Querying Automotive System Models and Safety Artifacts with MMINT and Viatra

Alessio Di Sandro, Sahar Kokaly, Rick Salay, Marsha Chechik {adisandro, skokaly, rsalay, chechik}@cs.toronto.edu

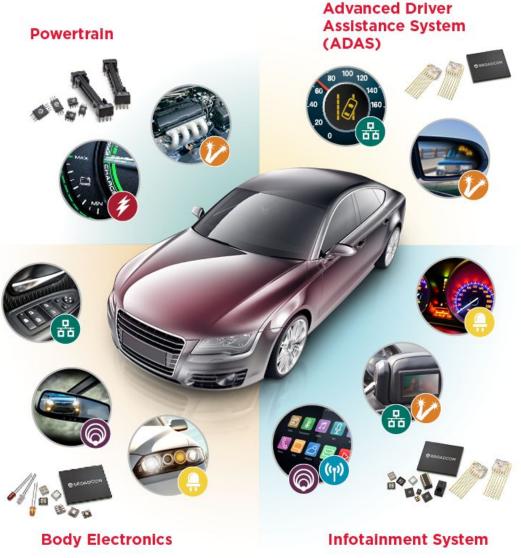
University of Toronto

MASE, Sep 15 2019, Munich, Germany



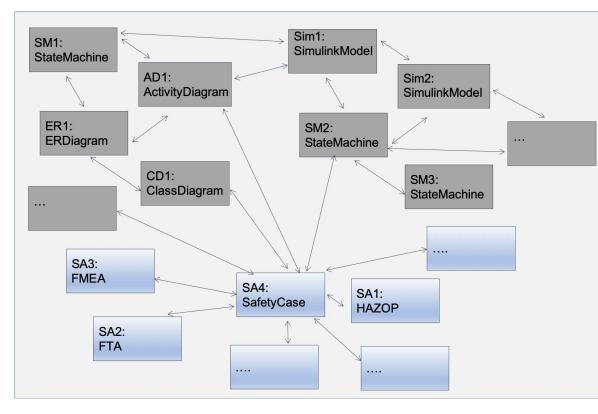
Automotive domain complexity

- Increasing number of interconnected electronic and software components
- ISO 26262 functional safety standard: analyze hazards and provide evidence that the system being designed is **safe**



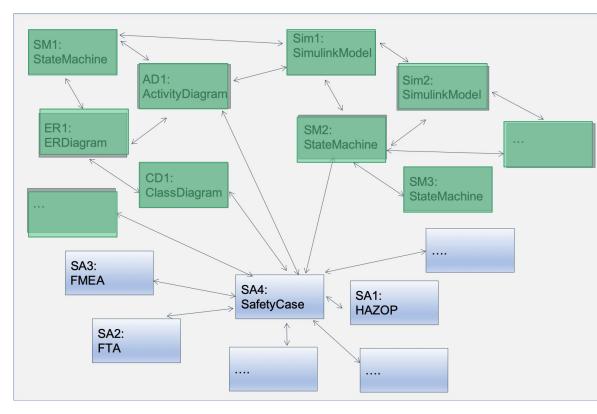
Automotive models

- Taming the domain complexity with models
 - heterogeneous
 - large
 - interconnected



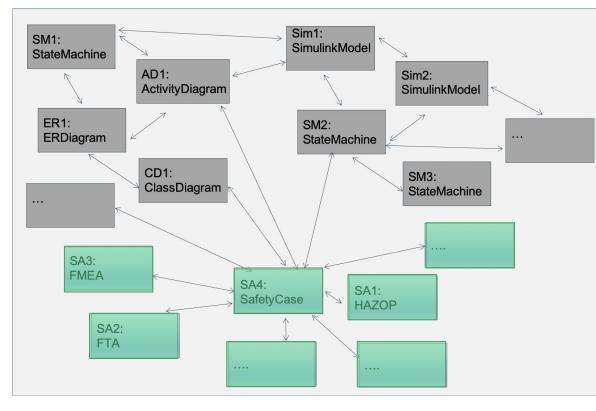
Automotive models

- Taming the domain complexity with models
 - heterogeneous
 - large
 - interconnected
- System models
 - SM, AD, ER, CD,
 Simulink



Automotive models

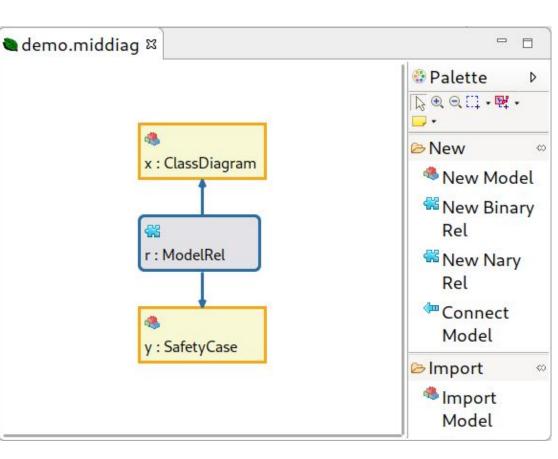
- Taming the domain complexity with models
 - heterogeneous
 - large
 - interconnected
- System models
 - UML models,
 Simulink models,
 etc.
- ISO 26262 safety artifacts
 - FMEA, FTA, HAZOP, Safety Case, etc.



