

Towards Examining the Complexity of Consistency

22nd Workshop on Model Driven Engineering, Verification and Validation in Michigan, USA

Romain Pascual, **Arne Lange**, Thomas Weber, Lars König, Michael Kirsten, Terru Stübinger | October 5, 2025

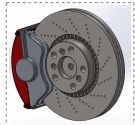


Managing Complexity in System Design

- System design intends to realize complex systems



Managing Complexity in System Design



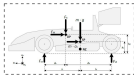
3D Brake Assembly View



Tribological Simulation View



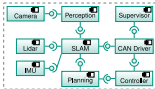
X-in-the-Loop Test Deployment View



Simplified Vehicle Model View



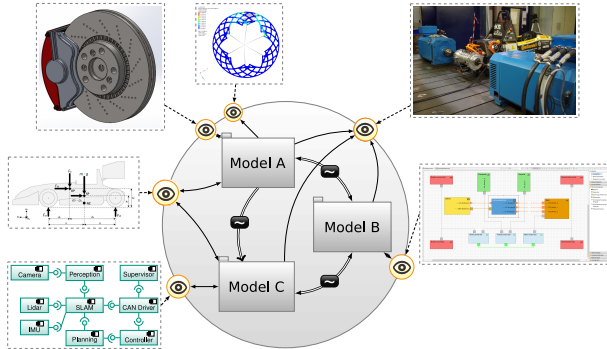
E/E Topology View (PREvision)



Autonomous Driving Components View

- System design intends to realize complex systems
- Details captured by different views
- Each view with a specific concern
- Views only **collectively** fully describe the system

Managing Complexity in System Design



- System design intends to realize complex systems
- Details captured by different views
- Each view with a specific concern
- Views only **collectively** fully describe the system
- Views instantiate models with overlapping relations
- Model **consistency** becomes necessity for **joint realizability**

Convide¹

Consistency in the **View-Based Development** of Cyber-Physical Systems

- Software engineering
- Mechanical engineering
- Electrical engineering
- Formal methods



¹Collaborative Research Centre (CRC) financed by the German Research Foundation (DFG)

Objective

Challenges

- Collaborative development leads to heterogeneous and overlapping models
- Intrinsic complexity of consistency remains unclear
- Consistency is crucial for system correctness and realizability

Claims

- Key features in complexity of consistency need better understanding
- Proper assessment of this complexity requires suitable measures
- Formalization and formal proofs may help to capture the above

Complexity in Modeling

Essential vs. Accidental Complexity

When modeling, we need to distinguish between **essential** and **accidental** complexity, i.e., whether complexity is inherent to the system and domain or if it arises from tools, languages and processes.

Our Focus

- The study of (and propose solutions to mitigate) essential complexity.
- As a starting point, size is a good proxy for complexity, meaning that we can assess complexity via the number of elements, constraints,

Consistency

Assumptions

- Two models are **consistent** if their combined statements do not contradict each other.
- Thus, consistency is a **relation** between multiple **(meta-)model elements** within **two** (meta-)models that might require some **computation** to assess whether the elements are indeed consistent.

Mechanized Proofs in Isabelle/HOL

- Featherweight OCL: closest we have to formal semantics of OCL (proposed for 2.5)
- shallow embedding into Higher-Order Logic (HOL)
- HOL: long established as suitable foundations for formal developments
- Isabelle: well-tested & powerful proof assistant

Specifying Consistency with OCL

Consistency is usually specified using languages like **OCL** or using model transformations.

We focus on the former and extend OCL with coextension operator (\sim):

- returns a Boolean: are two elements in correspondence?
- enables consistency specifications via constraints.

```
context manufacturer::Car
inv: Supplier::Car.allInstances()
  -> select(c|c ~ self)
  -> size() = 1
```

select gather all Car instances in Supplier model that coextend with the current instance (`self`) in the Manufacturer model.

`size() = 1` ensures that exactly one corresponding Car instance exists in the Supplier model

► We obtain a **1-to-1 mapping of the instances**.

