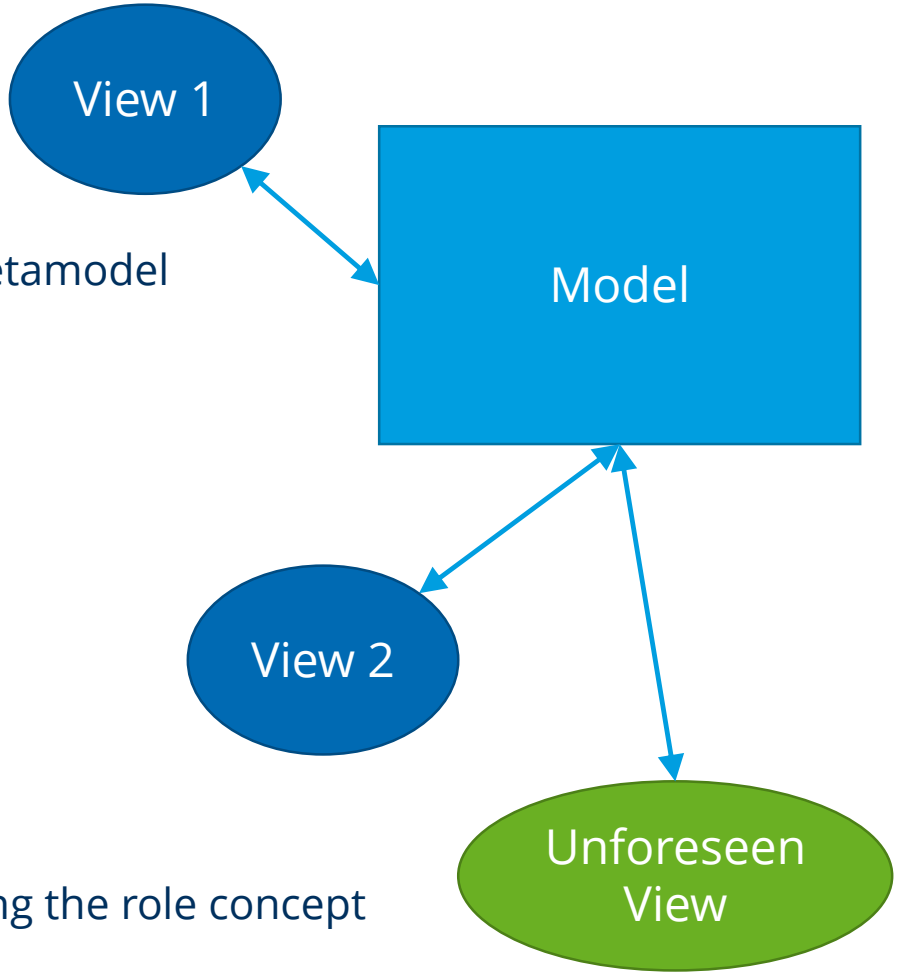
Christopher Werner
Institute of Software and Multimedia Technology, Software Technology Group

A Generic Language for Query and Viewtype Generation By-Example

Christopher Werner, Manuel Wimmer, and Uwe Aßmann

München, 15th September 2019

Query By-Example



Create viewtypes representing parts of the metamodel

Using a Domain Specific Language (DSL)
(e.g. ModelJoin [Burger2016])