MMINT

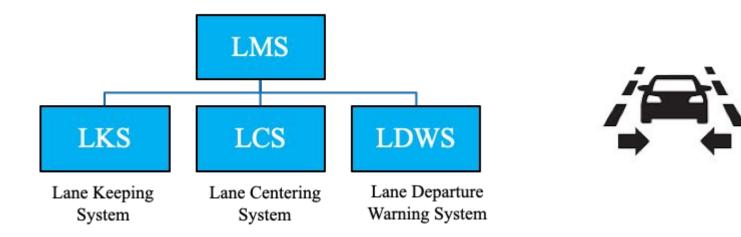
- Interactive
 framework for model
 management using
 Eclipse EMF
- Megamodels: collection of models connected by relationships
- Megamodel editor
 - create/import models and relationships
 - invoke operations



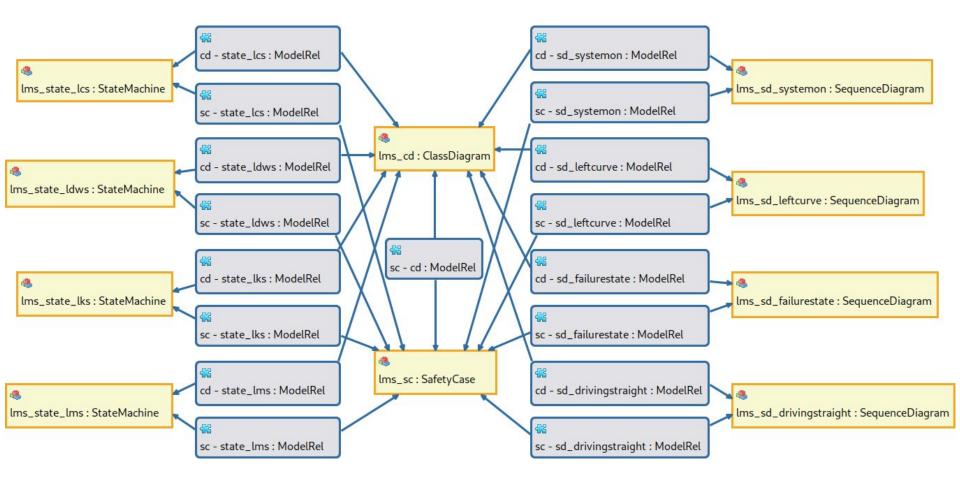


Lane Management System (LMS)

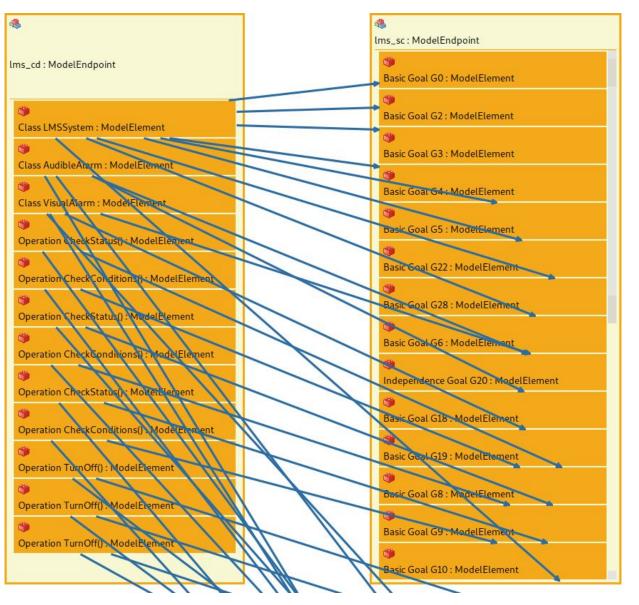
- Driver assistance system to keep the vehicle within a lane
- Takes control of braking and steering
- Safety critical, subject to the ISO 26262 standard



