Integrating UML/OCL Derived Properties into Validation and Verification Processes

Frank Hilken[1] Marcel Schuster[1] Karsten Sohr[2]
Martin Gogolla[1]

[1]Database Systems Group
University of Bremen, Germany

[2]Center 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

$$derived : p_1 \times \ldots \times p_n \rightarrow T$$

Derived Property Example

Function

$$derived : p_1 \times \ldots \times p_n \rightarrow T$$

```
association Grandparents between
  Person [*] role gparent   derived = self.parent.parent→asSet()
  Person [*] role gchild
end
```

Derived Property Example

Function

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

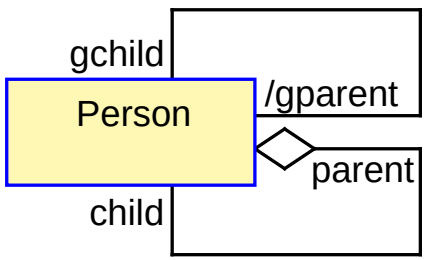
```
association Grandparents between
  Person [*] role gparent   derived = self.parent.parent→asSet()
  Person [*] role gchild
end
```

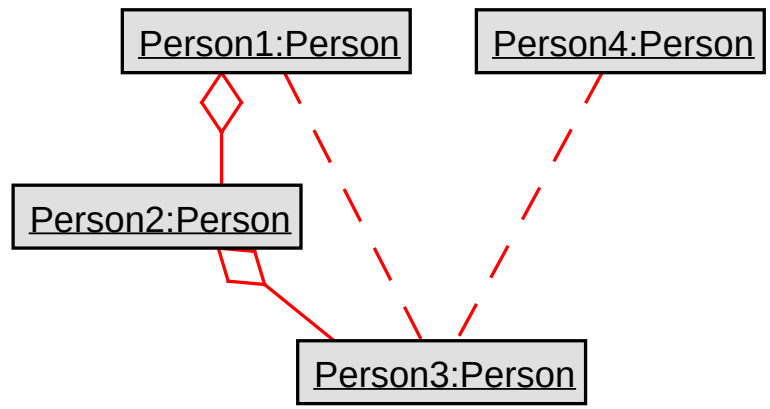
Derived Property Example

Function

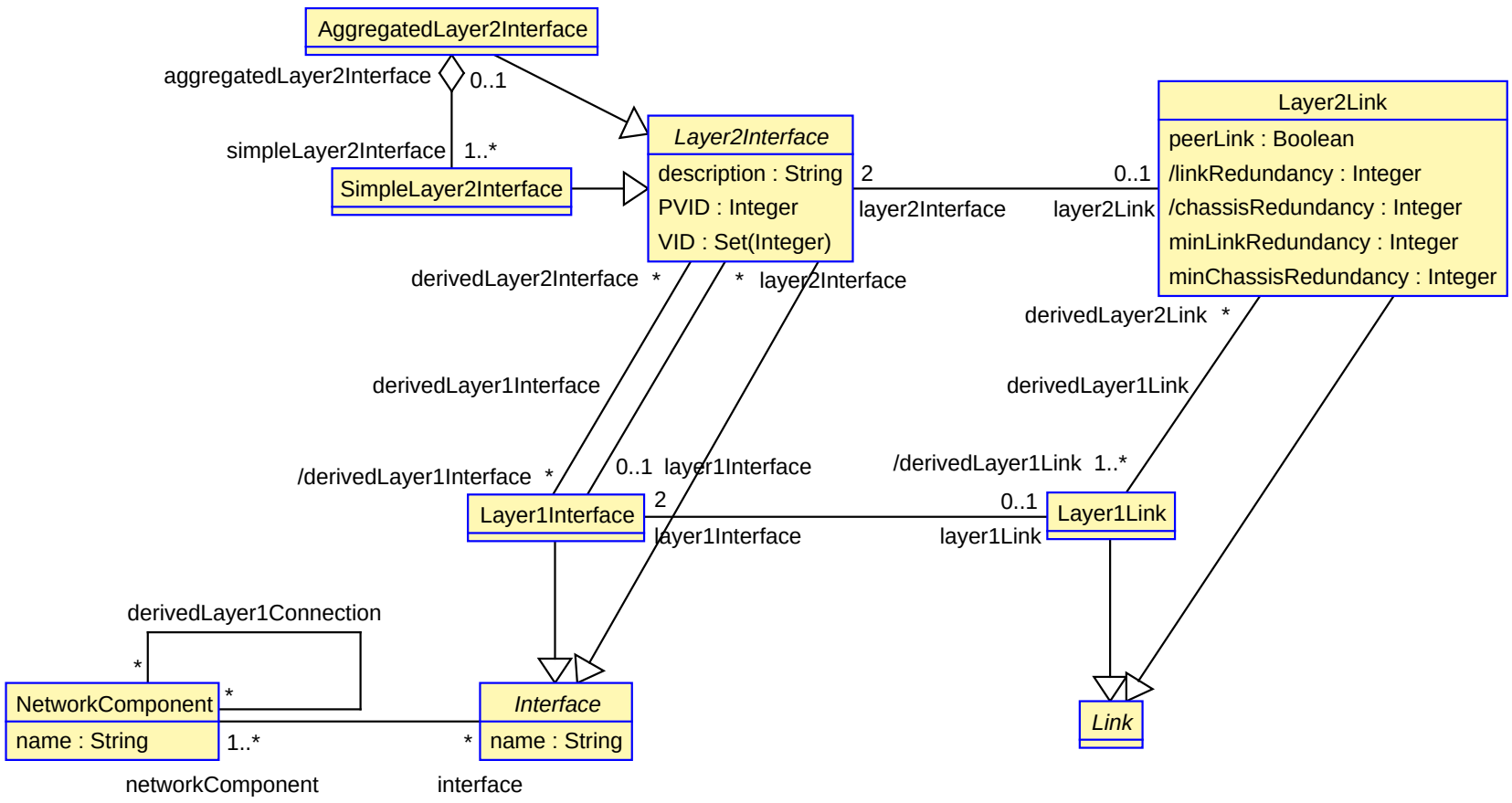
$$derived_{gparent} : \text{Person} \rightarrow \text{Set}(\text{Person})$$

```
association Grandparents between
  Person [*] role gparent   derived = self.parent.parent→asSet()
  Person [*] role gchild
end
```

