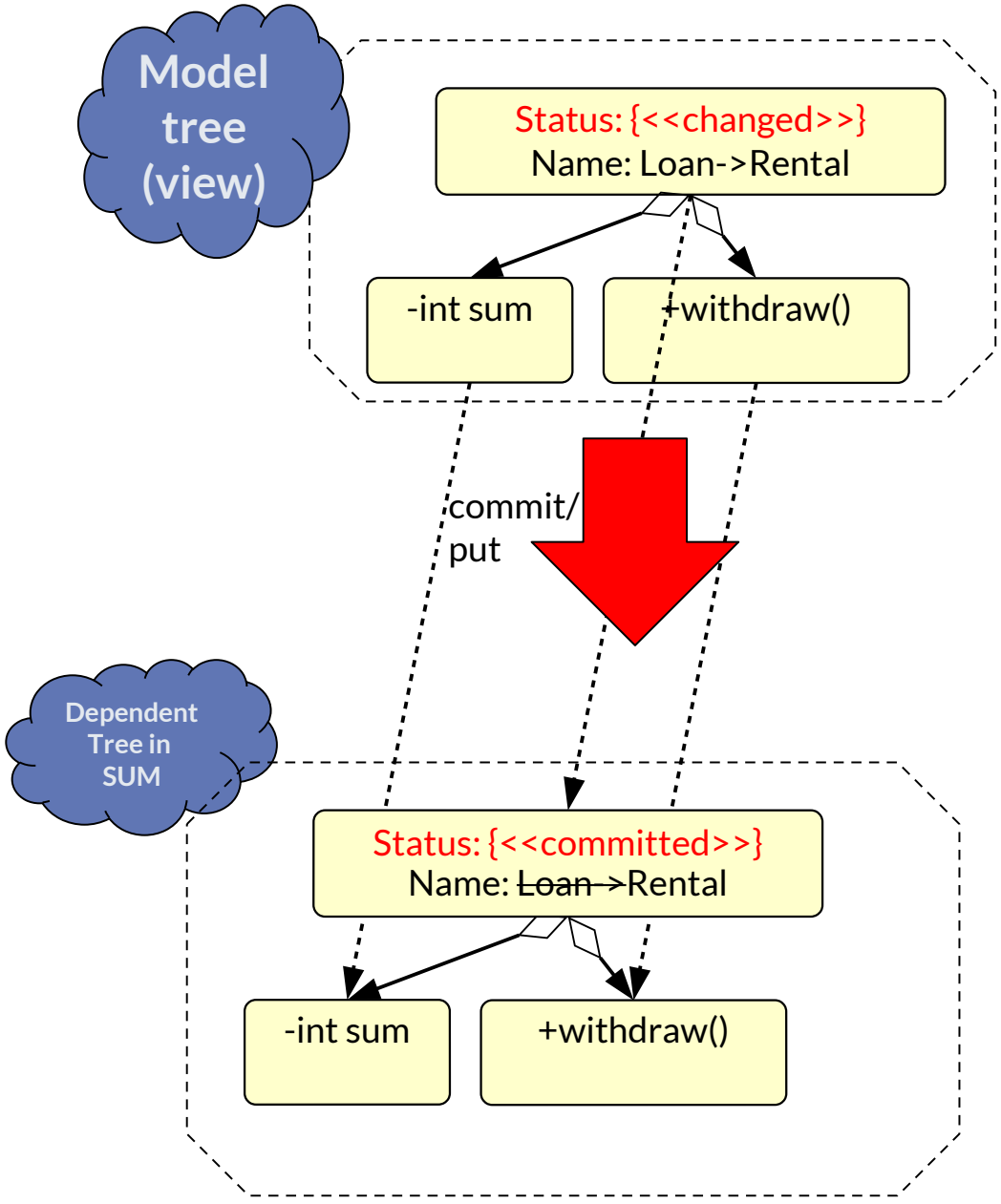
- In SUMs, not all Skeletons need not be a linked by isomorphic mappings
- A Skeleton mapping is isomorphic to a subset of the skeleton of the SUM, not touching private clothes of others
- Every Skeleton must be invariant, and within the SUM, a Skeleton—Skeleton mapping must exist



33.5.2 Put Operations in the MDA-Multi-Skeleton-SUM

Model Synchronization in RAG-MDA by `Put` Operations on Single Underlying Models (SUM)

- A single underlying model (SUM) is a cultimodel with *views*
- MDA can be arranged as MDA-SUM
- A **evolution operation** changes a global name or definition in one model tree a view, which used in several other model trees in the SUM
- To synchronize dependent model elements, we need a `commit/put` operation ("**commit/put to SUM**")
- Its implementation needs to repeat the rewrite in all referencing places
  - Follow the references introduced by global name analysis
  - Standard process in RAG
- Easy traceability by dependency graph between global names



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

[Diskin]

Delta-Based Lenses for Scalability and Applicability

- Simple minded implementation approach –
  - uni-directional *exhaustive* transformations
    - get: SUM-to-view, put: 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 too large grained
  - Not scalable (inefficient)
  - No incrementality
  - transformation more complex than necessary

⇒ The necessary get/put operations are called ***bidirectional lenses***

Delta-Based Lenses and Skeleton SUMs

- ***Lenses*** (Pierce et al. 2007) are pairs of 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:* "intermediary puts can be forgotten"

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

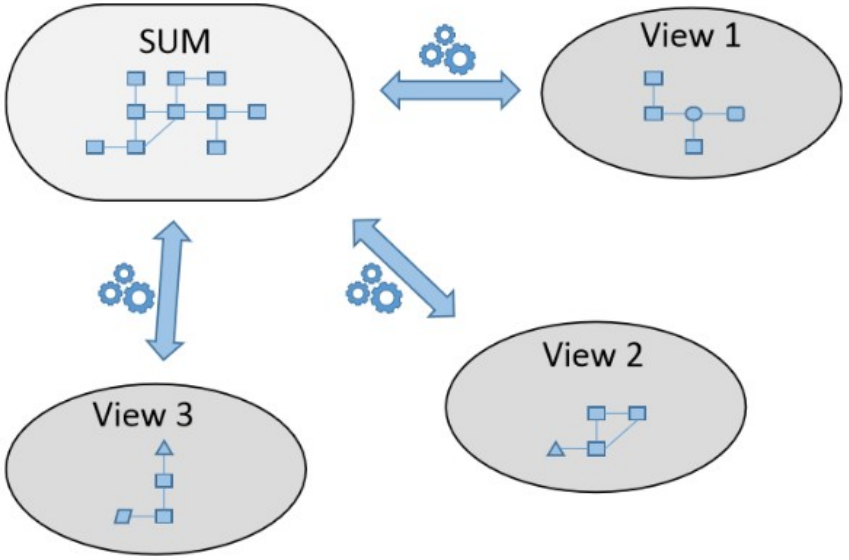
- ***Delta-based Lenses*** optimize the checkin/checkout (Diskin et al. 2011)
  - *Incremental* delta operations dput and dget are driven by the changes to the views
  - axiom for delta-put: "If a delta-commit results in a delta of the SUM, then the next delta-checkout refers only to this delta of the SUM"

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

  - much more fine-grained and scalable

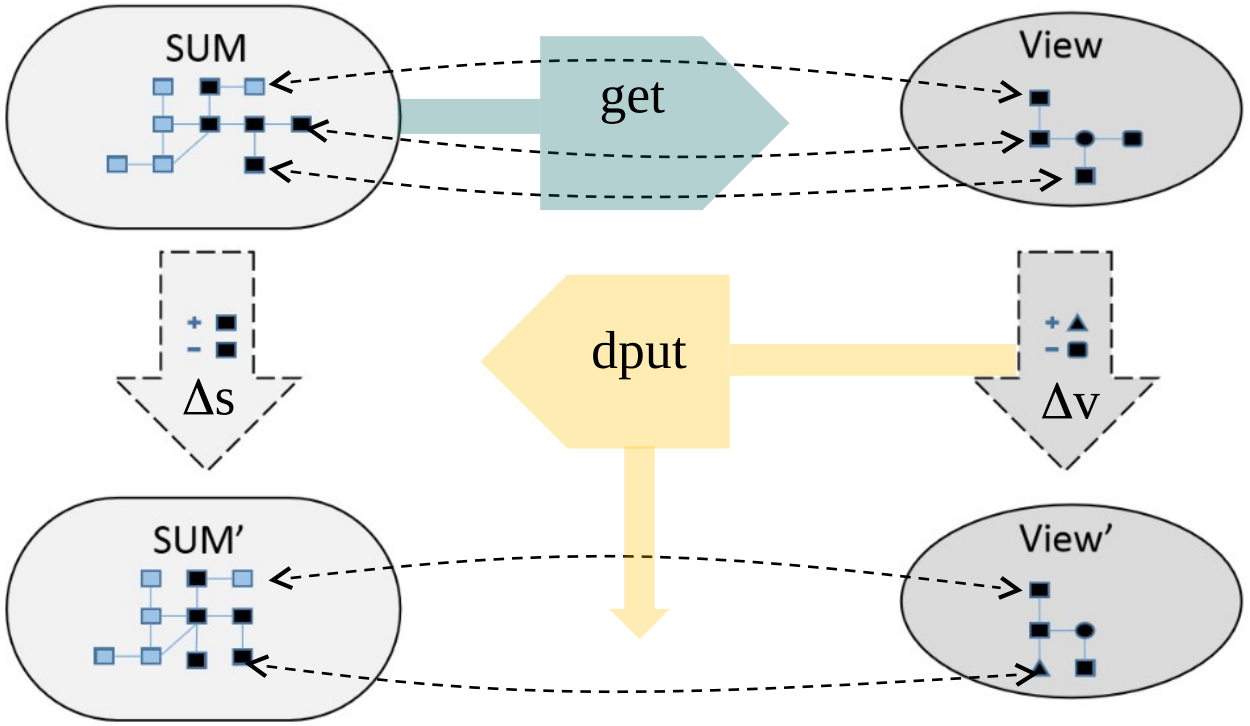
The Background of Orthographic Software Modeling (OSM)

- 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 with dput

- use ***get*** to create views from the SUM
- use ***dput*** (delta put) to update the SUM when a view is changed
  - incremental put operation only transmits the delta (increment)