LMS megamodel



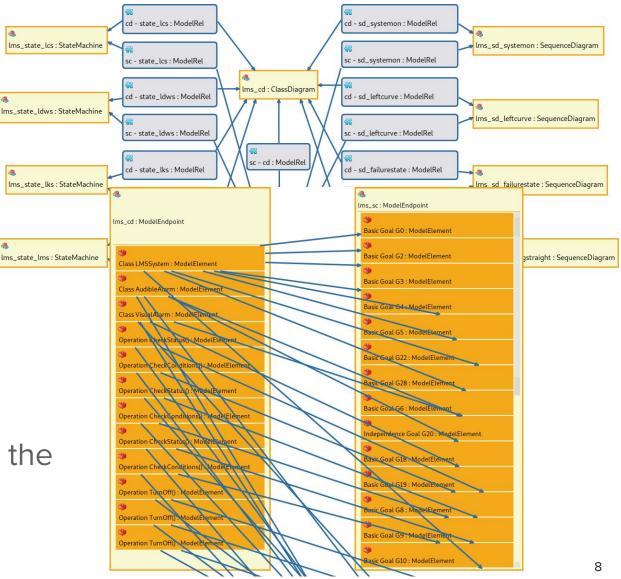
LMS megamodel



Extracting info from megamodels

- Megamodels can easily grow in size
- Like databases, they contain organized data (models and relationships)

Need a way to **query** the information required!



Query engine requirements

Generic

- 1. Navigation inter-model and intra-model
- 2. Handle heterogeneous models in the same query
- 3. Get a particular result or all results from a query
- 4. Select query inputs and display results in a megamodel
- 5. Scale with big models

Implementation-specific

- 1. Integration with Eclipse EMF
- 2. APIs to programmatically load and invoke queries

OCL

- OMG standard
- Default query and constraint language in Eclipse EMF
- Declarative
 syntax, functions
 with inputs and
 outputs, explicit
 collection of
 results

```
1 import 'http://se.cs.toronto.edu/mmint/MID'
 2 import 'http://se.cs.toronto.edu/modelepedia/SafetyCase'
 4⊖ context mid::ModelElement
 6@def: connectedModelElems : OrderedSet(ModelElement) =
     let mid = self.oclContainer().oclContainer().oclAsType(MID) in
     let rels = mid.models->select(rel |
 80
       rel.oclIsTypeOf(ModelRel))->collect(oclAsType(ModelRel)) in
 9
100
     let modelElems = rels->collect(mappings)
11
                          ->collect(modelElemEndpoints)
12
                          ->collect(target.oclAsType(ModelElement))
13
                           ->asOrderedSet() in
149
     modelElems->select(modelElemTgt |
15⊝
       self <> modelElemTqt and
160
       rels->exists(rel |
17⊖
         rel.mappings->exists(mapping |
           mapping.modelElemEndpoints->collect(target)
189
19
                                      ->includesAll(Set{self, modelElemTqt}))))
20
21⊖ context mid::MID
22
23@def: connectedModelElems1 : Set(Tuple(src : ModelElement,
24
                                          tgt : ModelElement)) =
250
     let modelElems = ModelElement.allInstances() in
26
     modelElems->collect(e | Set{e}->product(connectedModelElems2(e)))->asSet()
27
28@def: connectedModelElems2(modelElemSrc : ModelElement) : Set(ModelElement) =
     modelElemSrc.connectedModelElems->asSet()
29
30
31@def: connectedModelElems3(modelElemSrc : ModelElement,
32
                             modelElemTgt : ModelElement) : Boolean =
33
     connectedModelElems2(modelElemSrc)->includes(modelElemTgt)
34
35@def: allConnectedModelElems1 : Set(Tuple(src : ModelElement,
36
                                             tgt : ModelElement)) =
370
     let modelElems = ModelElement.allInstances() in
38
     modelElems->collect(e | Set{e}->product(allConnectedModelElems2(e)))->asSet()
39
40⊖ def: allConnectedModelElems2(modelElemSrc : ModelElement) : Set(ModelElement) =
41
     modelElemSrc->closure(connectedModelElems)->excluding(modelElemSrc)
42
43⊖def: allConnectedModelElems3(modelElemSrc : ModelElement,
44
                                modelElemTgt : ModelElement) : Boolean =
45
     allConnectedModelElems2(modelElemSrc)->includes(modelElemTqt)
```

https://www.eclipse.org/ocl

Viatra

- Incremental query engine based on the Rete algorithm
- Graph pattern
 based language
 (VQL)

```
1 package library
 2
 3⊖ import "http://se.cs.toronto.edu/mmint/MID"
   import "http://se.cs.toronto.edu/mmint/MID/Relationship"
 5
 6⊖ pattern connectedModelElems(modelElemSrc: ModelElement,
                                modelElemTqt: ModelElement) {
 70
 8
     modelElemSrc != modelElemTgt;
 9
     Model.modelElems(modelSrc, modelElemSrc);
     Model.modelElems(modelTgt, modelElemTgt);
10
11
     modelSrc != modelTqt;
12
     Mapping.modelElemEndpoints.target(mapping, modelElemSrc);
13
     Mapping.modelElemEndpoints.target(mapping, modelElemTgt);
14 }
15
16<sup>o</sup> pattern allConnectedModelElems(modelElemSrc: ModelElement,
170
                                   modelElemTgt: ModelElement) {
18
     modelElemSrc != modelElemTat;
19
     Model.modelElems(modelSrc, modelElemSrc);
20
     Model.modelElems(modelTgt, modelElemTgt);
21
     modelSrc != modelTqt;
     find connectedModelElems+(modelElemSrc, modelElemTgt);
22
23 }
```