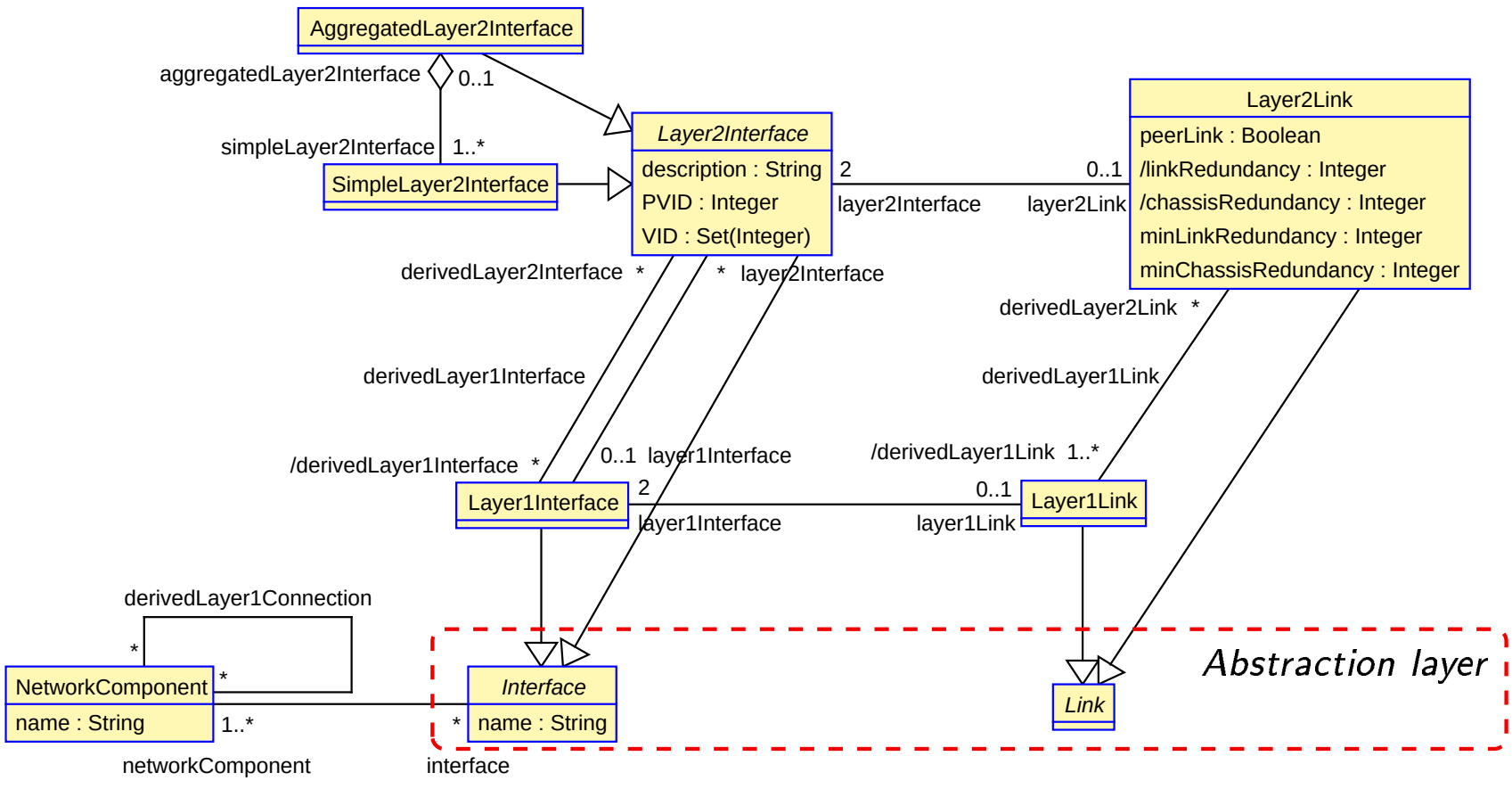




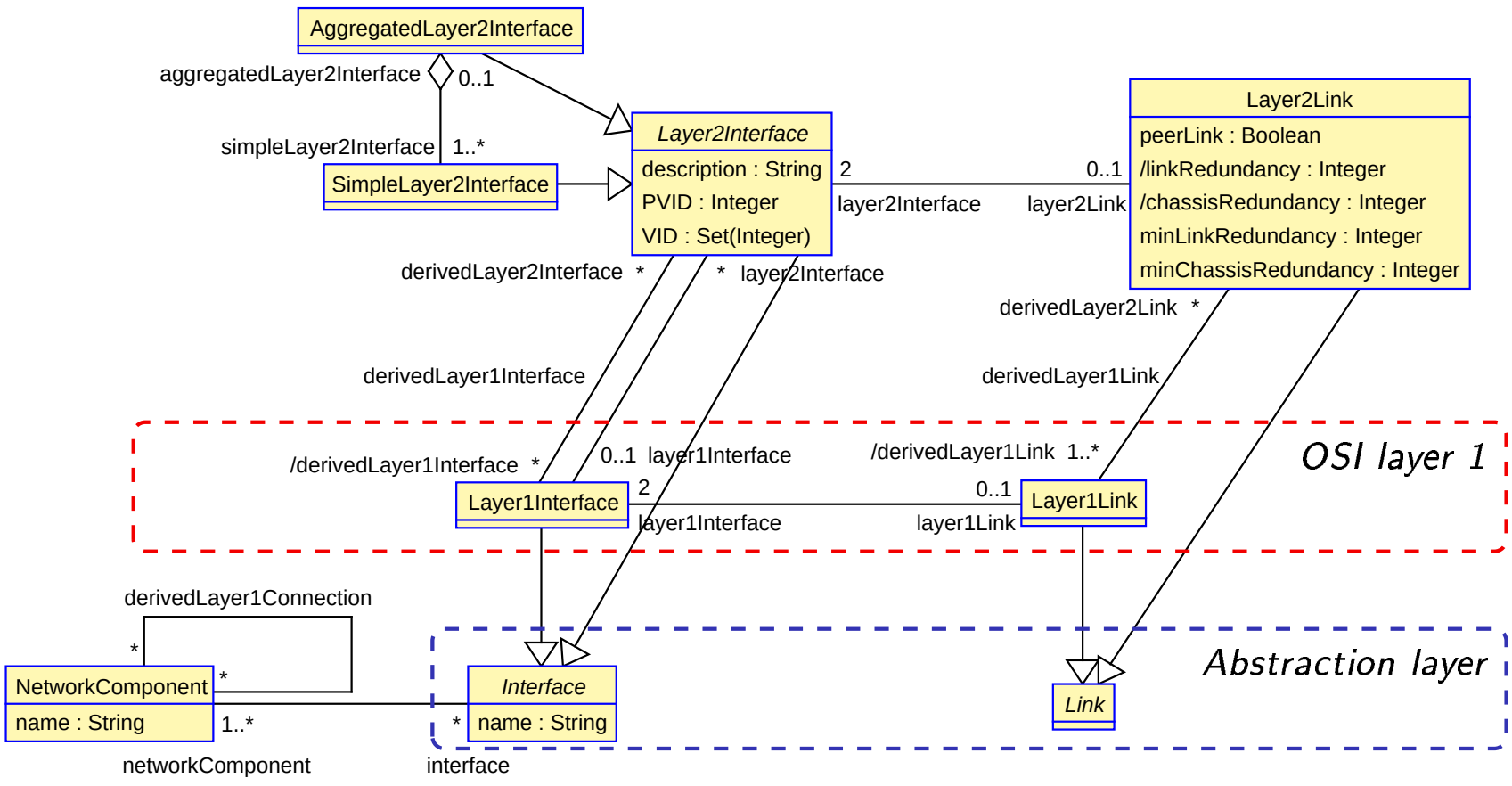
OSI Layer Model