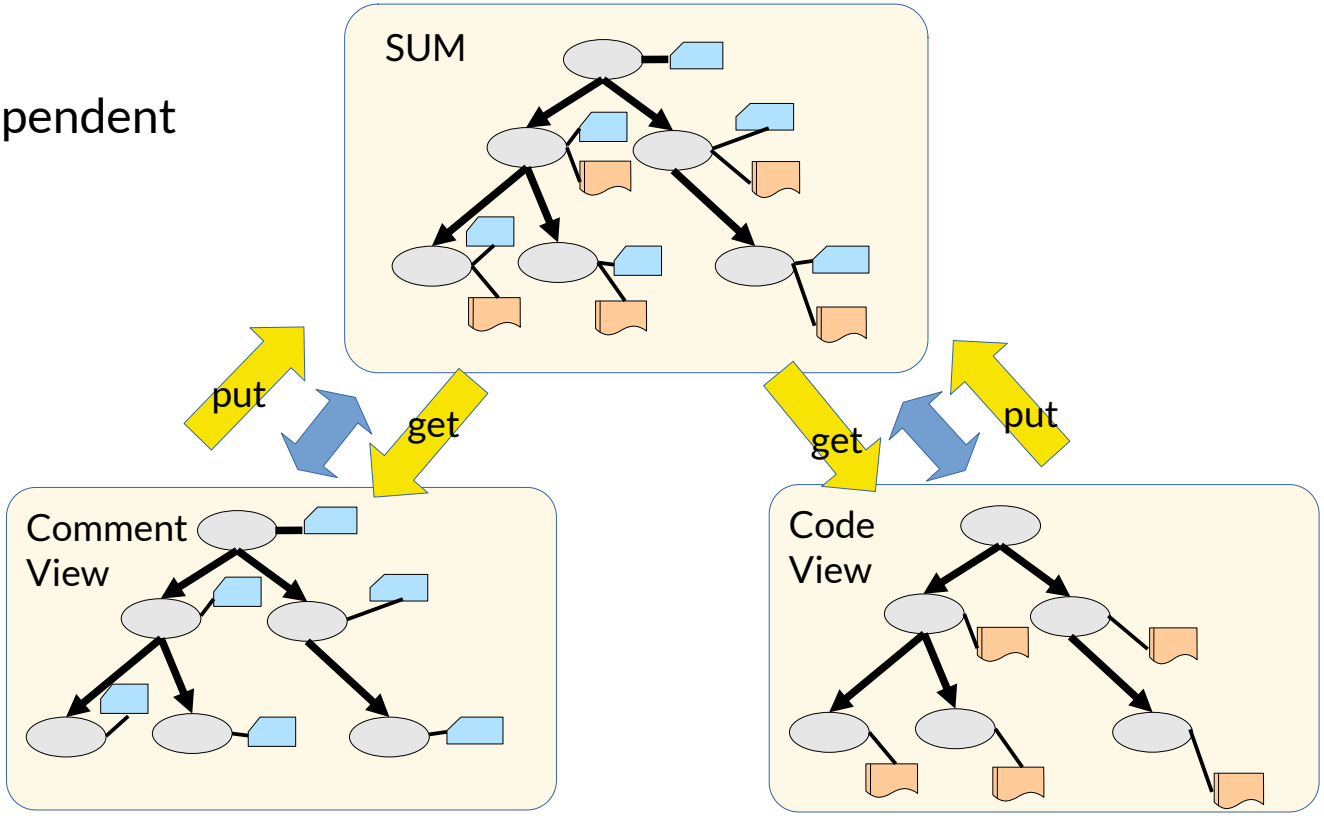


if Δs = dput(Δv, s), then dget(Δs) = Δv // *DeltaPUTPUT rule*

Skeleton-SUM and DeltaPutPut

A Skeleton-SUM fulfills the DeltaPutPut rule.

- Reason:
  - Partial functions are independent
  - Skeleton stays invariant
- Corollary
  - therefore OSM
  - therefore Javadoc-SUM



Pros and Cons of the Hybrid Approach

- **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
  - Traces 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?

33.7 Skeleton-SUM on RoSI CROM

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

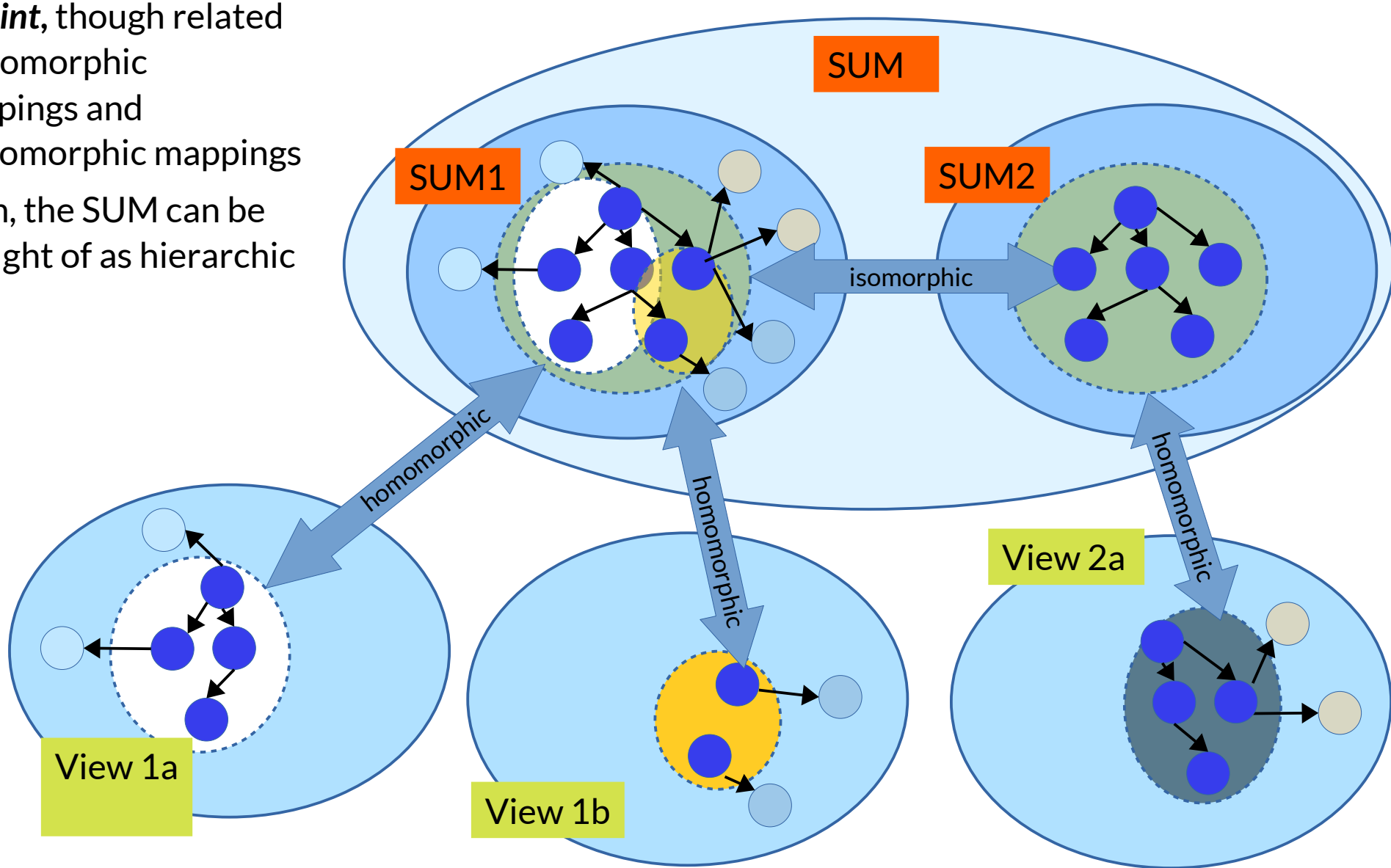
Theorem: A CROM-based Skeleton-SUM fulfils the delta-putput invariant.

33.8. Disjoint-Skeleton-SUM

[Seifert11]

Disjoint-Skeleton-SUM

- Skeletons can be ***disjoint***, though related by isomorphic mappings and homomorphic mappings
- Then, the SUM can be thought of as hierarchic