• Prolog-like, pattern arguments can be used as inputs or outputs, implicit collection of results

https://www.eclipse.org/viatra

OCL VQL Generic

1. Navigation inter-model and intra-model

OCL VQL Generic

- 1. Navigation inter-model and intra-model
- 2. Handle heterogeneous models in the same query

OCL VQL Generic

- 1. Navigation inter-model and intra-model
- 2. Handle heterogeneous models in the same query
- 3. Get a particular result or all results from a query

```
1 import 'http://se.cs.toronto.edu/mmint/MID'
                                                                              1 package library
2 import 'http://se.cs.toronto.edu/modelepedia/SafetyCase'
                                                                              2
3
                                                                              3⊖ import "http://se.cs.toronto.edu/mmint/MID"
4⊖ context mid::ModelElement
5
                                                                              4 import "http://se.cs.toronto.edu/mmint/MID/Relationship"
6@def: connectedModelElems : OrderedSet(ModelElement) =
7 let mid = self.oclContainer().oclContainer().oclAsType(MID) in
                                                                              6 pattern connectedModelElems(modelElemSrc: ModelElement,
80
    let rels = mid.models->select(rel |
9
      rel.oclIsTypeOf(ModelRel))->collect(oclAsType(ModelRel)) in
                                                                              70
    let modelElems = rels->collect(mappings)
100
                                                                              8
                                                                                   modelElemSrc != modelElemTat;
11
                        ->collect(modelElemEndpoints)
                                                                              9
                                                                                   Model.modelElems(modelSrc, modelElemSrc);
12
                        ->collect(target.oclAsType(ModelElement))
13
                        ->asOrderedSet() in
                                                                             10
                                                                                   Model.modelElems(modelTgt, modelElemTgt);
140
    modelElems->select(modelElemTgt |
                                                                             11
                                                                                   modelSrc != modelTqt;
15⊖
      self <> modelElemTgt and
                                                                             12
                                                                                   Mapping.modelElemEndpoints.target(mapping, modelElemSrc);
160
      rels->exists(rel |
17⊖
        rel.mappings->exists(mapping
                                                                             13
                                                                                   Mapping.modelElemEndpoints.target(mapping, modelElemTgt);
18⊖
          mapping.modelElemEndpoints->collect(target)
                                                                             14 }
19
                                  ->includesAll(Set{self, modelElemTgt}))))
                                                                             15
20
                                                                             16 pattern allConnectedModelElems(modelElemSrc: ModelElement,
21@ context mid::MID
22
                                                                             170
23@def: connectedModelElems1 : Set(Tuple(src : ModelElement,
                                                                             18
                                                                                   modelElemSrc != modelElemTqt;
24
                                      tat : ModelElement)) =
25⊖ let modelElems = ModelElement.allInstances() in
                                                                             19
                                                                                   Model.modelElems(modelSrc, modelElemSrc);
26
    modelElems->collect(e | Set{e}->product(connectedModelElems2(e)))->asSet()
                                                                             20
                                                                                   Model.modelElems(modelTgt, modelElemTgt);
27
                                                                             21
                                                                                   modelSrc != modelTat;
28@def: connectedModelElems2(modelElemSrc : ModelElement) : Set(ModelElement) =
29
    modelElemSrc.connectedModelElems->asSet()
                                                                             22
                                                                                   find connectedModelElems+(modelElemSrc, modelElemTqt);
30
                                                                             23 }
310 def: connectedModelElems3(modelElemSrc : ModelElement,
32
                           modelElemTqt : ModelElement) : Boolean =
33
    connectedModelElems2(modelElemSrc)->includes(modelElemTqt)
34
35@def: allConnectedModelElems1 : Set(Tuple(src : ModelElement,
36
                                         tgt : ModelElement)) =
37⊖
    let modelElems = ModelElement.allInstances() in
38
    modelElems->collect(e | Set{e}->product(allConnectedModelElems2(e)))->asSet()
39
40⊖ def: allConnectedModelElems2(modelElemSrc : ModelElement) : Set(ModelElement) =
41
    modelElemSrc->closure(connectedModelElems)->excluding(modelElemSrc)
42
43⊕ def: allConnectedModelElems3(modelElemSrc : ModelElement,
44
                             modelElemTqt : ModelElement) : Boolean =
45
    allConnectedModelElems2(modelElemSrc)->includes(modelElemTgt)
```