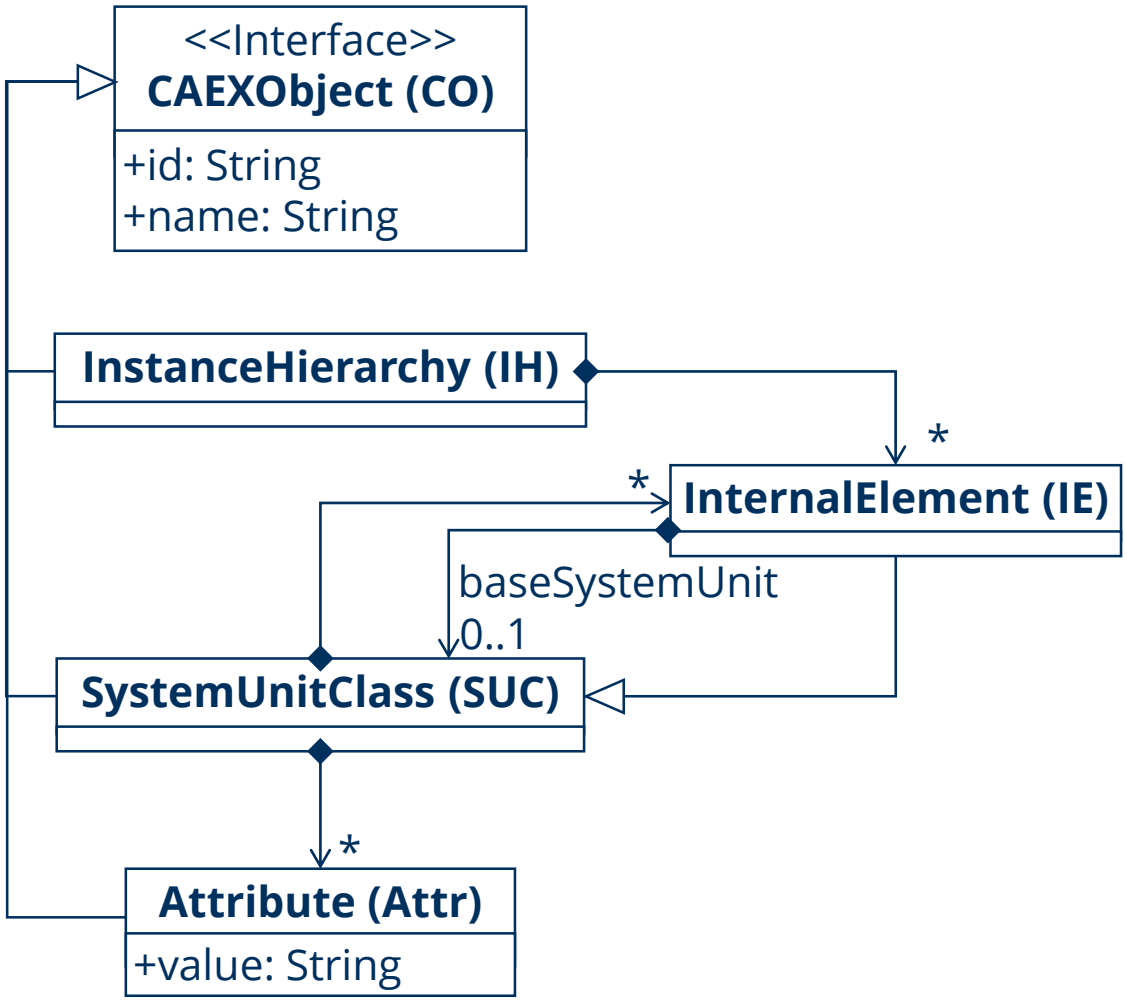
Problems:

- Learning DSL syntax
- Knowledge about underlying metamodel

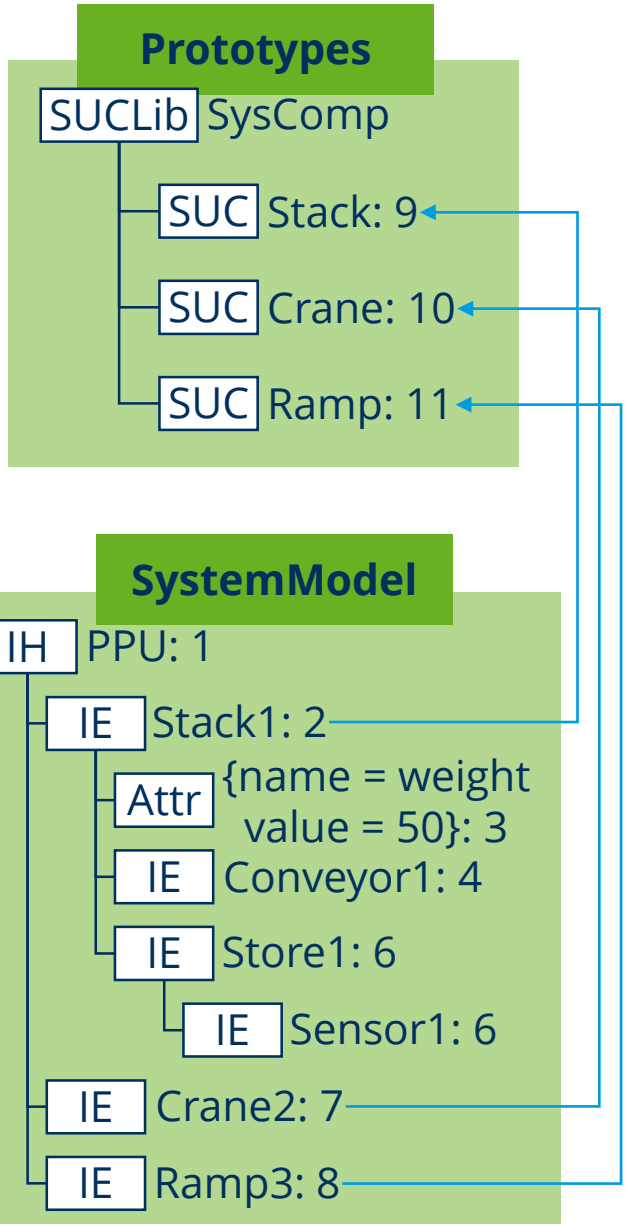
Present a query by-example mechanism:

1. Create queries from existing elements using the role concept
2. Automatically create ModelJoin syntax
3. Create viewtypes to visualize and modify the query results

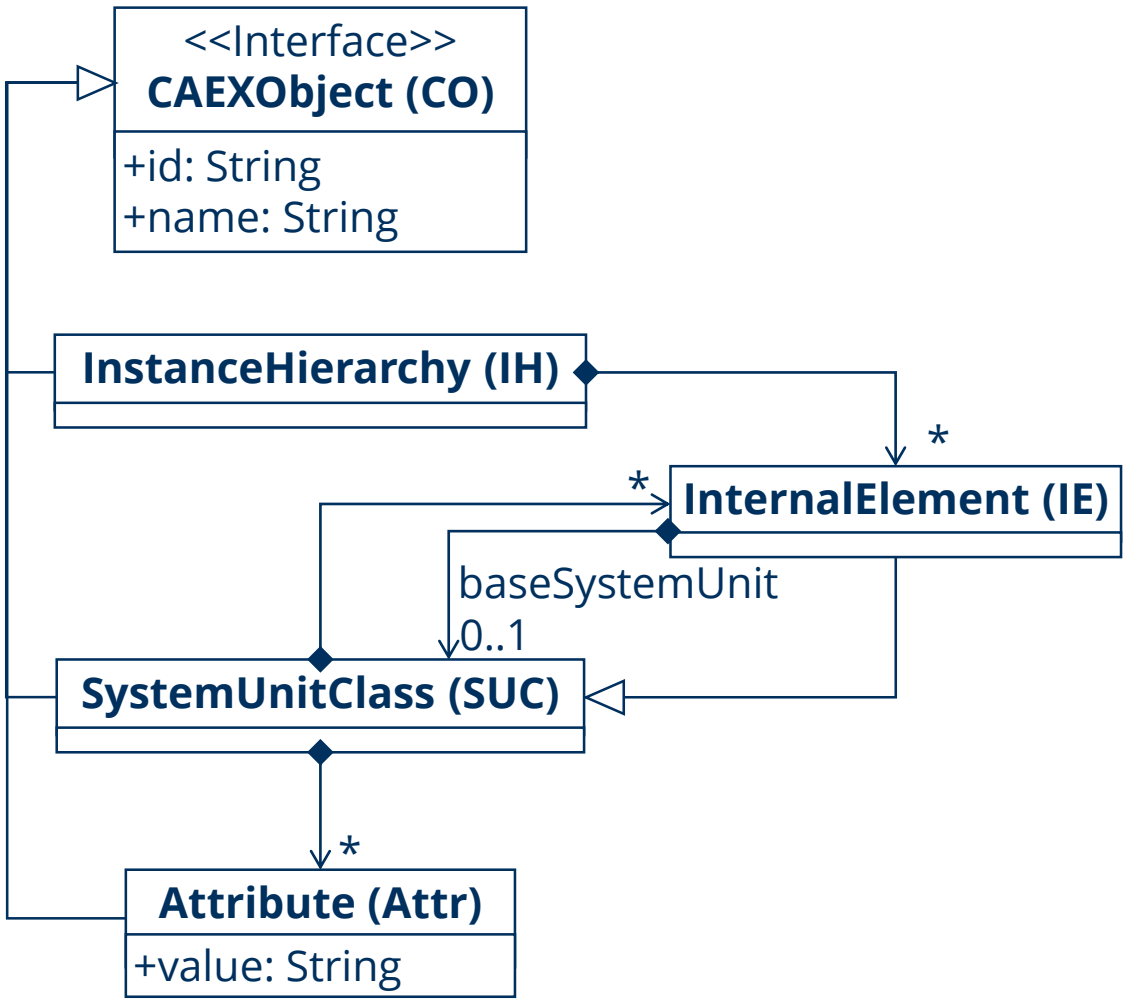
Running Example (AutomationML)