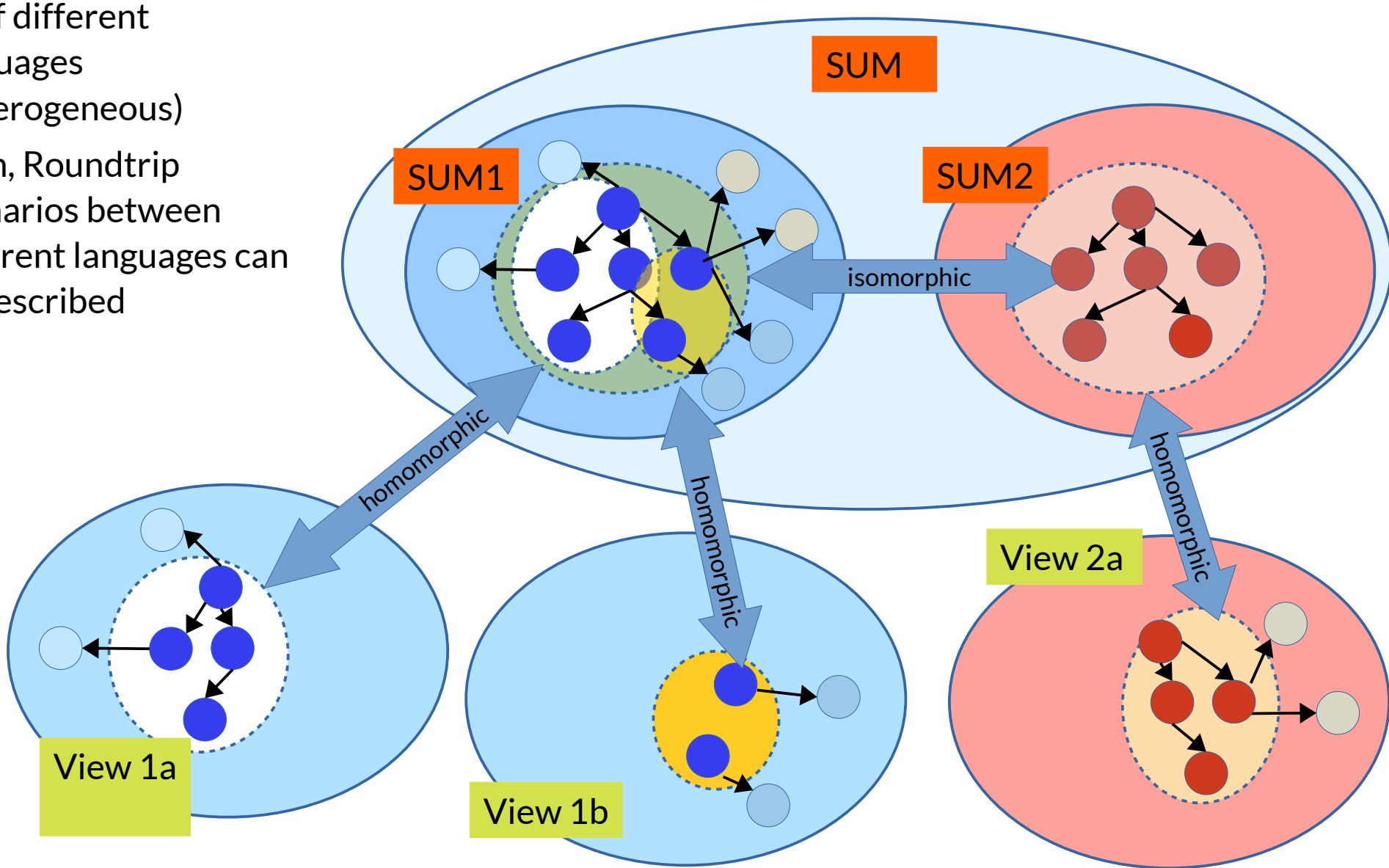


33.8.1 Heterogeneous-Language-Skeleton-SUM

[Seifert11]

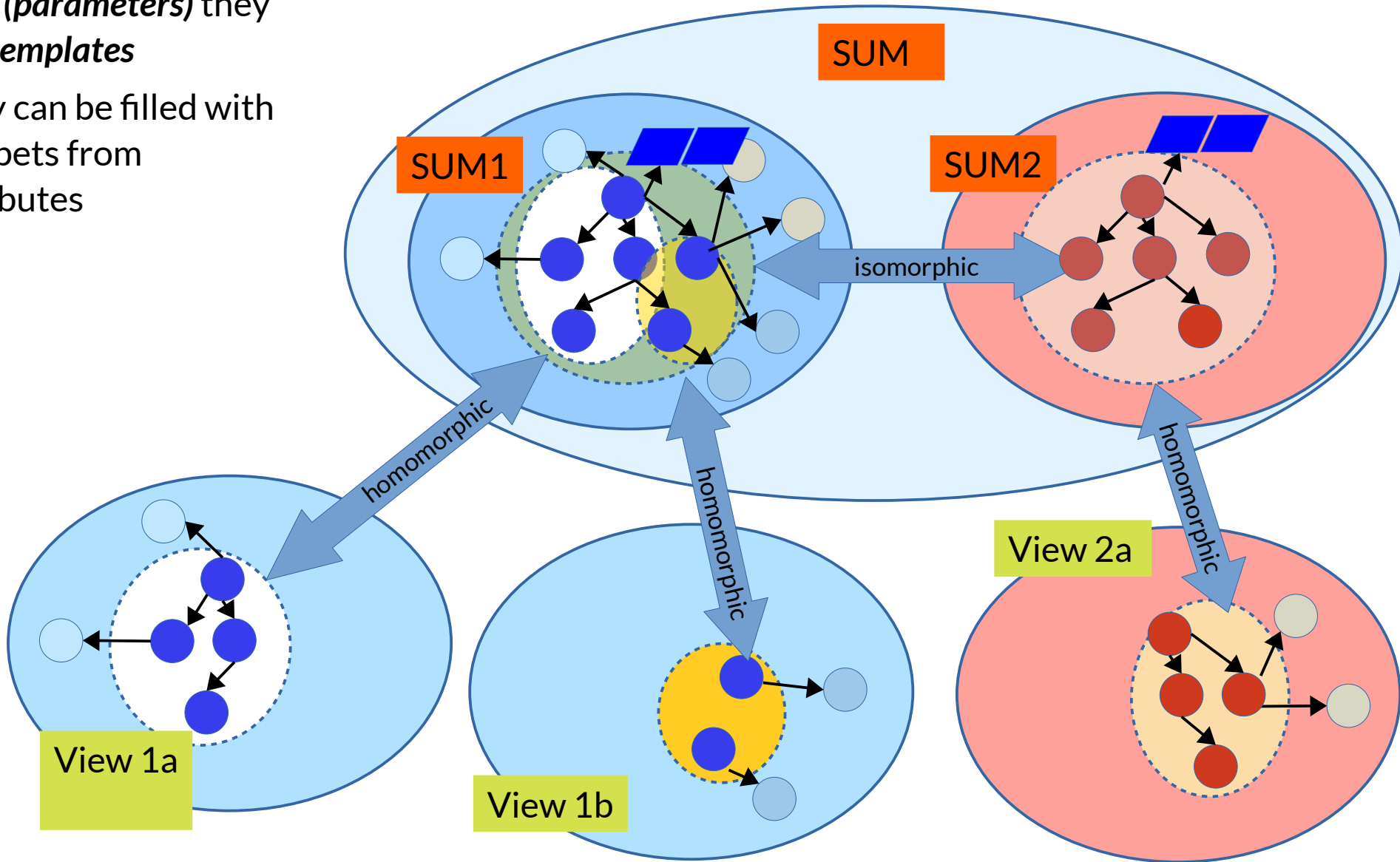
Heterogeneous-Language-Skeleton-SUM

- Disjoing Skeletons can be of different languages (heterogeneous)
- Then, Roundtrip Scenarios between different languages can be described



Heterogeneous-Language-Skeleton-SUM with Templates

- When skeletons have ***slots (parameters)*** they are ***templates***
- They can be filled with snippets from attributes



The End

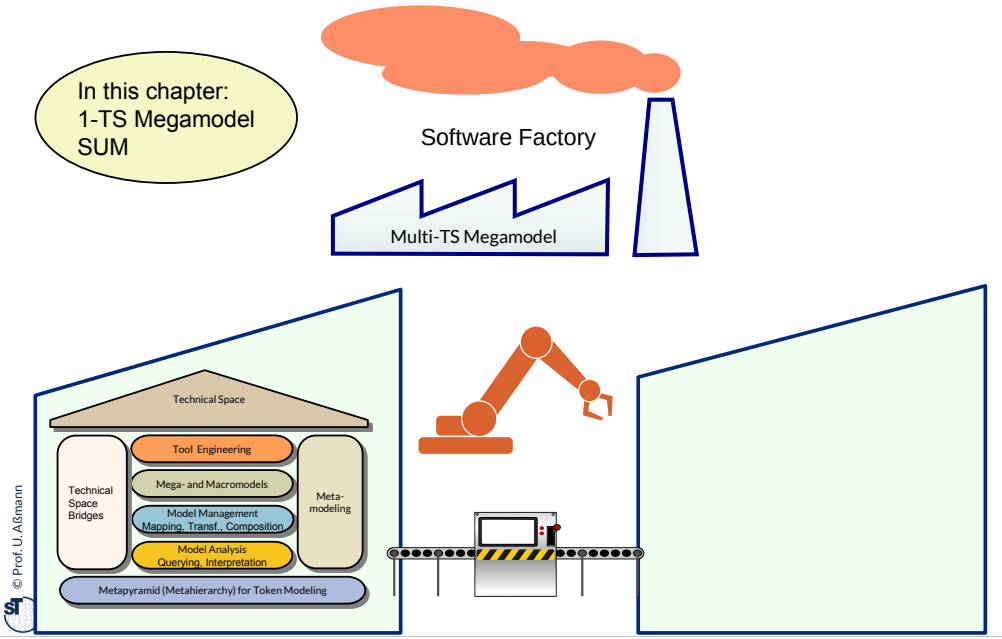
- Explain, how partial functions between objects and attributes enable the projections (get) and the merge functions (put) of a Skeleton-SUM
- Why are contexts important for views?
- What happens if the SUM has several skeletons?
- Which are the contexts of Javadoc-SUM? Why does Javadoc-SUM fulfill the DeltaPutPut rule?
- Which are the contexts of OSM? Why does OSM fulfill the DeltaPutPut rule?
- Why does ROSI-CROM enable Skeleton-SUM?
- Some slides are courtesy to Prof. Colin Atkinson, Mannheim. Used by permission.