modelElemTgt: ModelElement) {

modelElemTgt: ModelElement) {

```
1 import 'http://se.cs.toronto.edu/mmint/MID'
2 import 'http://se.cs.toronto.edu/modelepedia/SafetyCase'
3
4⊖ context mid::ModelElement
5
6@def: connectedModelElems : OrderedSet(ModelElement) =
7 let mid = self.oclContainer().oclContainer().oclAsType(MID) in
    let rels = mid.models->select(rel |
80
9
       rel.oclIsTypeOf(ModelRel))->collect(oclAsType(ModelRel)) in
100
     let modelElems = rels->collect(mappings)
11
                          ->collect(modelElemEndpoints)
12
                          ->collect(target.oclAsType(ModelElement))
13
                          ->asOrderedSet() in
140
     modelElems->select(modelElemTgt |
15⊖
       self <> modelElemTgt and
160
       rels->exists(rel |
17⊖
         rel.mappings->exists(mapping)
18⊖
           mapping.modelElemEndpoints->collect(target)
19
                                     ->includesAll(Set{self, modelElemTgt}))))
20
21⊖ context mid::MID
22
23@def: connectedModelElems1 : Set(Tuple(src : ModelElement,
24
                                          tat : ModelElement)) =
25⊖
    let modelElems = ModelElement.allInstances() in
26
     modelElems->collect(e | Set{e}->product(connectedModelElems2(e)))->asSet()
27
28@def: connectedModelElems2(modelElemSrc : ModelElement) : Set(ModelElement) =
29
     modelElemSrc.connectedModelElems->asSet()
30
31⊖def: connectedModelElems3(modelElemSrc : ModelElement,
32
                             modelElemTgt : ModelElement) : Boolean =
33
     connectedModelElems2(modelElemSrc)->includes(modelElemTqt)
34
35@def: allConnectedModelElems1 : Set(Tuple(src : ModelElement,
                                             tgt : ModelElement)) =
36
37⊖
    let modelElems = ModelElement.allInstances() in
38
     modelElems->collect(e | Set{e}->product(allConnectedModelElems2(e)))->asSet()
39
40⊖ def: allConnectedModelElems2(modelElemSrc : ModelElement) : Set(ModelElement) =
41
     modelElemSrc->closure(connectedModelElems)->excluding(modelElemSrc)
42
43⊕ def: allConnectedModelElems3(modelElemSrc : ModelElement,
44
                                modelElemTqt : ModelElement) : Boolean =
45
     allConnectedModelElems2(modelElemSrc)->includes(modelElemTgt)
```

```
1 package library
 2
 3⊖ import "http://se.cs.toronto.edu/mmint/MID"
 4 import "http://se.cs.toronto.edu/mmint/MID/Relationship"
 5
 6 pattern connectedModelElems(modelElemSrc: ModelElement.
 70
                                modelElemTgt: ModelElement) {
 8
     modelElemSrc != modelElemTat;
 9
     Model.modelElems(modelSrc, modelElemSrc);
10
     Model.modelElems(modelTgt, modelElemTgt);
11
     modelSrc != modelTqt;
12
     Mapping.modelElemEndpoints.target(mapping, modelElemSrc);
     Mapping.modelElemEndpoints.target(mapping, modelElemTgt);
13
14 }
15
16 pattern allConnectedModelElems(modelElemSrc: ModelElement,
170
                                   modelElemTgt: ModelElement) {
18
     modelElemSrc != modelElemTqt;
19
     Model.modelElems(modelSrc, modelElemSrc);
20
     Model.modelElems(modelTgt, modelElemTgt);
21
     modelSrc != modelTat;
22
     find connectedModelElems+(modelElemSrc, modelElemTqt);
23 }
```

```
    OCL requires multiple queries
to achieve the same flexibility
of a single VQL query
```

OCL VQL Generic

(sep)

- 1. Navigation inter-model and intra-model
- 2. Handle heterogeneous models in the same query
- 3. Get a particular result or all results from a query
- 4. Select query inputs and display results in a megamodel

OCL VQL Generic

- (sep) ★ [1,2]
- 1. Navigation inter-model and intra-model
- 2. Handle heterogeneous models in the same query
- 3. Get a particular result or all results from a query
- 4. Select query inputs and display results in a megamodel

14

5. Scale with big models

 G. Bergmann, Á. Horváth, I. Ráth, D. Varró, A. Balogh, Z. Balogh, and A. Ökrös, "Incremental evaluation of model queries over EMF models", MODELS 2010, Oslo, Norway, October 3-8, 2010
 Z. Ujhelyi, G. Szoke, Á. Horváth, N. I. Csiszár, L. Vidács, D. Varró, and R. Ferenc, "Performance comparison of query-based techniques for anti-pattern detection", Information & Software Technology, vol. 65, pp. 147–165, 2015