From OCL to Isabelle/HOL

Proof skeleton in
Isabelle/Featherweight OCL

```
lemma example_2:
  fixes a :: "Car2"
  shows " $\tau \models \text{One\_to\_One\_Refined}_{inv} a$ "
proof -
  obtain b :: "Car1" where "{b} =
    {x.  $\tau \models \text{Car1} . \text{allInstances}() \rightarrow \text{includes}_{Set}(x)$ 
       $\wedge \tau \models x \sim_{Car} a$ }"
  hence " $\tau \models b \sim a$ "
  hence " $\tau \models b . \text{vin}_{Car1} \triangleq a . \text{vin}_{Car2}$ "
  hence " $\tau \models \text{Car1} . \text{allInstances}()$ 
     $\rightarrow \text{select}_{Set}(c \mid c . \text{vin}_{Car1} \triangleq a . \text{vin}_{Car2})$ 
     $\doteq \text{Set}\{b\}$ "
  moreover have " $\tau \models \text{Set}\{b\} \rightarrow \text{size}_{Set}() \triangleq \mathbf{1}$ "
  ultimately have
    " $\tau \models \text{Car1} . \text{allInstances}()$ 
       $\rightarrow \text{select}_{Set}(c \mid c . \text{vin}_{Car1} \triangleq a . \text{vin}_{Car2})$ 
       $\rightarrow \text{size}_{Set}() \triangleq \mathbf{1}$ "
  thus ?thesis unfolding One_to_One_Refined_inv_def by simp
qed
```

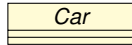
Dimensions of Consistency Complexity

- **Arity of the mapping:** How many (meta-)model elements are involved in each side?
- **Complexity of the computation:** How complex is the computation needed to assess the consistency?
- **Compositional complexity:** Can the consistency specification be broken down into simpler ones?

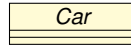
Our idea is to link the **complexity of the consistency** specification to the **complexity of providing a formal proof** that the given metamodels are consistent.

A 1-to-1 nominal consistency

Car manufacturer



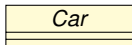
Supplier



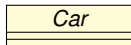
```
context Car manufacturer::Car
inv: Supplier::Car.allInstances()
  -> select(c|c ~ self)
  -> size() = 1
```

A 1-to-1 nominal consistency

Car manufacturer



Supplier

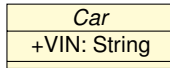


```
context Car manufacturer::Car
inv: Supplier::Car.allInstances()
  -> select(c|c ~ self)
  -> size() = 1
```

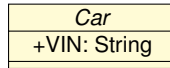
```
definition One_to_One_inv :: "Car2  $\Rightarrow$  Boolean" where
  "One_to_One_inv (self)  $\equiv$  Car1
  .allInstances()->select_Set(c | c ~ self)
  ->size_Set()  $\triangleq$  1"
```

A 1-to-1 structural consistency specification

Car manufacturer

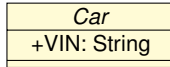


Supplier

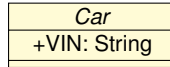


A 1-to-1 structural consistency specification

Car manufacturer



Supplier



```
context Car manufacturer::Car
inv: Supplier::Car.allInstances()
  -> select(c|c.VIN = self.VIN)
  -> size() = 1
```

A 1-to-1 structural consistency specification

```
context Car manufacturer::Car
inv: Supplier::Car.allInstances()
    -> select(c|c.VIN = self.VIN)
    -> size() = 1
```