33. Macromodel 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 21-1.1, 22.01.22

Software Factories with Only 1 Technical Space



References

- [Atkinson19] Johannes Meier, Heiko Klare, Christian Tunjic, Colin Atkinson, Erik Burger, Ralf Reussner, and Andreas Winter. Single underlying models for projectional, multi-view environments. In Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD, pages 119--130. INSTICC, SciTePress, 2019.
- 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 Link-Tree Macromodels

The Link-Treeware TS is well apt for macromodel construction in a software factory

- A tree node abstracts a subtree (representant)
  - Attributes and attributions are *composable partial mappings* from treenodes
- RAGs are useful for all kinds of structure- and function-modeling in Link-Tree 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
- Relational RAGs (RelRAGs) are useful, because they have bidirectional constraints

	(Plain) MDA	General SUM	Skeleton SUM (partial function extension)
RAGs in Repositories	Markings		Repository-SUM: get/put as higher-order attributions of link trees
			• Javadoc-SUM
RAGs in Data-flow architectures	Needs trace models	get/put as model transformations (lenses)	Flow-SUM: Communicating link trees; In-place transformations of SUM
			• Google Docs, Stream-Based MDA

Other Examples form

- Olympic ring decomposition (EAI) marks all modules with “rings” and thereby decomposes them (course ST-1)
- VSUM (Reussner, Burger et al) generates dependent parts by create trace links

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

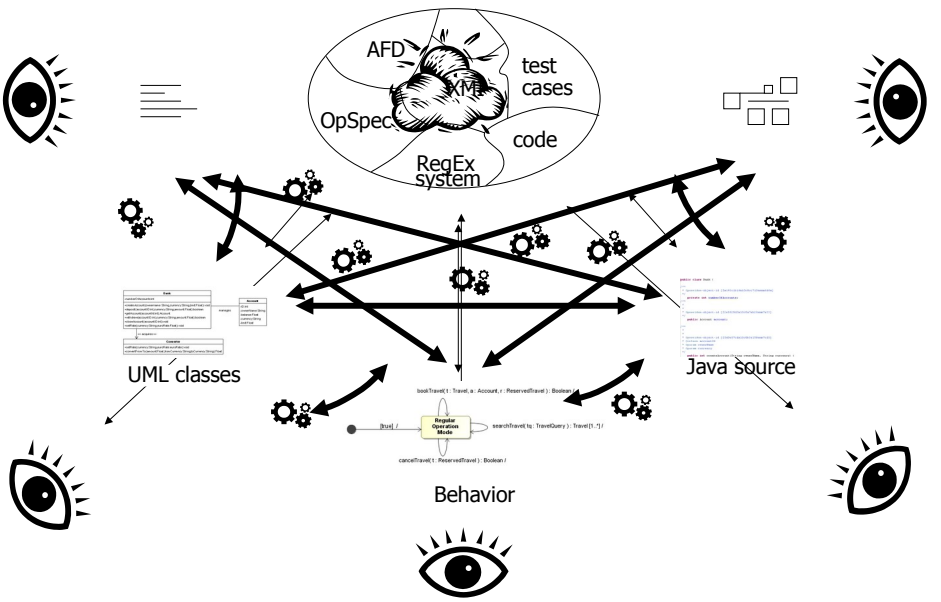
L'Aquila. Italy
21 July, 2015

UNIVERSITÄT MANNHEIM

33.1. The Macromodel “Single-Underlying Model (SUM)”

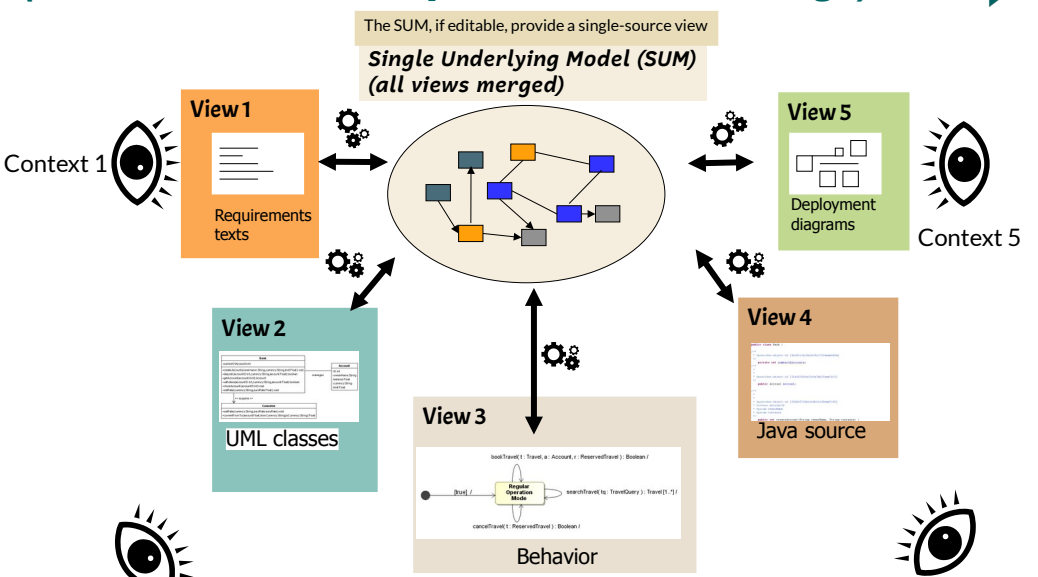
- is based on a ***repository (repository-based SUM)*** [Atkinson19]

Traditional View-based Software Engineering (VOSE)



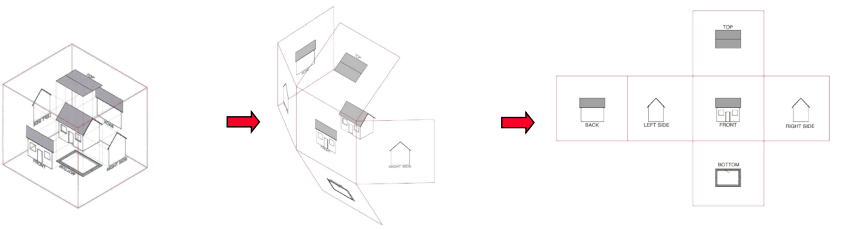
let‘s take a look again at the current status with a picture

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

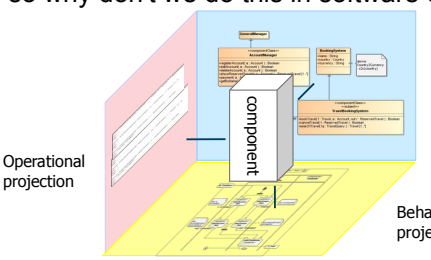


Orthographic Software Modeling (OSM) as a SUM

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

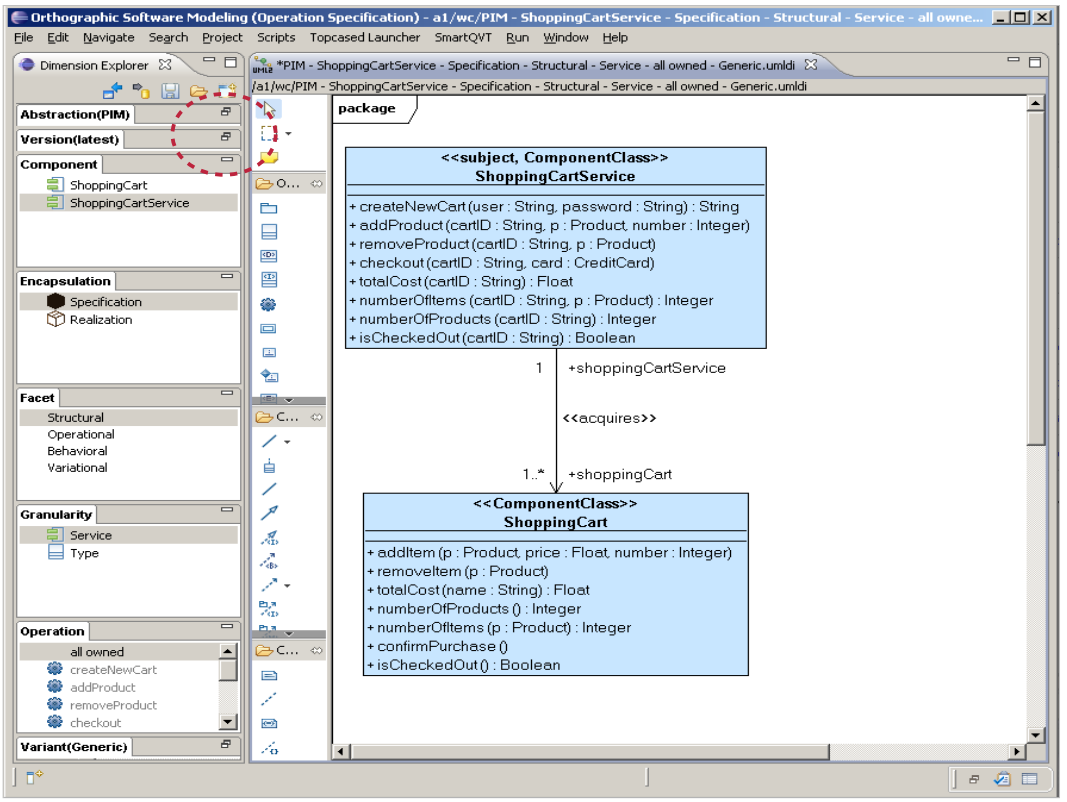


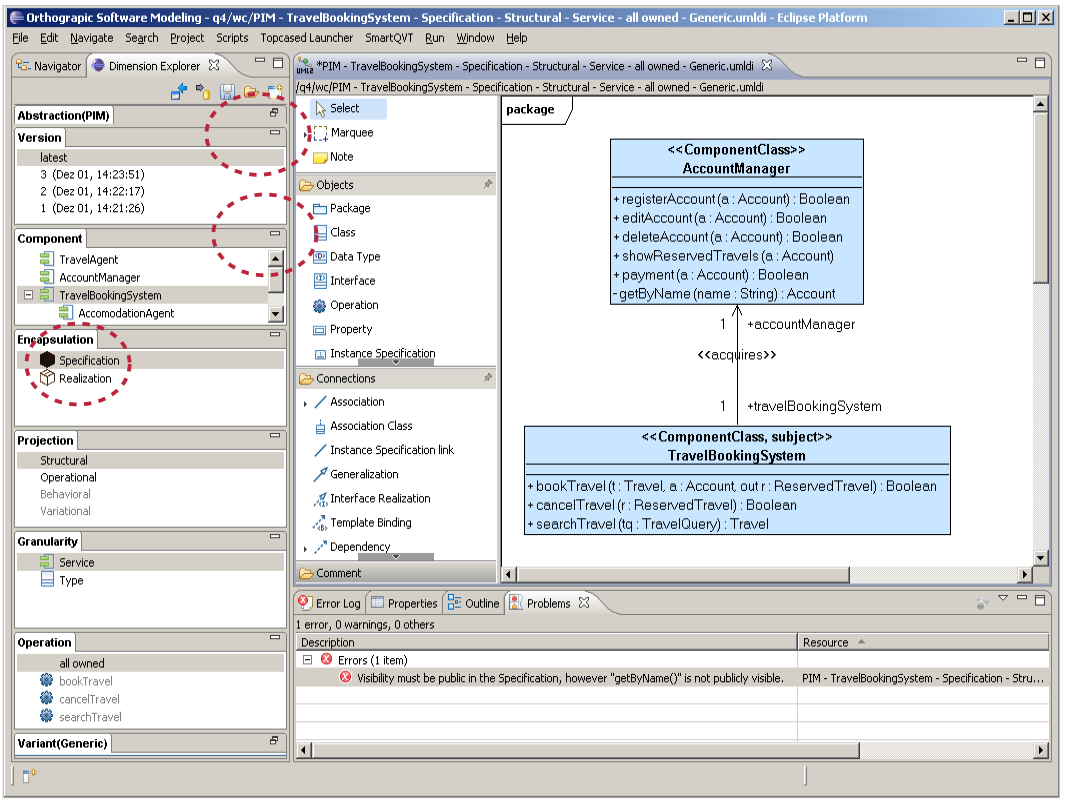
- so why don't we do this in software engineering?