OCL VQL Generic

(sep) V

X [1,2]

- 1. Navigation inter-model and intra-model
- 2. Handle heterogeneous models in the same query
- 3. Get a particular result or all results from a query
- 4. Select query inputs and display results in a megamodel
 - 5. Scale with big models

Implementation-specific

- 1. Integration with Eclipse EMF
 - 2. APIs to programmatically load and invoke queries

 G. Bergmann, Á. Horváth, I. Ráth, D. Varró, A. Balogh, Z. Balogh, and A. Ökrös, "Incremental evaluation of model queries over EMF models", MODELS 2010, Oslo, Norway, October 3-8, 2010
 Z. Ujhelyi, G. Szoke, Á. Horváth, N. I. Csiszár, L. Vidács, D. Varró, and R. Ferenc, "Performance comparison of query-based techniques for anti-pattern detection", Information & Software Technology, vol. 65, pp. 147–165, 2015

Viatra integration in MMINT

- Query Abstraction
 Layer (QAL)
 programming
 interface
 - a. select query inputs graphically
 - b. select query
 - c. dispatch
 query+inputs to
 specific engine
 - return query
 results as EMF
 objects

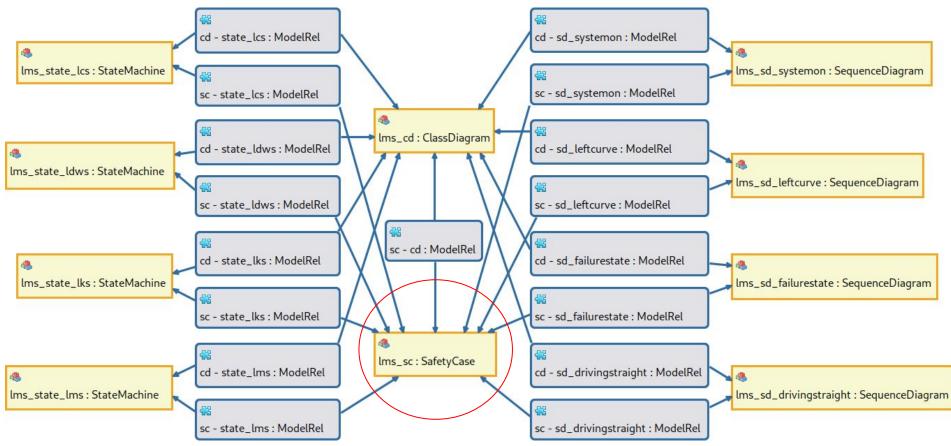
```
/**
  Evaluates a query to find elements within a megamodel.
  @param queryFilePath
            The path to the query file.
  @param gueryName
            The name of the query to be evaluated
            (a query file can contain multiple queries).
  @param context
            The context where the query is executed,
            i.e. a megamodel, or one of its contained elements.
  @param gueryArgs
            The actual arguments to the guery.
 * @return A list of megamodel elements that match the query.
public default List<Object> evaluateQuery(String queryFilePath,
                                          String queryName,
                                          EObject context,
                                          List<? extends EObject> queryArgs) {
```

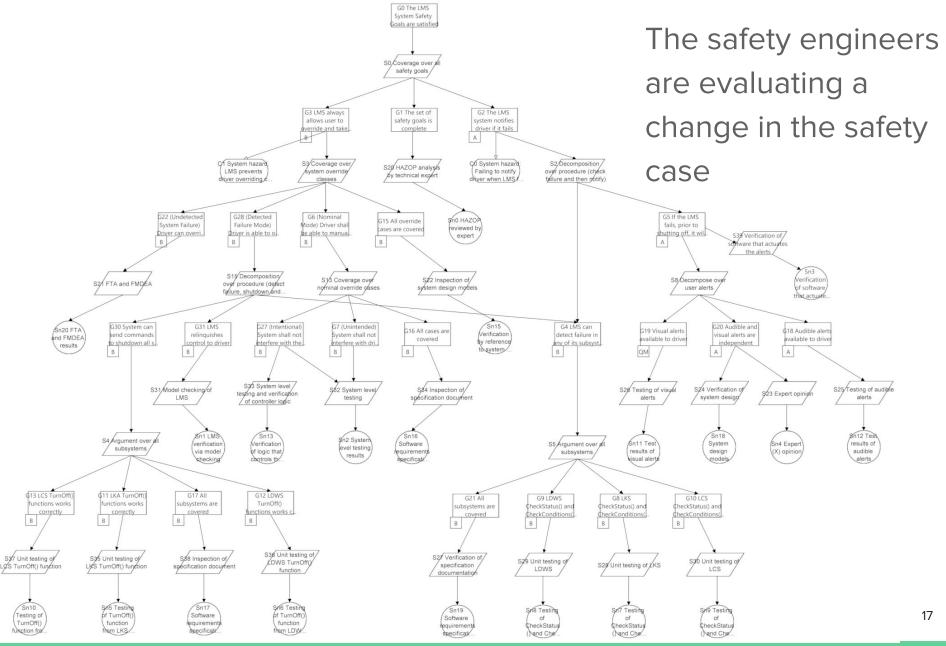
```
return List.of();
```

}

- Viatra QAL implementation
- VQL library
 - \circ extract megamodel navigation
 - users can focus on the automotive questions