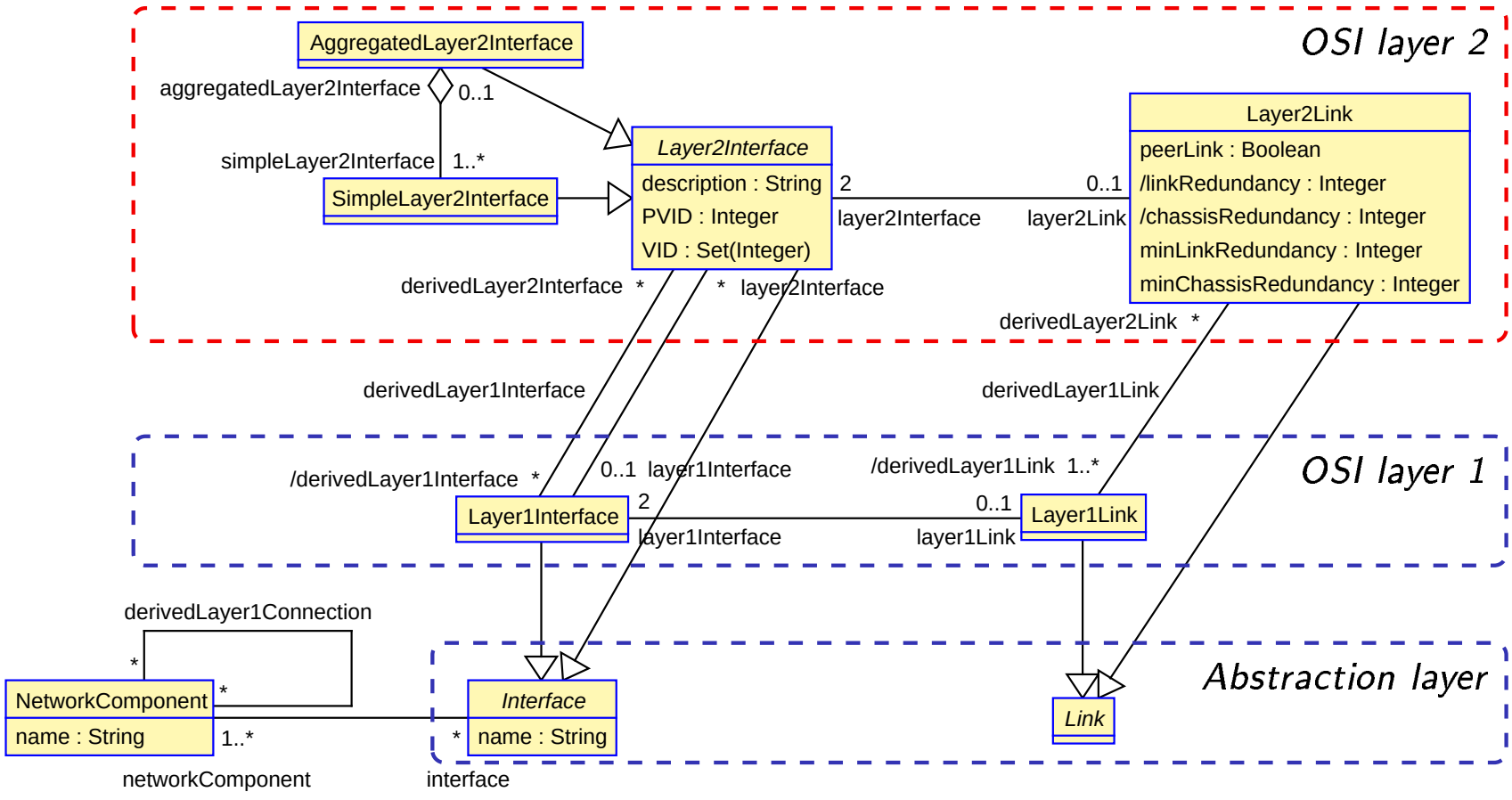
OSI Layer Model



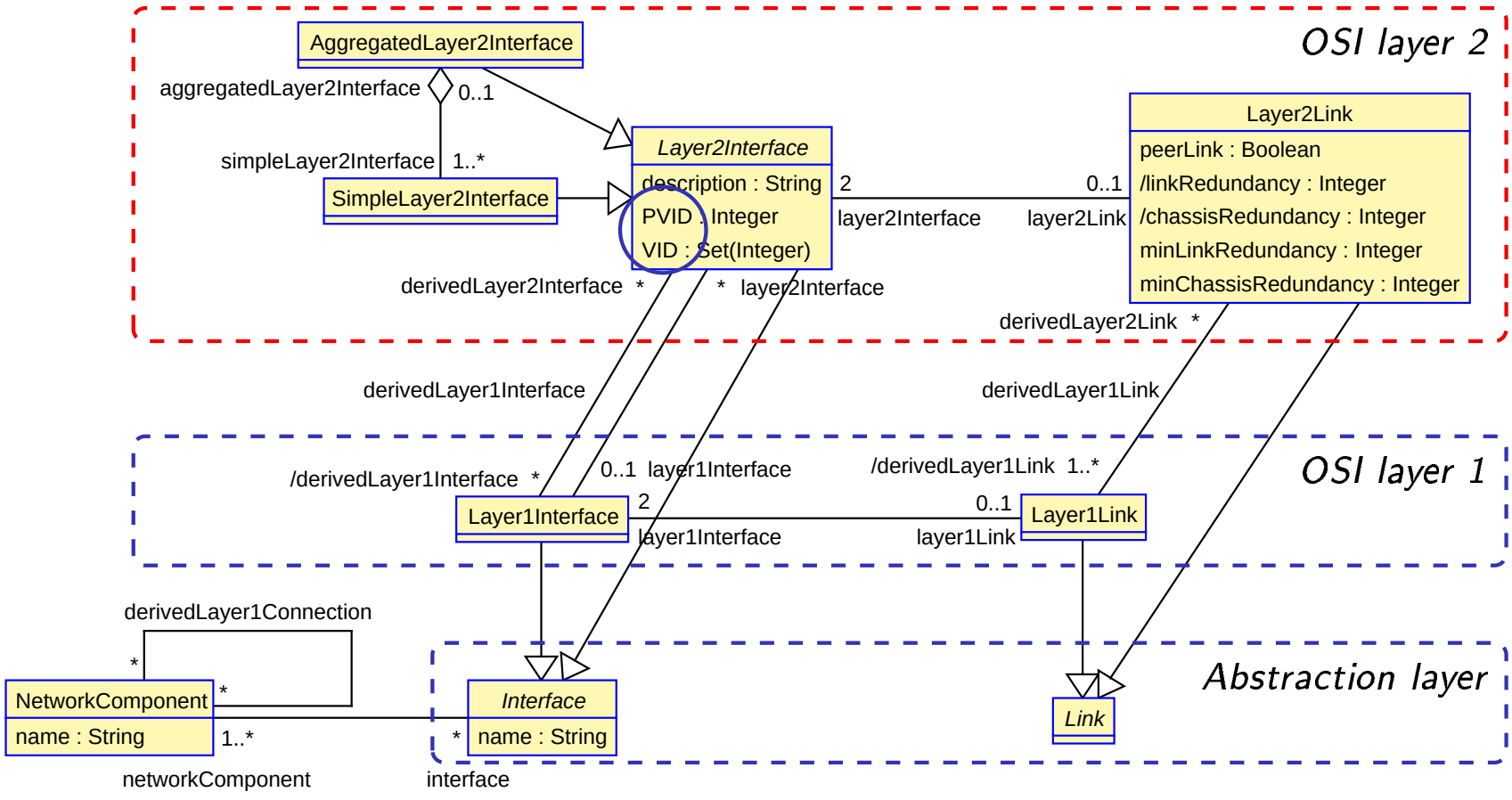
OSI Layer Model