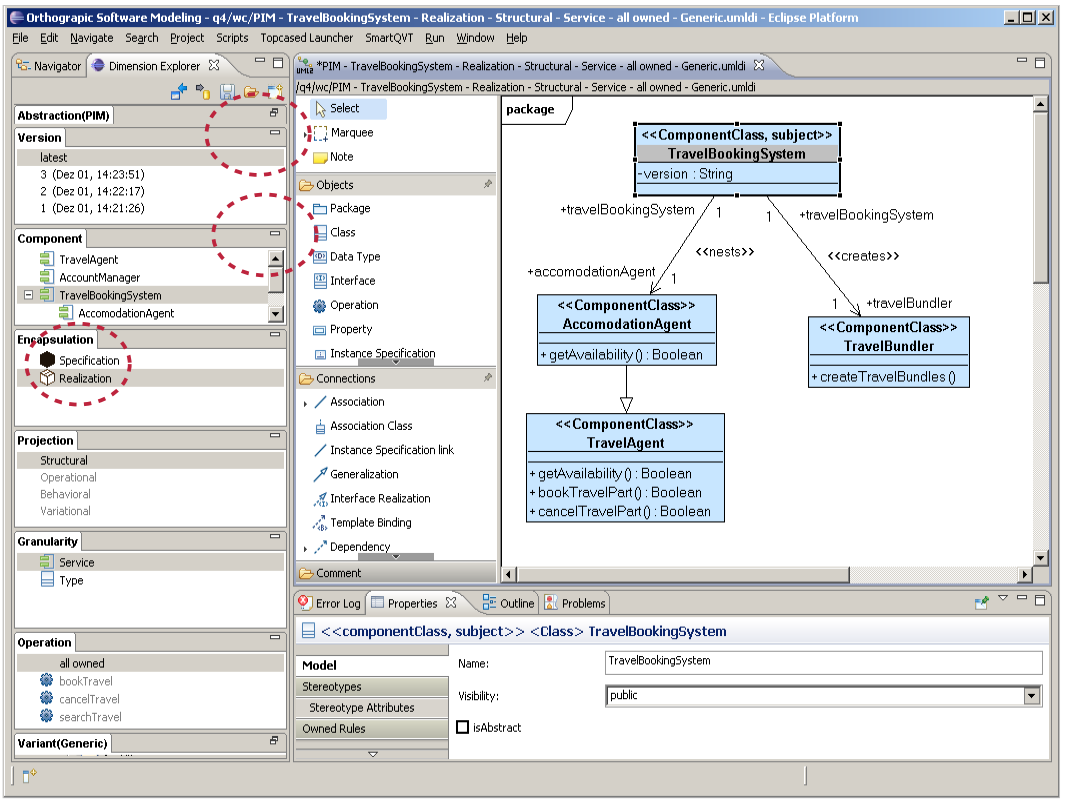


- On demand view generation (projective views)
- Dimension-based navigation
- View-based methodology
- Arrangement in a multidimentional SUM

apply this metaphor to SE


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...

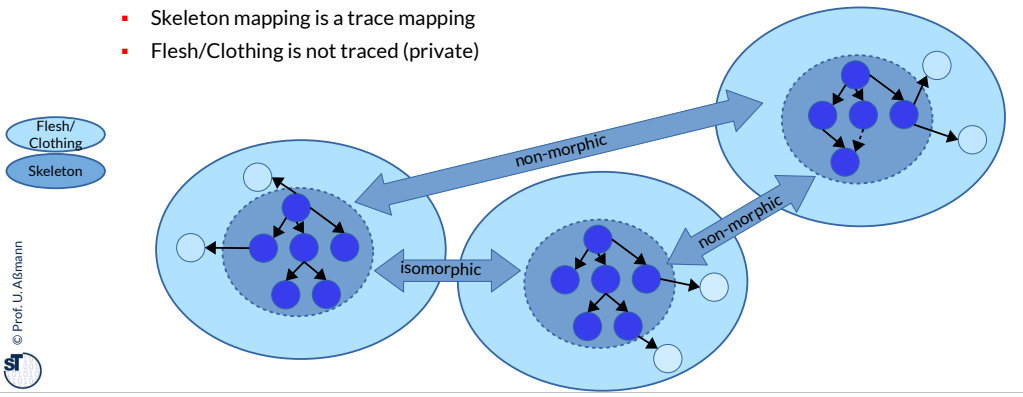

Orthograpic Software Modeling - q4/wc/PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi - Eclipse Platform
File Edit Navigate Search Project Scripts Topcased Launcher SmartQVT Run Window Help
Navigator
Dimension Explorer
Abstraction(PIM)
Version
latest
3 (Dez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

33.2. The Skeleton-SUM

[Hettel08]
[Seifert11]

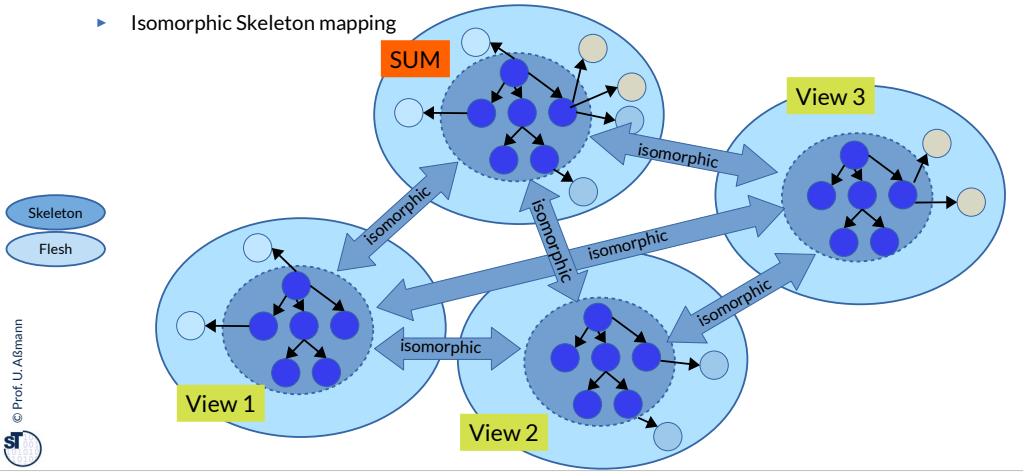
Skeletons and Flesh

- 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
  - Skeleton mapping is a trace mapping
  - Flesh/Clothing is not traced (private)



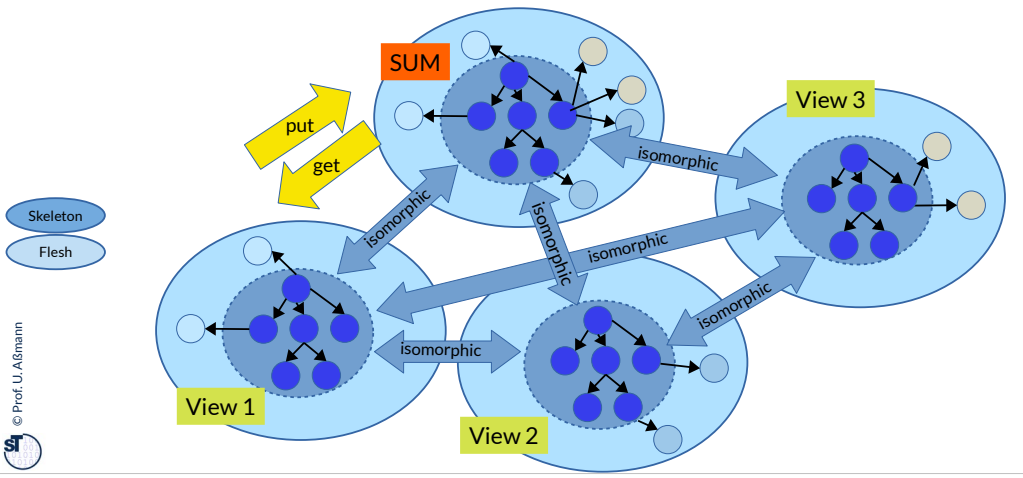
Mono-Skeleton-SUM

- Mono-Skeleton-SUM splits models into
  - **One common Skeleton** (redundant) (several contexts)
  - and **flesh (clothing)** (locally different stuff in views, mono-context) is stored in SUM together with skeleton
- Flesh must be non-overlapping, extending the skeleton
- Isomorphic Skeleton mapping



Get/Put in Mono-Skeleton-SUM

- From a Skeleton-SUM
  - get operation produces a view
  - put operation commits it into SUM