The safety engineers are evaluating a change in the safety case



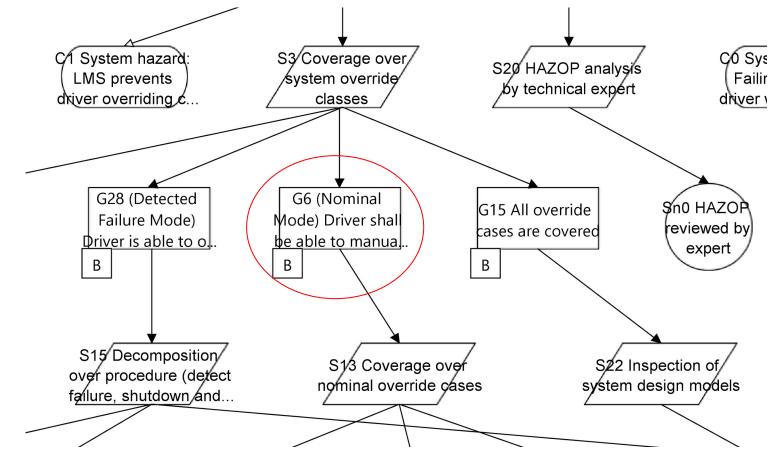


Safety case for LMS:

- Uses Goal Structured Notation (GSN)
- Structured argument that the LMS is safe to operate, supported by evidence
- Top level goal gets decomposed into solution leaves



The safety engineers are evaluating a change to the Goal G6 in the safety case



Querying the LMS megamodel

connectedModelElems

• Which system elements are directly connected to G6?

```
CV-System hazard: []
   prevents driver
                                                                         ailing to notify driver
                                                 SZO HAZOP analysis b
                        53 Overage over system
  overriding control of
                                                                         hen LMS fails. Vehicle
                          override classes
                                                   technical expert
 steering. Vehicle hazard
                                                                         hazard: Unintended
                                                                        operation of vehicle fea
 Vehicle feature preven
                         G6 (Nominal
     G28 (Detected Failure
                                    Navigate
                                                               .
     lode) Driver is able to
                         Driver shall be
                                                                   Sn0 HAZOP
     override the system
                         manually over
                                                                   reviewed by
     pon detecting failure
                         system at ar
                                    Edit
                                                                    expert
                                    Show/Hide
                                    Layout
                                    Format
    S15 Decomposition over
                             513
                                    Profiles
                                                               ion of system
    procedure (detect failure
     utdown and relinguist
                                                                 models
         control)
                                    Show EClass information
                                    ➡ Show References
                                    OCL
             G27 (Intentional
             System shall not
                                    MMINT
                                                                    Evaluate Querv
 uishes
             interfere with the
 iver
    package library
 1
 2
 3⊖ import "http://se.cs.toronto.edu/mmint/MID"
    import "http://se.cs.toronto.edu/mmint/MID/Relationship"
 4
 5
 6 pattern connectedModelElems(modelElemSrc: ModelElement,
 70
                                          modelElemTat: ModelElement) {
 8
       modelElemSrc != modelElemTat;
 9
       Model.modelElems(modelSrc, modelElemSrc);
       Model.modelElems(modelTat, modelElemTat);
10
       modelSrc != modelTat;
11
       Mapping.modelElemEndpoints.target(mapping, modelElemSrc);
12
       Mapping.modelElemEndpoints.target(mapping, modelElemTgt);
13
14 }
15
16@pattern allConnectedModelElems(modelElemSrc: ModelElement,
170
                                              modelElemTgt: ModelElement) {
18
       modelElemSrc != modelElemTat:
      Model.modelElems(modelSrc, modelElemSrc);
19
20
       Model.modelElems(modelTat, modelElemTat);
       modelSrc != modelTgt;
21
       find connectedModelElems+(modelElemSrc, modelElemTgt);
22
```

23 }

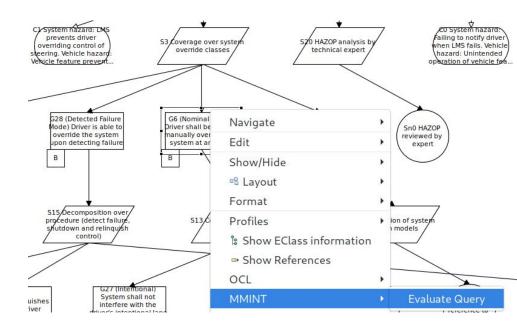
Querying the LMS megamodel

connectedModelElems

• Which system elements are directly connected to G6?

