

Integrating UML/OCL Derived Properties into Validation and Verification Processes

Frank Hilken¹ Marcel Schuster¹ Karsten Sohr²
Martin Gogolla¹

¹Database Systems Group
University of Bremen, Germany

²Center for Computing Technologies (TZI)
University of Bremen, Germany

OCL Workshop, Saint-Malo, 2016

Introduction

- ▶ Network security becoming more and more of an issue in modern connected world
- ▶ Attacks are revealed more frequently
- ▶ UML/OCL Model for network structures based on global standard (OSI)
- ▶ Abstractions/simplifications using derived properties

Derived Property Example

Function

$$\textit{derived} : p_1 \times \dots \times p_n \rightarrow T$$

Derived Property Example

Function

$$\textit{derived} : p_1 \times \dots \times p_n \rightarrow T$$

association Grandparents **between**

Person [*] **role** gparent derived = **self**.parent.parent→asSet()

Person [*] **role** gchild

end

Derived Property Example

Function

$$\textit{derived}_{gparent} : \text{Person} \rightarrow \text{Set}(\text{Person})$$

association Grandparents **between**

Person [*] **role** gparent derived = **self**.parent.parent→**asSet**()

Person [*] **role** gchild

end

Derived Property Example

Function

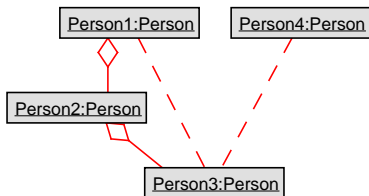
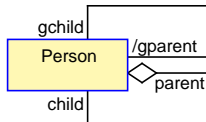
$$\text{derived}_{gparent} : \text{Person} \rightarrow \text{Set}(\text{Person})$$

association Grandparents **between**

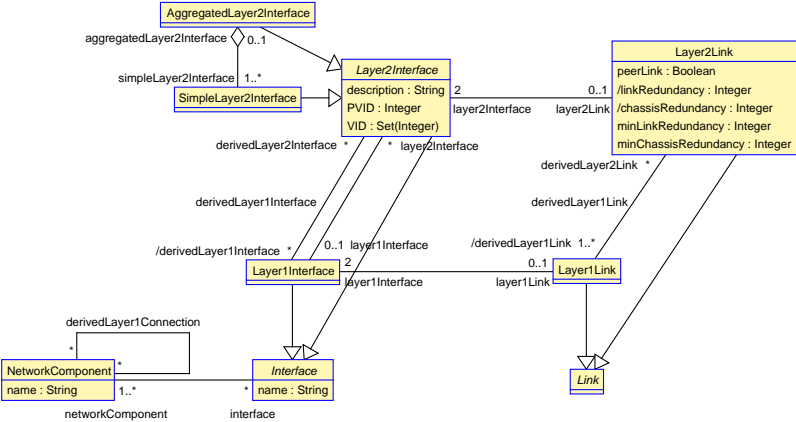
Person [*] **role** gparent derived = **self**.parent.parent→asSet()