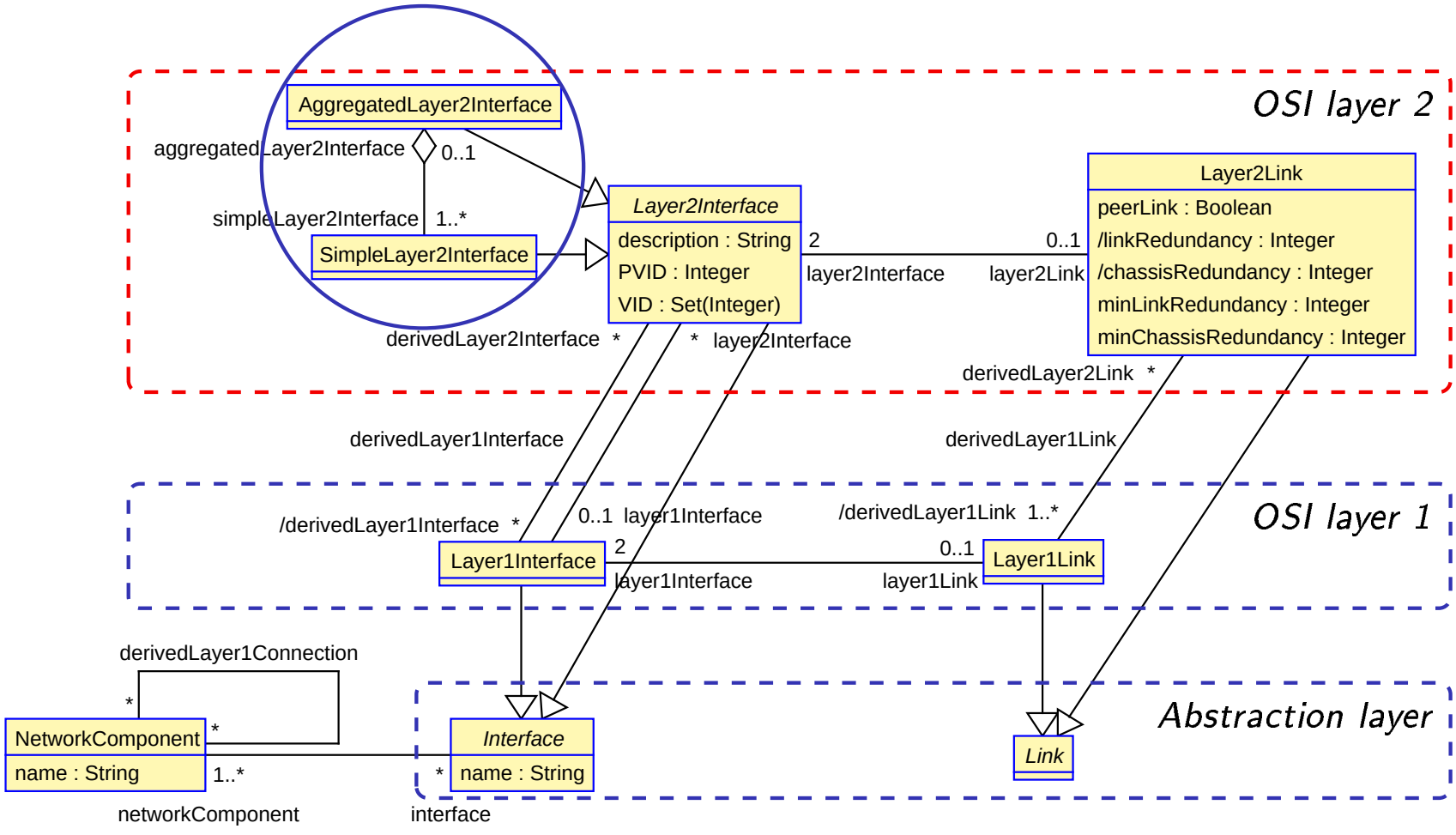
OSI Layer Model



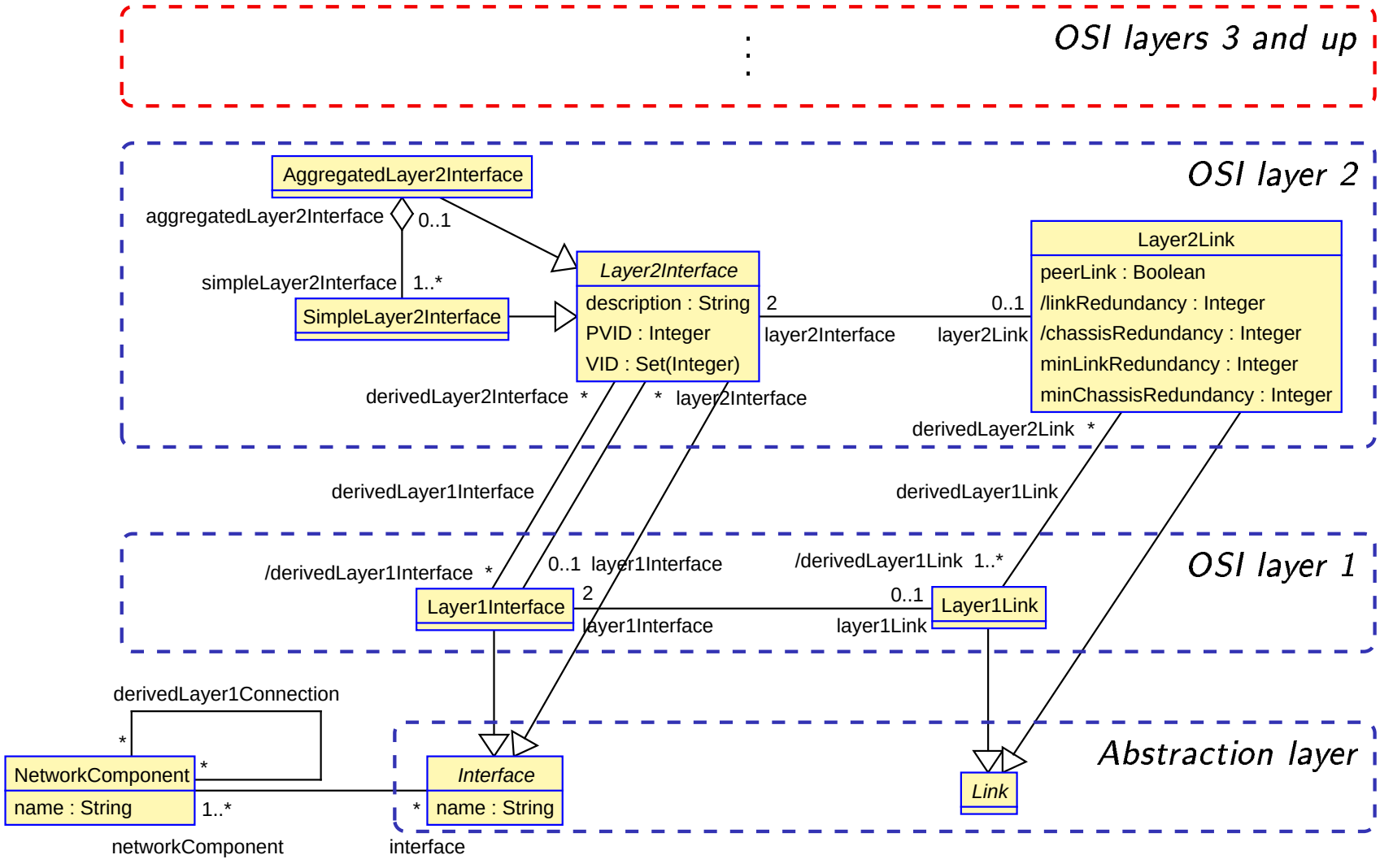
OSI Layer Model



OSI Layer Model