allConnectedModelElems

 Which system elements are directly and indirectly connected to G6?



package library

```
2
 3⊖ import "http://se.cs.toronto.edu/mmint/MID"
   import "http://se.cs.toronto.edu/mmint/MID/Relationship"
 6 pattern connectedModelElems(modelElemSrc: ModelElement,
 70
                                modelElemTgt: ModelElement) {
 8
     modelElemSrc != modelElemTgt;
 9
     Model.modelElems(modelSrc, modelElemSrc);
10
     Model.modelElems(modelTat, modelElemTat);
     modelSrc != modelTat;
11
     Mapping.modelElemEndpoints.target(mapping, modelElemSrc);
12
     Mapping.modelElemEndpoints.target(mapping, modelElemTgt);
13
14 }
15
16@pattern allConnectedModelElems(modelElemSrc: ModelElement,
170
                                   modelElemTgt: ModelElement) {
18
     modelElemSrc != modelElemTat:
     Model.modelElems(modelSrc, modelElemSrc);
19
20
     Model.modelElems(modelTat, modelElemTat);
     modelSrc != modelTgt;
21
     find connectedModelElems+(modelElemSrc, modelElemTgt);
22
23 }
```

Querying the LMS megamodel

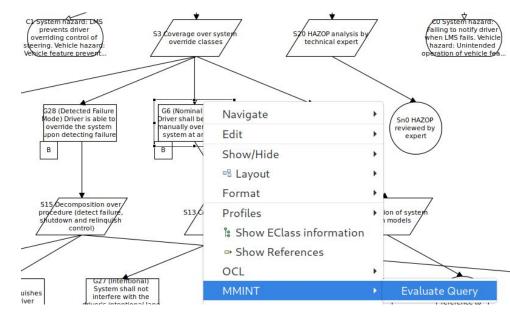
connectedModelElems

• Which system elements are directly connected to G6?

allConnectedModelElems

• Which system elements are directly and indirectly connected to G6?

(Opposite direction works too: change in a system model, which goals are affected?)



```
1 package library
```

```
2
 3@ import "http://se.cs.toronto.edu/mmint/MID"
   import "http://se.cs.toronto.edu/mmint/MID/Relationship"
 6 pattern connectedModelElems(modelElemSrc: ModelElement,
 70
                                modelElemTat: ModelElement) {
 8
     modelElemSrc != modelElemTgt;
 9
     Model.modelElems(modelSrc, modelElemSrc);
10
     Model.modelElems(modelTat, modelElemTat);
     modelSrc != modelTat;
11
     Mapping.modelElemEndpoints.target(mapping, modelElemSrc);
12
     Mapping.modelElemEndpoints.target(mapping, modelElemTgt);
13
14 }
15
16@pattern allConnectedModelElems(modelElemSrc: ModelElement,
170
                                   modelElemTgt: ModelElement) {
18
     modelElemSrc != modelElemTat:
     Model.modelElems(modelSrc, modelElemSrc);
19
20
     Model.modelElems(modelTat, modelElemTat);
     modelSrc != modelTgt;
21
22
     find connectedModelElems+(modelElemSrc, modelElemTgt);
23 }
```

MMINT demo

Conclusion

- Developed tool support for automotive model management with integrated querying
- Identified query engine requirements and compared between OCL and VQL
 - VQL is easier to use and faster
- Showcased three scenarios using the LMS example from industry
- Challenges:
 - creating a Query Abstraction Layer to plug in arbitrary languages
 - creating a query library for common tasks

Future work

- Expand the LMS megamodel with more safety-related artifacts (e.g., hazard analysis, FTA, test results, etc.) and write queries on top of them
- Evaluation of effectiveness and usability
- Expand library of megamodel queries
- Display results graphically
- Experiment with live queries

Thank you!

MMINT: https://github.com/adisandro/MMINT

Alessio Di Sandro, Sahar Kokaly, Rick Salay, Marsha Chechik {adisandro, skokaly, rsalay, chechik}@cs.toronto.edu

University of Toronto

MASE, Sep 15 2019, Munich, Germany



- Test the scalability requirement #4
- OCL QAL implementation
- 3 example scenarios
 - a. safety case change
 - b. identify medium risk elements:

(hazards with Automotive Safety Integrity Level == B)

c. identify highly interconnected elements:

(elements with #connections > 5)

• Execution times for 3 example scenarios:

Scenario	OCL time (s)	VQL time (s)
1	0.411	0.686
2	2.220	0.830
3	32.996	0.599

- Threats to validity:
 - limited expertise with OCL and VQL queries
 - only 3 scenarios