Person [*] **role** gchild

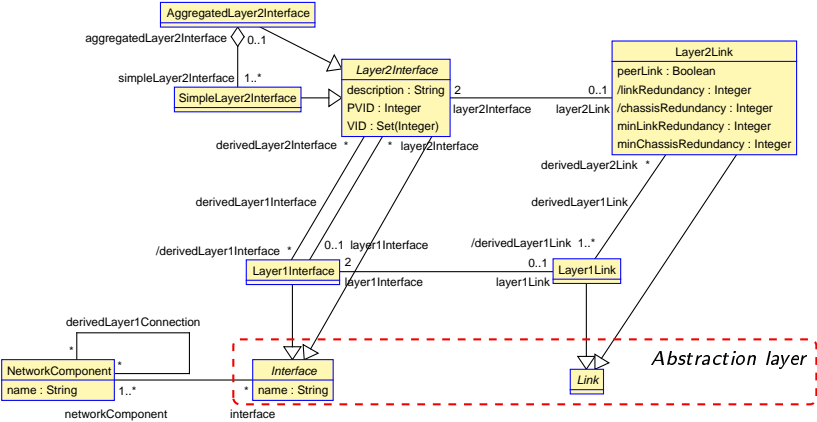
end



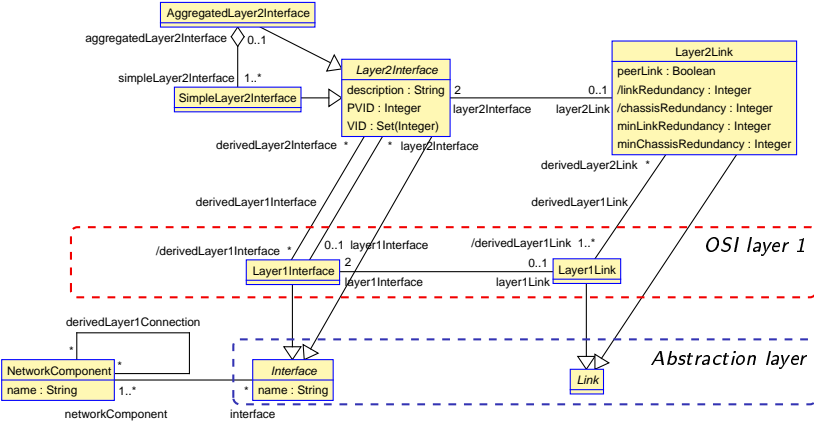
OSI Layer Model



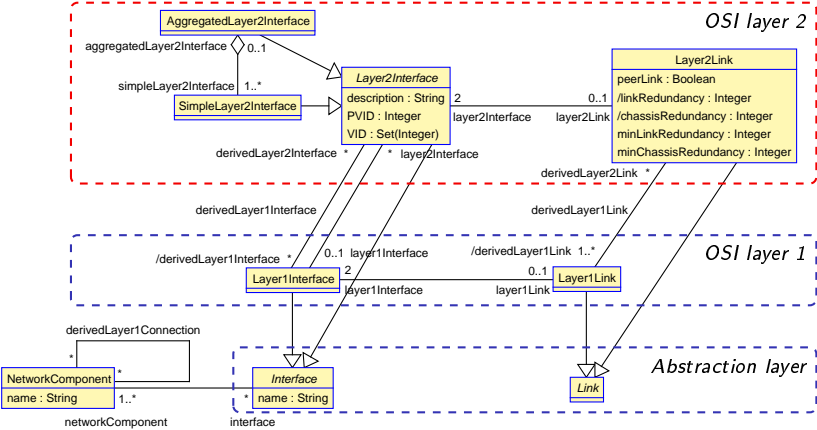
OSI Layer Model



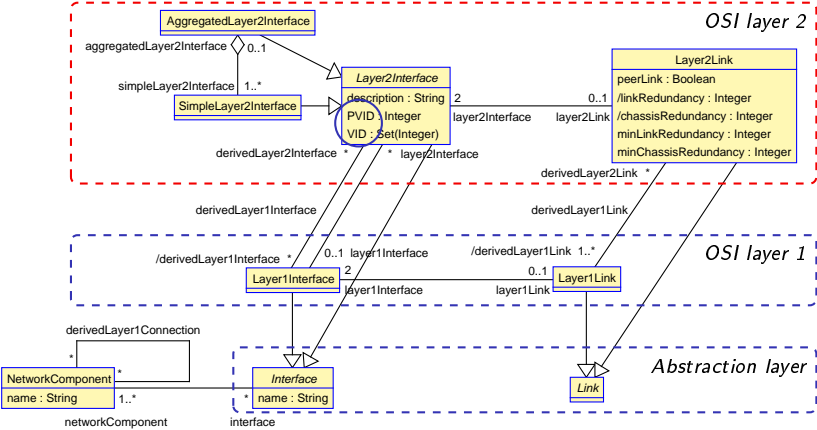
OSI Layer Model



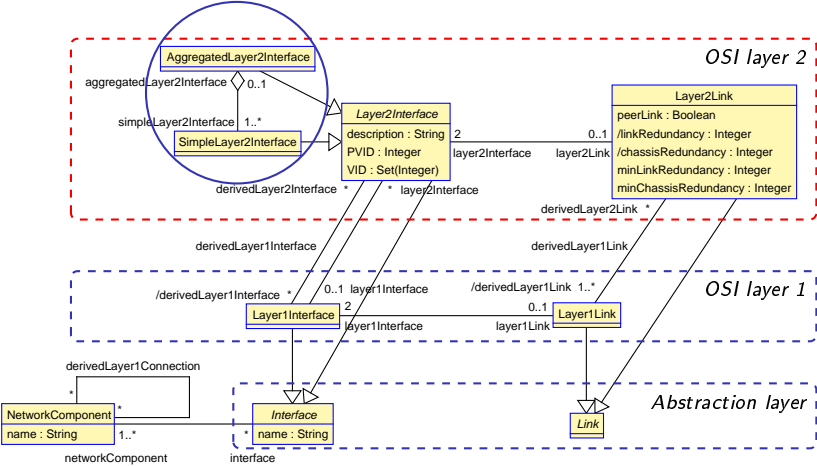
OSI Layer Model



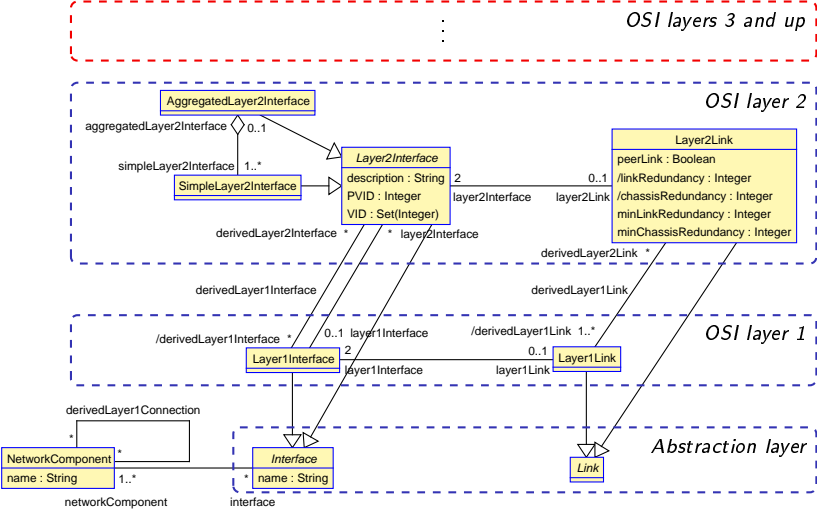
OSI Layer Model



OSI Layer Model



OSI Layer Model



Derived Attribute Examples

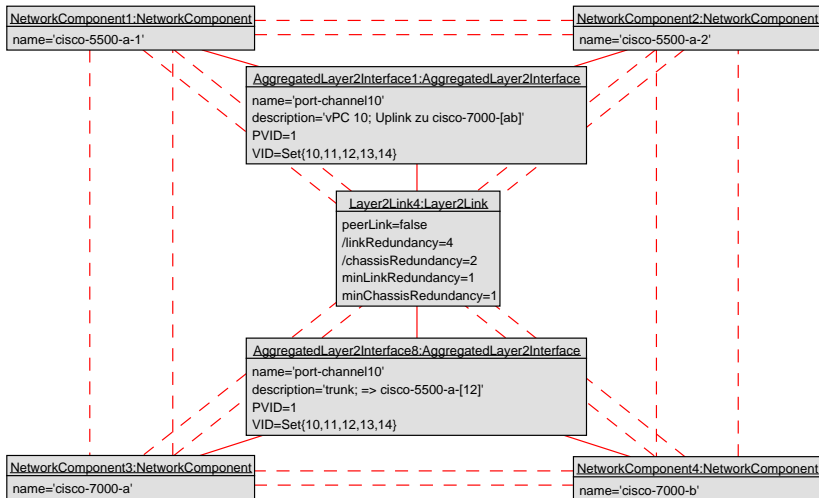
- ▶ **linkRedundancy**: number of parallel, physical links for this connection
- ▶ **chassisRedundancy**: number of parallel, physical machines for this connections

Derived Attribute Definitions

```
linkRedundancy : Integer derived =  
  self.derivedLayer1Link→size()
```

```
chassisRedundancy : Integer derived =  
  self.layer2Interface→collect( i |  
    i.getLayer1Interfaces().networkComponent→asSet()→size()  
  )→min()
```

Derived Expression Example



Use Case I: Checking Existing Network Configurations

The screenshot displays a software analysis tool with three main panels. The top-left panel shows object counts for various classes. The bottom-left panel shows link counts for various associations. The right panel shows a list of class invariants and their satisfaction status.

Class	# Objects
AggregatedLayer2Interface	84
Interface	0
Layer1Interface	1157
Layer1Link	12
Layer2Interface	0
Layer2Link	15
Link	0
NetworkComponent	4
SimpleLayer2Interface	1157

Association	# Links
derivedHasLayer1Connection	12
derivedHasLayer1Interface	146
derivedHasLayer1Link	12
hasInterface	2400
hasLayer1Interface	1157
hasLayer1Link	24
hasLayer2Link	10
hasSimpleLayer2Interface	146