OSI Layer Model



Derived Attribute Examples

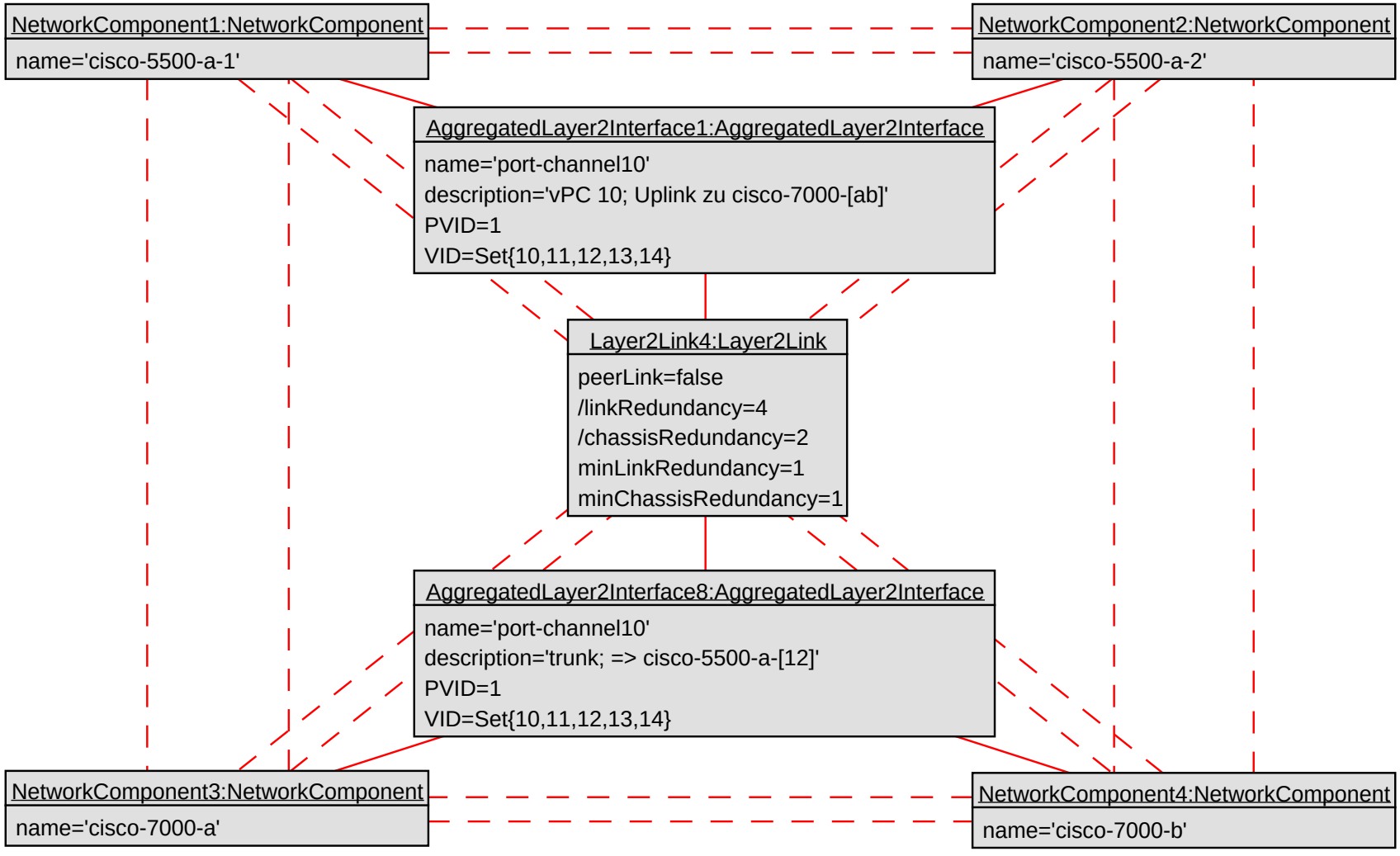
- linkRedundancy: number of parallel, physical links for this connection
- chassisRedundancy: number of parallel, physical machines for this connections