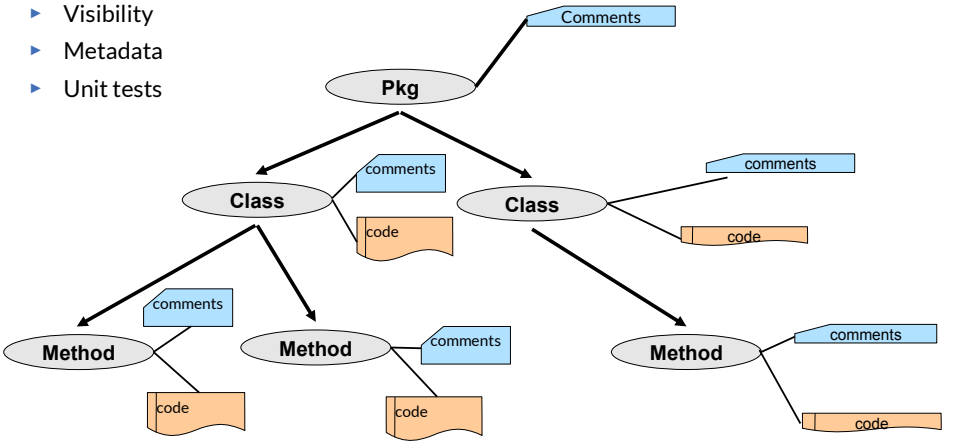


33.2.1 Javadoc-SUM, a Mono-Skeleton-SUM for Documentation

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

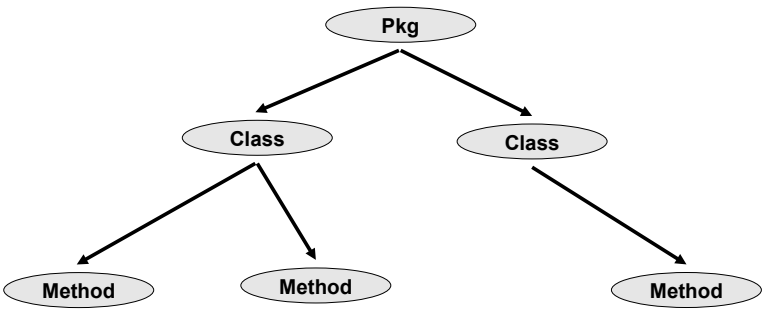
Javadoc comment relies on several attributes of nodes of the syntax tree:

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



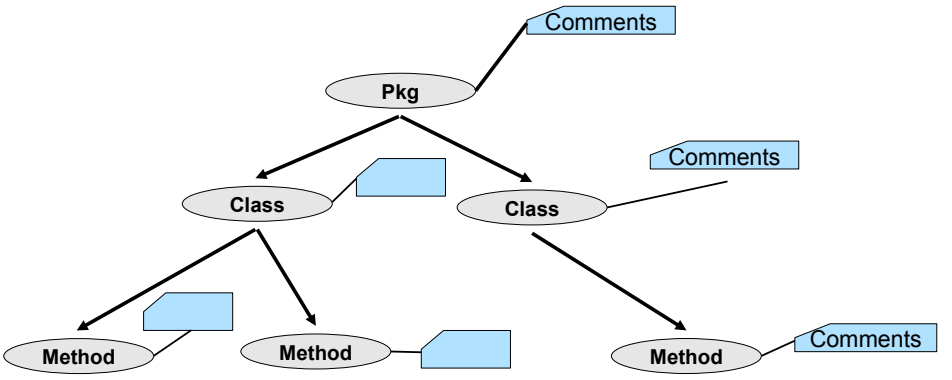
Projecting A Scope Tree for Skeleton

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



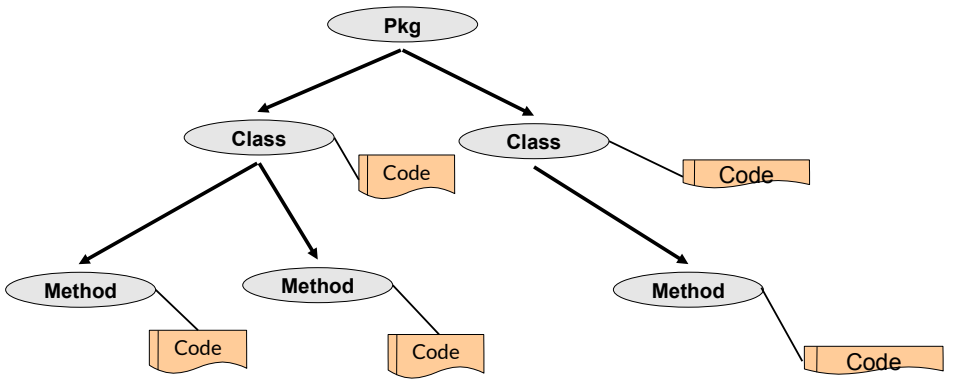
Projecting A Scope Tree for Skeleton

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



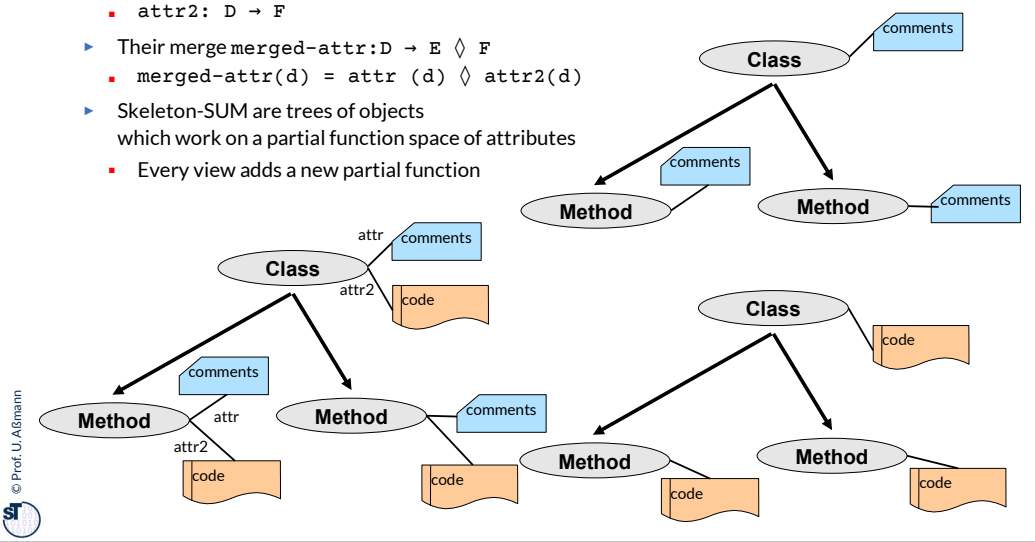
Projecting A Scope Tree for Skeleton

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

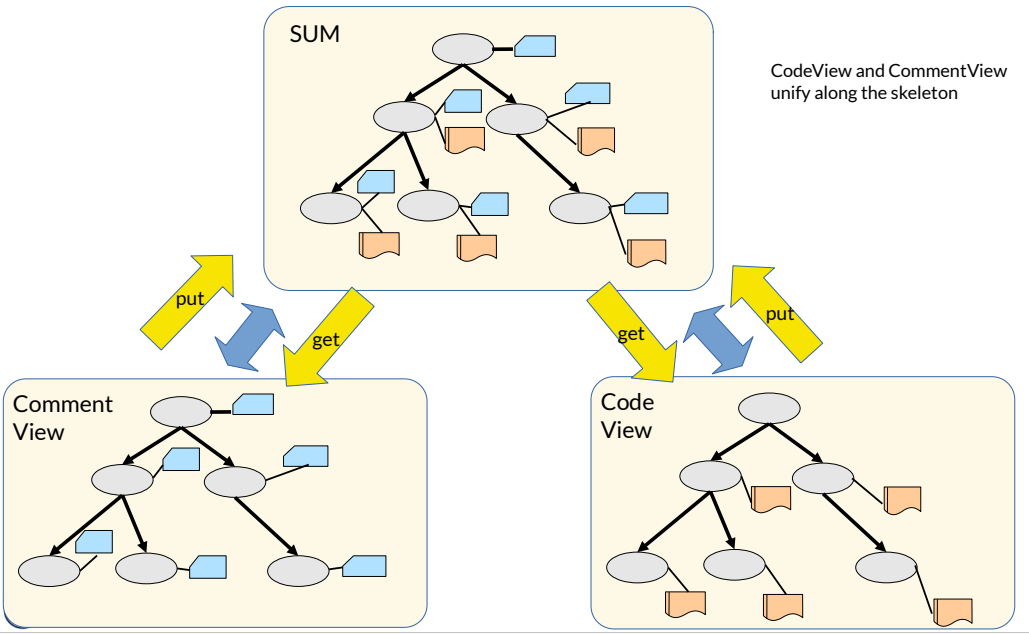


Merge of Partial Functions and Partial Trees in a Mono-Skeleton-SUM

- Given two partial functions on tree-node domain D and two domains E, F:
  - `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



Javadoc-SUM:
A Simple Metamodel-based Mono-Skeleton-SUM



Remarks on Mono-Skeleton-SUM

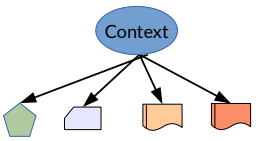
- **Generality:** The Skeleton need not be a link tree; it can be an arbitrary graph data structure
  - But RAGs can model Mono-Skeleton-SUMs very easily: inherit the flesh attributes to all nodes
- Between Skeleton and Flesh there holds a **key dependency**
  - A partial function describes the mapping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

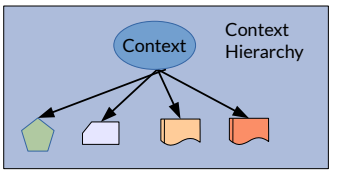
33.3. Context-Based Skeleton-SUM

[Hettel08]
[Seifert11]