Invariant	Satisfied
AggregatedLayer2Interface::AssociatedLayer1InterfacesAreProhibited	true
AggregatedLayer2Interface::AssociatedLayer1InterfacesAreProperlyConnected	true
AggregatedLayer2Interface::AssociatedNetworkComponentsHaveAPeerLink	true
AggregatedLayer2Interface::AssociatedWithMaxTwoNetworkComponents	true
AggregatedLayer2Interface::SimpleLayer2InterfacesAreAssociatedToSameNet...	true
AggregatedLayer2Interface::SimpleLayer2InterfacesDontHaveLinks	true
AggregatedLayer2Interface::SimpleLayer2InterfacesHaveConsistentVLANConfi...	true
AggregatedLayer2Interface::SimpleLayer2InterfacesHaveDistinctLayer1Interfac...	true
Layer1Interface::AssociatedWithOneNetworkComponent	true
Layer1Link::SelfCommunicationProhibited	true
Layer2Link::AssociatedLayer2InterfacesHaveConsistentVLANConfiguration	true
Layer2Link::MinChassisRedundancyLowerEqualThanChassisRedundancy	true
Layer2Link::MinLinkRedundancyLowerEqualThanLinkRedundancy	true
Layer2Link::SelfCommunicationProhibited	true
NetworkComponent::AllLayer1InterfaceNamesAreUnique	true
NetworkComponent::AllLayer2InterfaceNamesAreUnique	true
NetworkComponent::NetworkComponentNamesGloballyUnique	true
SimpleLayer2Interface::AssociatedLayer1InterfacesAreRequired	true
SimpleLayer2Interface::AssociatedWithOneNetworkComponent	true
SimpleLayer2Interface::Layer1InterfacesAssociatedToSameNetworkComponent	true

Constraints ok. (50ms) 100%

Use Case I: Checking Existing Network Configurations

- ▶ Extraction of system state from network configuration
- ▶ Interactive querying of system state
- ▶ On-the-fly checking of model constraints (invariants, multiplicities, ...)

Use Case I: Checking Existing Network Configurations

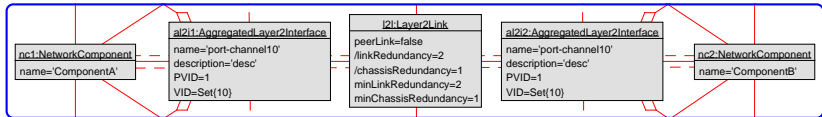
- ▶ Extraction of system state from network configuration
- ▶ Interactive querying of system state
- ▶ On-the-fly checking of model constraints (invariants, multiplicities, ...)
- ▶ roughly 13.400 lines of configuration from multiple files

Excerpt from Network Configuration File

```
interface Ethernet1/24
  description Po1183
  switchport mode trunk
  switchport trunk allowed vlan 10,20,30-40
  channel-group 1183 mode active
```

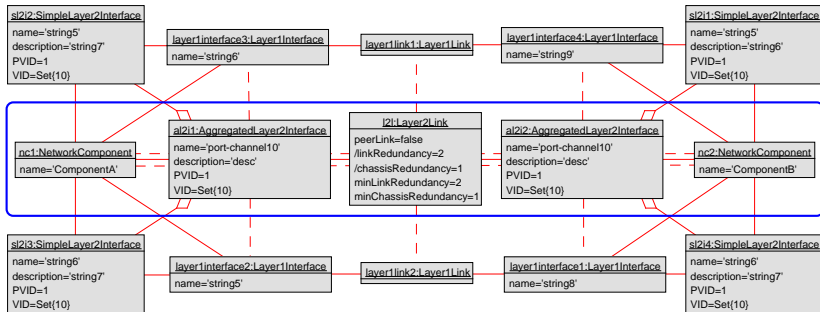
Use Case 2: Generating Network Configurations

- ▶ USE model validator to complete partial system states



Use Case 2: Generating Network Configurations

- ▶ USE model validator to complete partial system states



Transformation into Relational Logic

Derived Attributes

- ▶ Application of derived function on current object (source expression)
- ▶ Not added to the search space (implementation similar to query)
- ▶ No further constraints necessary

Example

OCL Query:

```
self . attribute
```

Translation (application of function):

```
derivedattribute(self)
```

Transformation into Relational Logic

Derived Associations

association AB between

A [2] **role a**

B [1..4] **role b** derived = <OCL expression>

end

- Towards derived role end, the same as derived attribute

Transformation into Relational Logic

Derived Associations

association AB between

A [2] **role** a

B [1..4] **role** b derived = <OCL expression>

end

Reverse Navigation OCL Definition

A.allInstances()→select(a | derived(a)→includes(self))