Derived Attribute Definitions

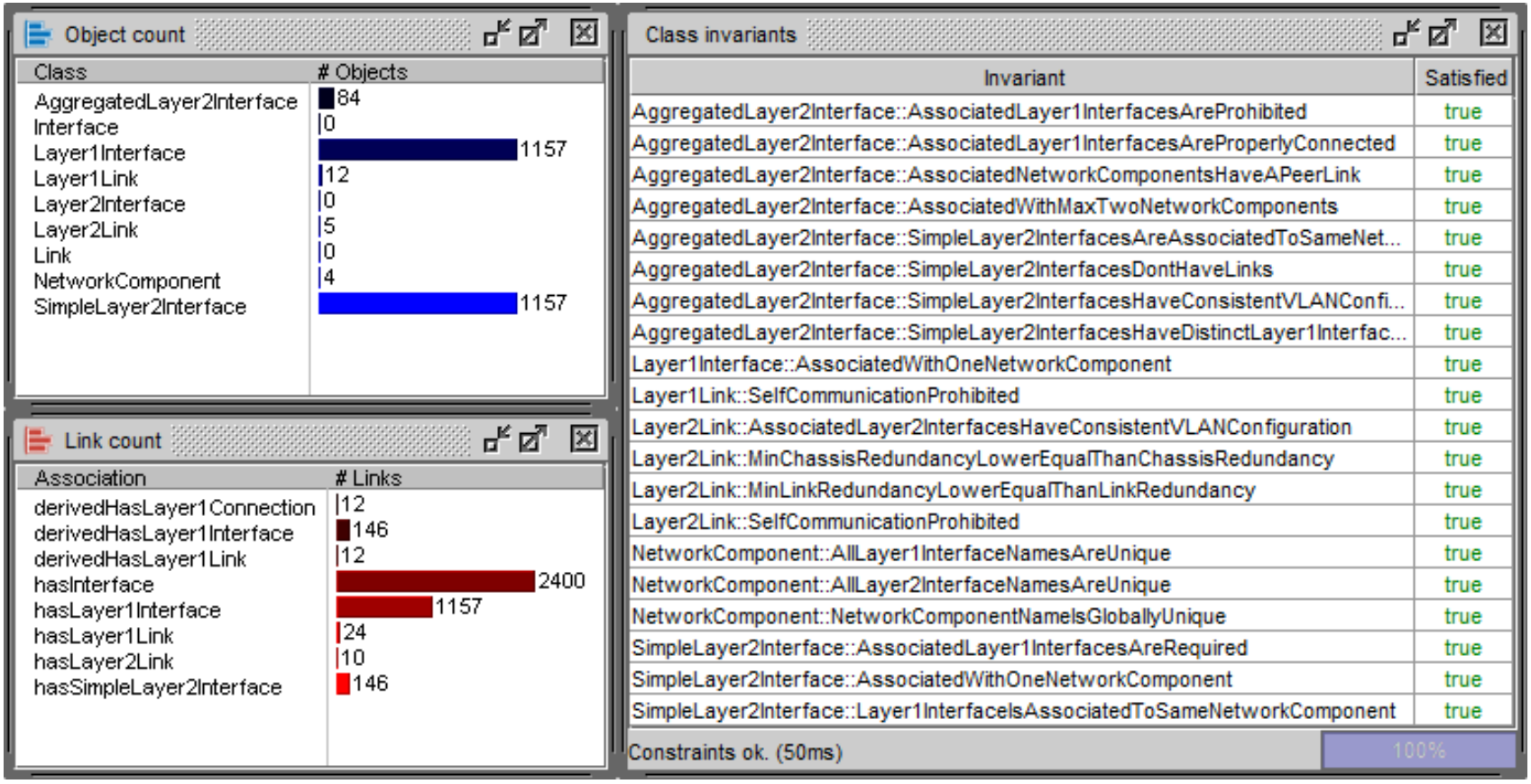
```
linkRedundancy : Integer derived =
  self.derivedLayer1Link→size()

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

Derived Expression Example



Use Case I: Checking Existing Network Configurations



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

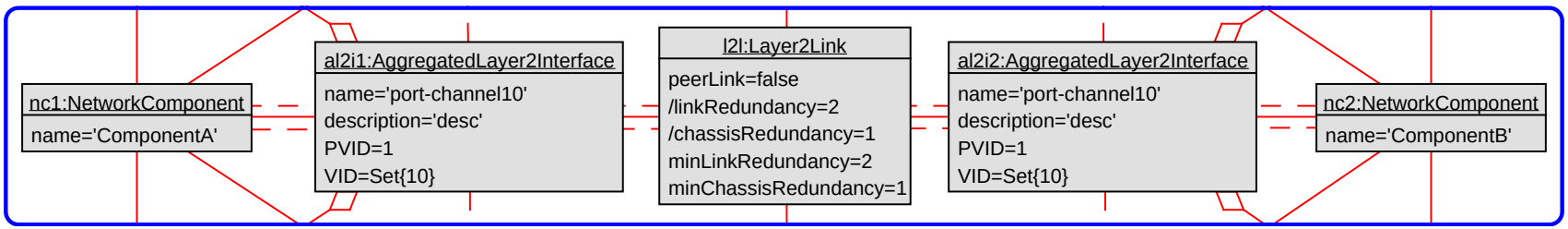
- Extraction of system state from network configuration
- Interactive querying of system state
- On-the-fly checking of model constraints (invariants, multiplicities, . . .)
- roughly 13.400 lines of configuration from multiple files

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
```