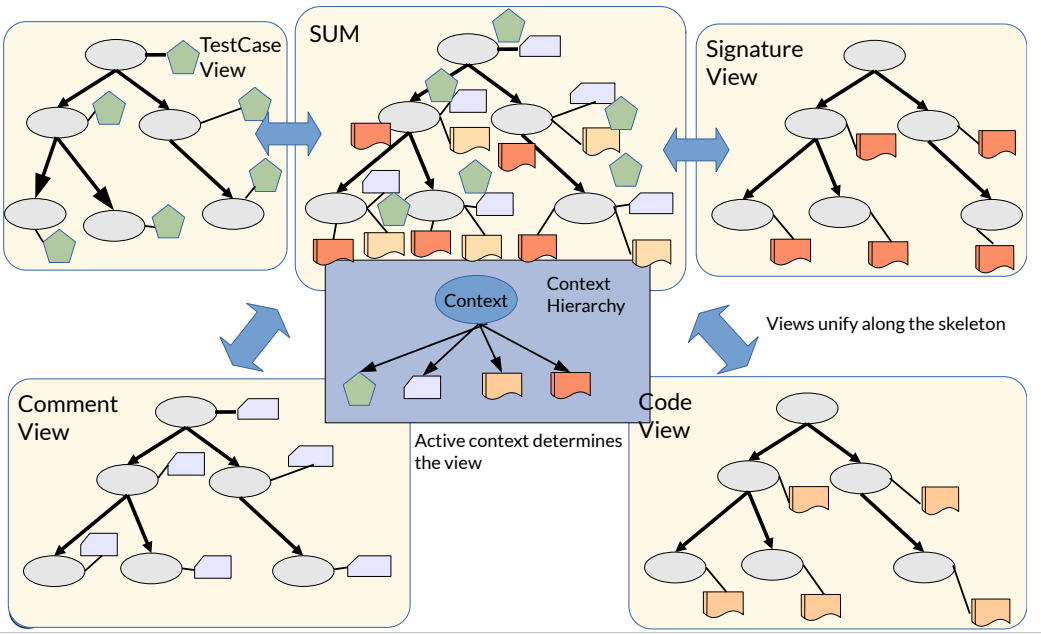
Skeleton-SUM

- 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



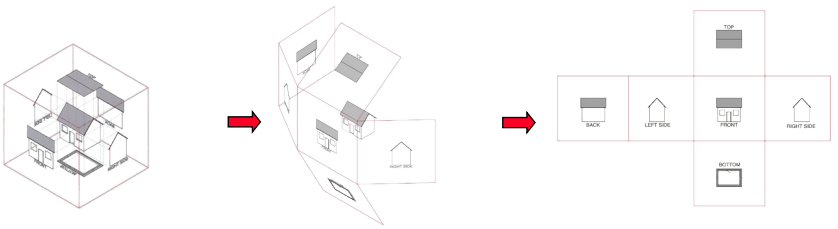
33.3.1. Orthographic Software Modeling (OSM) as a Dimensional, Context-Based Skeleton-SUM

[Hettel08]
[Seifert11]

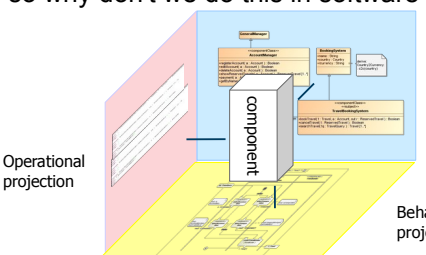
DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

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?

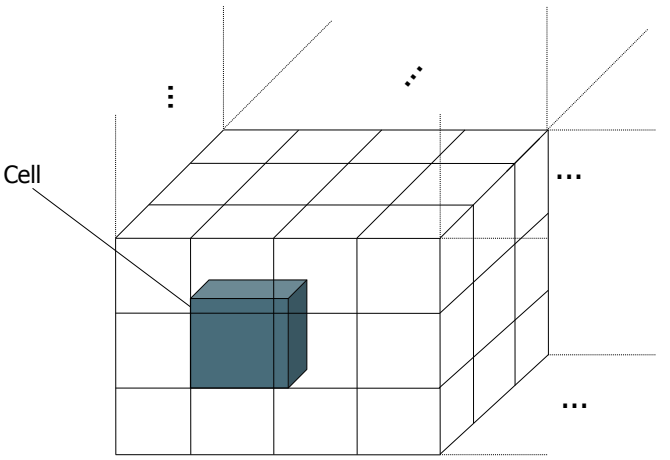


- On demand view generation (projective views)
- Dimension-based navigation
- View-based methodology

apply this metaphor to SE

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 dimensions (contexts) are used for projection
- Instead of attributes, model elements have roles (CROM-Skeleton-SUM)

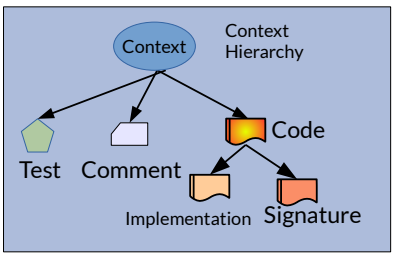
- ROSIMA is a CROM-Skeleton-SUM

33.4. Hierarchic Context-Based Skeleton-SUM

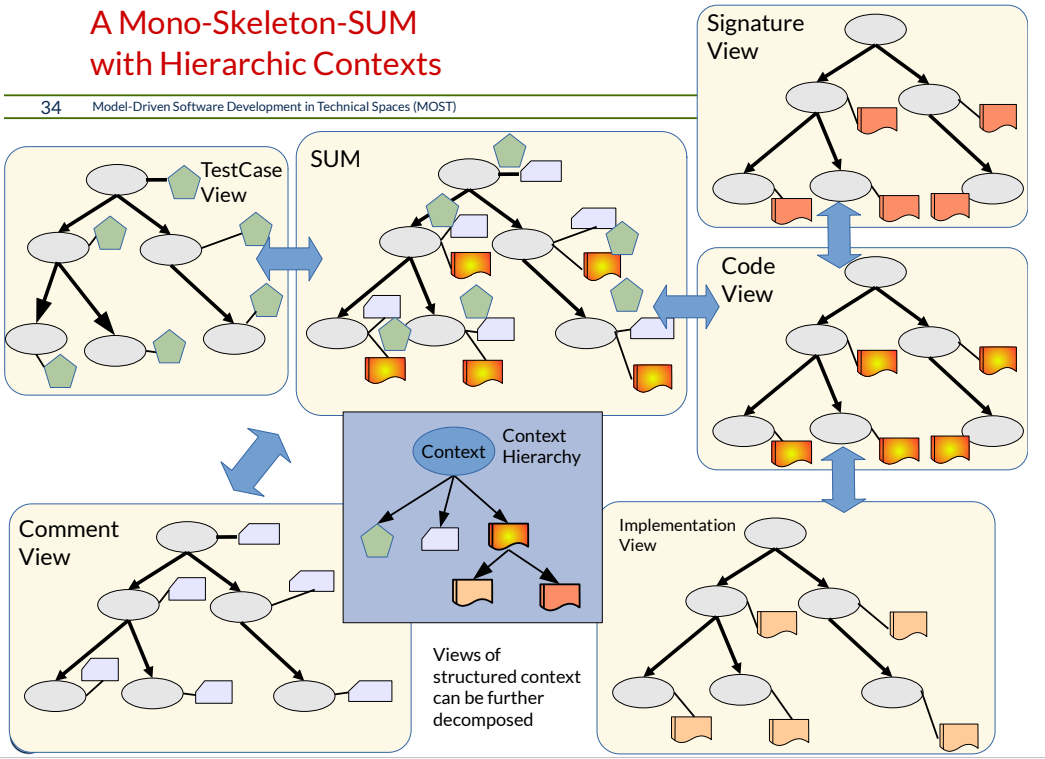
[Hettel08]
[Seifert11]

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

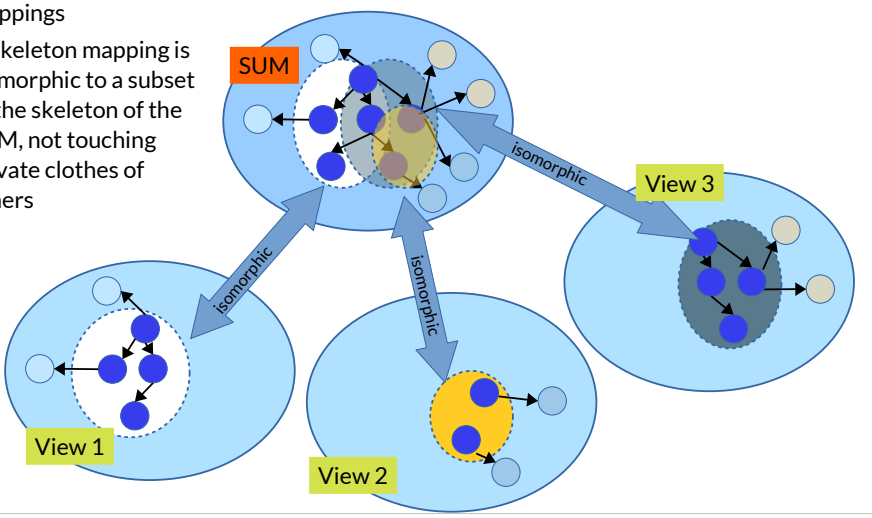


33.5. Multi-Skeleton-SUM

[Seifert11]

Multi-Skeleton-SUM

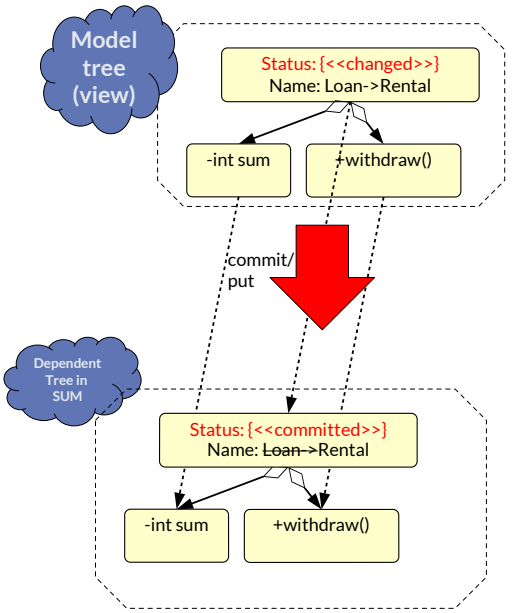
- In SUMs, not all Skeletons need not be a linked by isomorphic mappings
- A Skeleton mapping is isomorphic to a subset of the skeleton of the SUM, not touching private clothes of others
- Every Skeleton must be invariant, and within the SUM, a Skeleton—Skeleton mapping must exist



33.5.2 Put Operations in the MDA-Multi-Skeleton-SUM

Model Synchronization in RAG-MDA by `Put` Operations on Single Underlying Models (SUM)

- A single underlying model (SUM) is a cultimodel with *views*
- MDA can be arranged as MDA-SUM
- A **evolution operation** changes a global name or definition in one model tree a view, which used in several other model trees in the SUM
- To synchronize dependent model elements, we need a `commit/put` operation ("**commit/put to SUM**")
- Its implementation needs to repeat the rewrite in all referencing places
  - Follow the references introduced by global name analysis
  - Standard process in RAG
- Easy traceability by dependency graph between global names



Example: different class implementations of a connector class in a PIM

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

[Diskin]