A 1-to-1 structural consistency specification

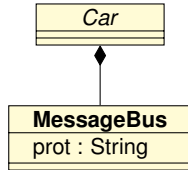
```
context Car manufacturer::Car
inv: Supplier::Car.allInstances()
    -> select(c|c.VIN = self.VIN)
    -> size() = 1
```

definition *One_to_One_Refined_{inv}* :: "Car2 \Rightarrow
Boolean" **where**

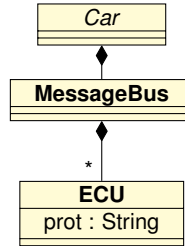
```
"One_to_One_Refinedinv (self)  $\equiv$   
  Car1 .allInstances()  
  ->selectSet(c | c .vinCar1  $\triangleq$  self .vinCar2)  
  ->sizeSet()  $\triangleq$  1"
```

An explicit MessageBus with 1-to- n consistency

Car manufacturer

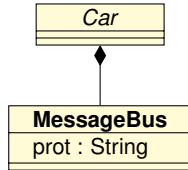


Supplier

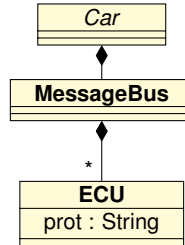


An explicit MessageBus with 1-to- n consistency

Car manufacturer



Supplier



```
context Car manufacturer::MessageBus
inv: let mb:Supplier::MessageBus
    = self.select(~) in mb.ecu
    -> forAll(e|e.prot = self.prot)
```

A 1-to- n consistency specification

```
context Car manufacturer::MessageBus
inv: let mb:Supplier::MessageBus
     = self.select(~) in mb.ecu
     -> forAll(e|e.prot = self.prot)
```

A 1-to- n consistency specification

```
context Car manufacturer::MessageBus
inv: let mb:Supplier::MessageBus
     = self.select(~) in mb.ecu
     -> forAll(e|e.prot = self.prot)
```

definition $One_to_N_{inv} :: "MessageBus1 \Rightarrow Boolean"$ **where**

```
"One_to_Ninv (self)  $\equiv$ 
  let mb = MessageBus2 .allInstances()
      ->selectSet(m| self ~ m)
      ->asSequenceSet()->firstSeq()
  in (mb .ecuMessageBus2 ->forAllSet(e|
    e .protECU2  $\triangleq$  self .protMessageBus1))"
```

Discussion

Besides rigor, formalization provides a way to analyze complexity of consistency via the complexity of its specification's formal proof.

Proof complexity in Isabelle/HOL reflects the **structural and logical complexity** of consistency specifications:

- Arity and aggregation influence proof obligations.
- Computational complexity appears both in definitions and proofs.

Current limitations

Manual Encoding – translating UML/OCL to Isabelle/HOL is largely manual, which is time-consuming and error-prone. Additional tooling might help.

Scalability – we experimented on simplified models. Real-world systems may introduce additional complexity.

Conclusion

- Coextension operator (\sim) checks the correspondence on the object level \rightarrow extension to standard OCL

Conclusion

- Coextension operator (\sim) checks the correspondence on the object level \rightarrow extension to standard OCL
- We related consistency specification to OCL-like expressions, which are translated into Isabelle/HOL proof obligations (proof of concept)

Conclusion

- Coextension operator (\sim) checks the correspondence on the object level \rightarrow extension to standard OCL
- We related consistency specification to OCL-like expressions, which are translated into Isabelle/HOL proof obligations (proof of concept)
- We showed that we can decompose complex proof obligations into simpler and more manageable parts.

Conclusion

- Coextension operator (\sim) checks the correspondence on the object level \rightarrow extension to standard OCL
- We related consistency specification to OCL-like expressions, which are translated into Isabelle/HOL proof obligations (proof of concept)
- We showed that we can decompose complex proof obligations into simpler and more manageable parts.
- Next step is to automate the transformation from OCL to Isabelle.

Conclusion

- Coextension operator (\sim) checks the correspondence on the object level \rightarrow extension to standard OCL
- We related consistency specification to OCL-like expressions, which are translated into Isabelle/HOL proof obligations (proof of concept)
- We showed that we can decompose complex proof obligations into simpler and more manageable parts.
- Next step is to automate the transformation from OCL to Isabelle.
- First step in, hopefully, many to analyze the complexity of consistency in view-based modeling systems.