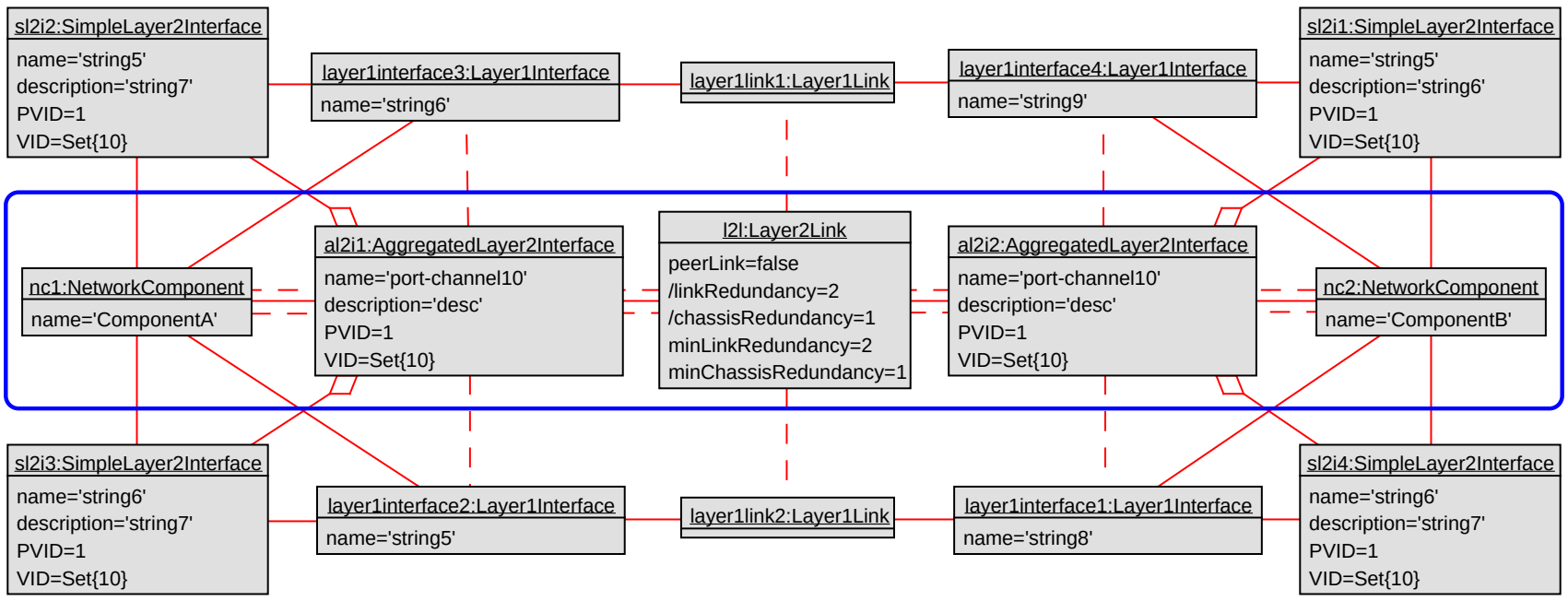
Use Case 2: Generating Network Configurations

- 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):

$derived_{attribute}(\texttt{self})$

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

$$\mathrm{A.allInstances}()\rightarrow\mathrm{select}(\ a\ \mid\ \mathrm{derived}(a)\rightarrow\mathrm{includes}(\ \mathrm{self}\ )\ )$$

Relational Logic Formula

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

Transformation into Relational Logic

Derived Associations

```
association AB between
  A [2]     role a
  B [1..4] role b   derived = <OCL expression>
end
```

Multiplicity Constraints

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

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

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

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	
O Aggregation	limited support of cycle freeness (otherwise ✓)
O 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 – O 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