Delta-Based Lenses for Scalability and Applicability

- Simple minded implementation approach –
  - uni-directional *exhaustive* transformations
    - get: SUM-to-view, put: 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 too large grained
  - Not scalable (inefficient)
  - No incrementality
  - transformation more complex than necessary

⇒ The necessary get/put operations are called ***bidirectional lenses***

Delta-Based Lenses and Skeleton SUMs

- ***Lenses*** (Pierce et al. 2007) are pairs of 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:* "intermediary puts can be forgotten"

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

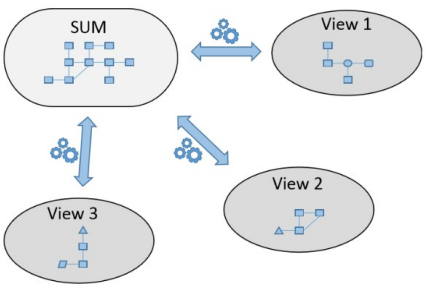
- ***Delta-based Lenses*** optimize the checkin/checkout (Diskin et al. 2011)
  - *Incremental* delta operations dput and dget are driven by the changes to the views
  - axiom for delta-put: "If a delta-commit results in a delta of the SUM, then the next delta-checkout refers only to this delta of the SUM"

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

  - much more fine-grained and scalable

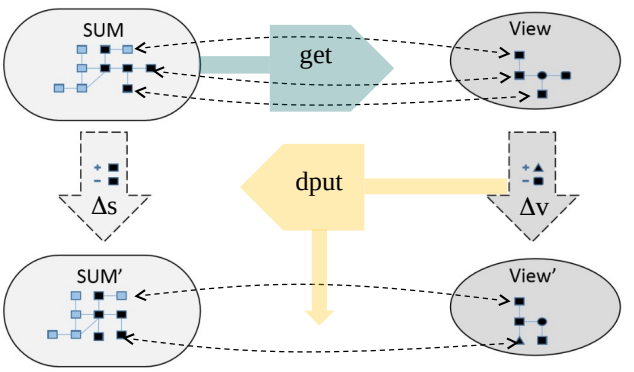
The Background of Orthographic Software Modeling (OSM)

- 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 with dput

- use ***get*** to create views from the SUM
- use ***dput*** (delta put) to update the SUM when a view is changed
  - incremental put operation only transmits the delta (increment)

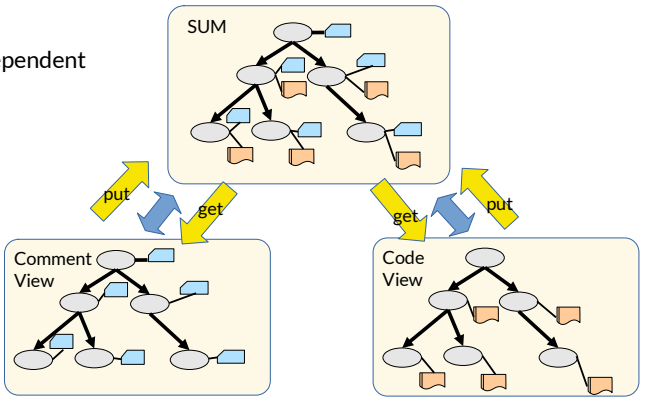


if Δs = dput(Δv, s), then dget(Δs) = Δv // *DeltaPUTPUT rule*

Skeleton-SUM and DeltaPutPut

A Skeleton-SUM fulfills the DeltaPutPut rule.

- Reason:
  - Partial functions are independent
  - Skeleton stays invariant
- Corollary
  - therefore OSM
  - therefore Javadoc-SUM



Pros and Cons of the Hybrid Approach

- **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
  - Traces 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?

33.7 Skeleton-SUM on RoSI CROM

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

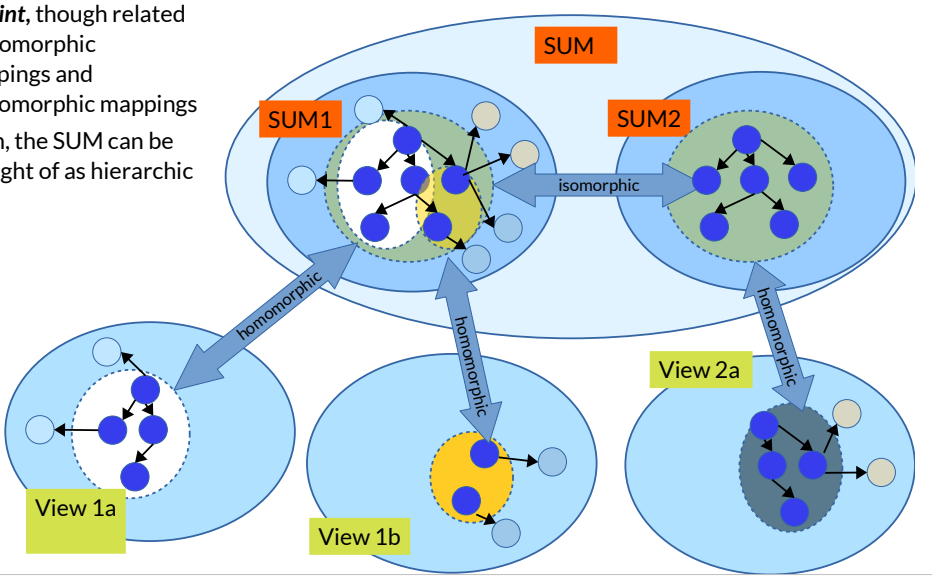
Theorem: A CROM-based Skeleton-SUM fulfils the delta-putput invariant.

33.8. Disjoint-Skeleton-SUM

[Seifert11]

Disjoint-Skeleton-SUM

- Skeletons can be ***disjoint***, though related by isomorphic mappings and homomorphic mappings
- Then, the SUM can be thought of as hierarchic

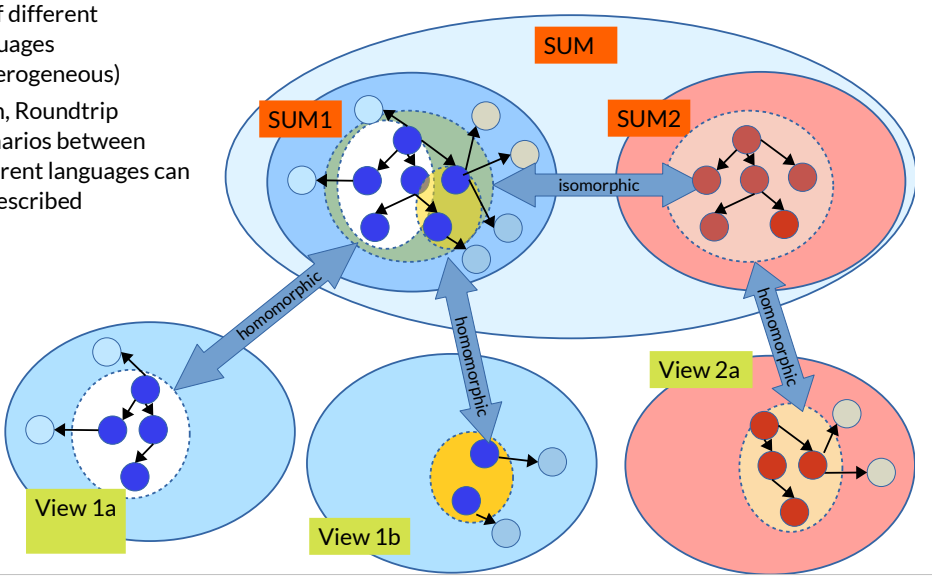


33.8.1 Heterogeneous-Language-Skeleton-SUM

[Seifert11]

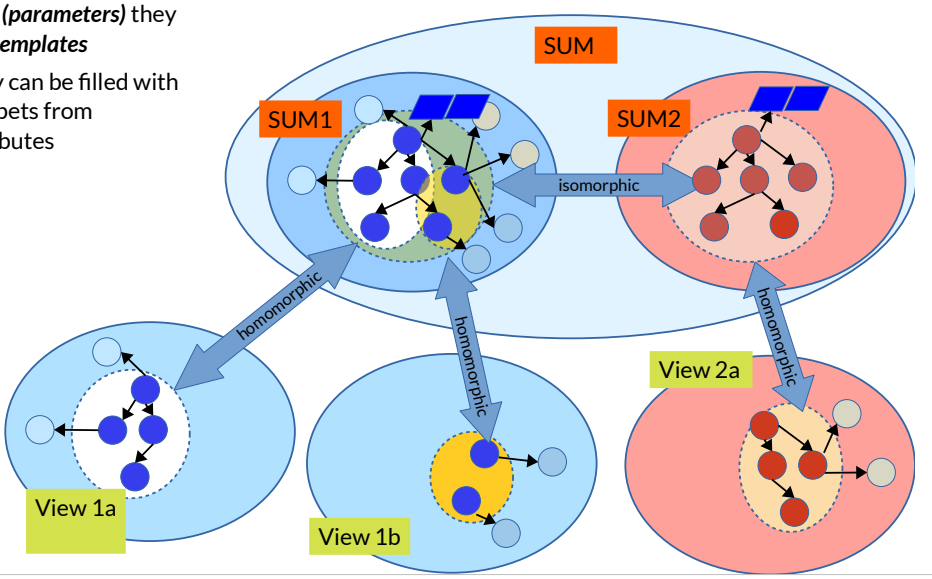
Heterogeneous-Language-Skeleton-SUM

- Disjoing Skeletons can be of different languages (heterogeneous)
- Then, Roundtrip Scenarios between different languages can be described



Heterogeneous-Language-Skeleton-SUM with Templates

- When skeletons have ***slots (parameters)*** they are ***templates***
- They can be filled with snippets from attributes



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?
- What happens if the SUM has several skeletons?
- Which are the contexts of Javadoc-SUM? Why does Javadoc-SUM fulfill the DeltaPutPut rule?
- Which are the contexts of OSM? Why does OSM fulfill the DeltaPutPut rule?
- Why does ROSI-CROM enable Skeleton-SUM?
- Some slides are courtesy to Prof. Colin Atkinson, Mannheim. Used by permission.