Relational Logic Formula

$\{a : \text{one } A \mid \text{self} \in \text{derived}(a)\}.$

Transformation into Relational Logic

Derived Associations

association AB between

A [2] **role** a

B [1..4] **role** b **derived** = <OCL expression>

end

Multiplicity Constraints

$$(\text{all } a : \text{one } A \mid \#\text{derived}(a) \geq 1 \wedge \#\text{derived}(a) \leq 4) \wedge$$
$$(\text{all } b : \text{one } B \mid \#\{a : \text{one } A \mid b \in \text{derived}(a)\} = 2)$$

Transformation into Relational Logic

Derived Associations

association AB between

A [2] **role** a

B [1..4] **role** b derived = <OCL expression>

end

Support for n -ary Derived Properties

$$\left\{ a : \text{one } A \mid \underbrace{(\text{some } c : \text{one } C \mid \text{self} \in \text{derived}(a, c))}_{\text{bind additional parameters}} \right\}.$$

Lessons Learned and Future Ideas

- ▶ Tool Support

- ▶ USE implements derived properties as ever evaluated values
- ▶ In order to build partial system states for completion, setting values manually is desired

Lessons Learned and Future Ideas

▶ Tool Support

- ▶ USE implements derived properties as ever evaluated values
- ▶ In order to build partial system states for completion, setting values manually is desired

▶ Derived Classes and Association Classes

- ▶ Non-existent in standards so far
- ▶ Ideas based on derived attributes exist since >10 years
 - ▶ `Typed Set(Tuple())`
- ▶ Classes may only have derived properties attached
- ▶ Similar to Views

Conclusion

- ▶ Using derived properties to employ model constraints
- ▶ Support for derived properties in model checking tool (USE) by transformation into relational logic
- ▶ Network topology model for layers 1 and 2

Conclusion

- ▶ Using derived properties to employ model constraints
- ▶ Support for derived properties in model checking tool (USE) by transformation into relational logic
- ▶ Network topology model for layers 1 and 2

Future Work

- ▶ Implement union and subsets in USE model validator
- ▶ More layers of network topology model (layer 3 = IP including firewall rules)

Conclusion

- ▶ Using derived properties to employ model constraints
- ▶ Support for derived properties in model checking tool (USE) by transformation into relational logic
- ▶ Network topology model for layers 1 and 2
- ▶ USE model validator becomes a very feature rich model checking tool

Future Work

- ▶ Implement union and subsets in USE model validator
- ▶ More layers of network topology model (layer 3 = IP including firewall rules)

Unified Modeling Language (UML)

Class features

- ✓ Class
- ✓ Abstract Class
- ✓ Inheritance
- ✓ Multiple Inheritance
- ✓ Attribute
 - ✓ Derived Value new in this paper
 - ✗ Initial Value
- ✓ Enumeration
- ✓ Invariant

Association features

- ✓ Binary Association
- ✓ N-ary Association
 - Aggregation limited support of cycle freeness (otherwise ✓)
 - Composition limited support of cycle freeness (otherwise ✓)
- ✓ Multiplicity
- ✓ Association Class
- ✓ Derived Association End new in this paper
- ✗ Qualified Association
- ✗ Redefines, Subsets, Union

Operation features

- ✓ Query Operation
 - ✓ Parameter
 - ✓ Return Value
 - ✗ Recursion
- ✗ Operation Call (non query) checking behavior possible via filmstripping
 - ✗ Parameter ⊥ with filmstripping
 - ✗ Return Value ⊥ with filmstripping
 - ✗ Pre-/Postcondition ⊥ with filmstripping
- ✗ Nested Operation Call

✓ supported element – ✗ unsupported element – ○ partially supported element

Object Constraint Language (OCL)

OCL types

✓ Boolean	✓ Integer	✓ Class Type
○ String	○ Real	✗ UnlimitedNatural
✓ Set	✗ Bag	✗ Sequence
✗ OrderedSet	✗ Nested collections	

OCL operations

✓ Comparison Operators	✓ Boolean Operations	✓ Integer Operations
○ String Operations	✗ substring	✗ concat
✓ <Class>.allInstances	✗ <Assoc>.allInstances	✓ size
✓ isEmpty/notEmpty	✓ includes/excludes	✓ including/excluding
✓ forAll/exists	✓ select/reject	✓ one
✓ isUnique	✓ union/intersection	○ any
✓ collect	✓ closure	✗ iterate
✓ toString	○ sum	✓ oclIsType/KindOf
✓ selectByType/Kind	✓ oclAsType	✗ oclType

✓ supported element – ✗ unsupported element – ○ partially supported element

Thanks for your attention!

Frank Hilken

fhilken@informatik.uni-bremen.de