[Wimmer2018]



Queries on AutomationML



Queries:

Query 1:

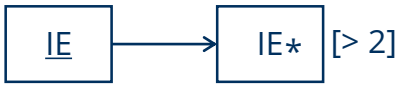
Query 2:

Query 3:

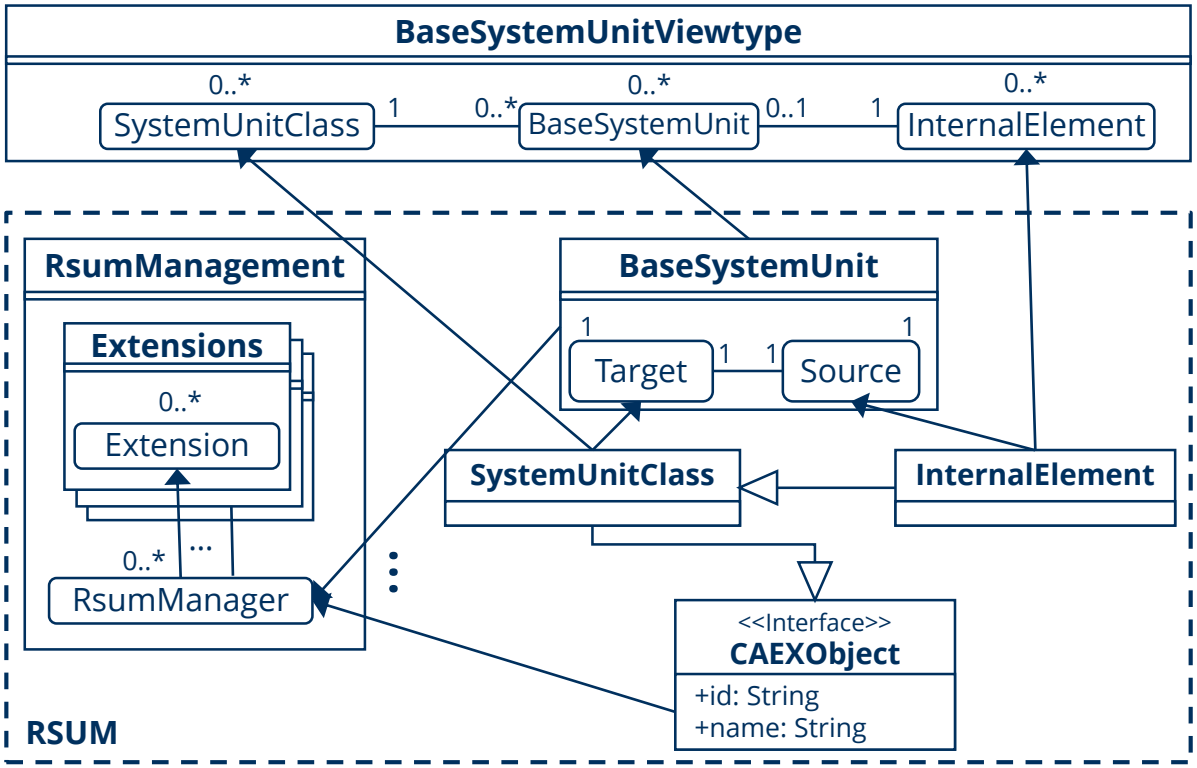
Query 4:

Query 5:

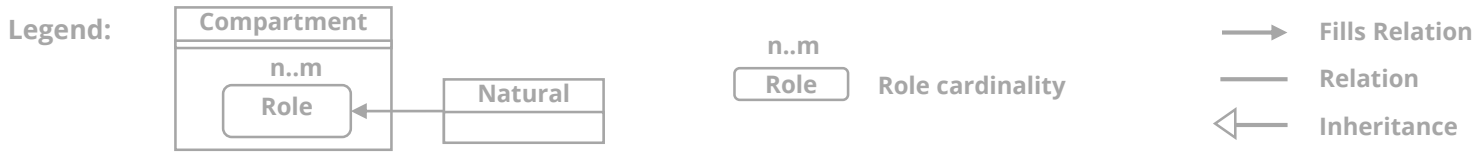
Query 6:



Role-based Single Underlying Model Approach (RSUM)



[Werner2018]



Advantages of the Role Concept and RSUM

Role advantages:

- Roles adapt the behavior of the naturals
- Roles are context dependent
- Bind and unbind roles at runtime

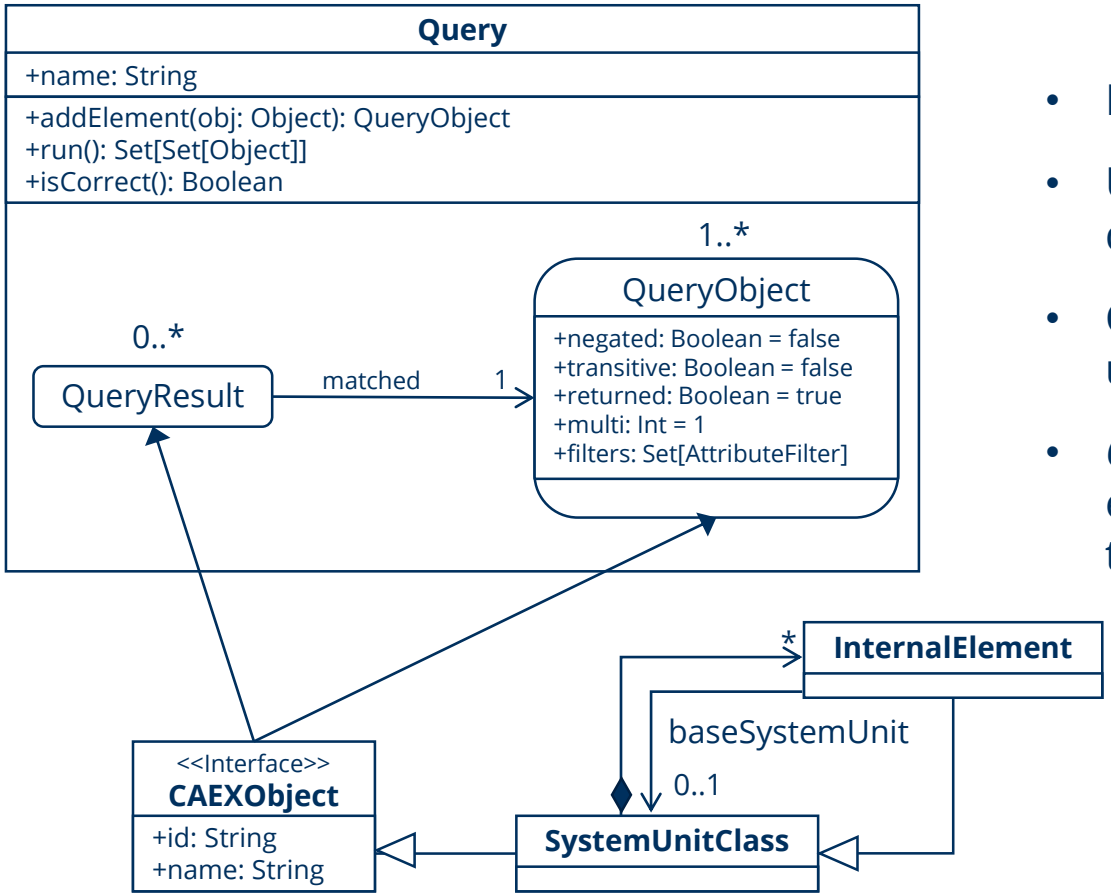
RSUM advantages:

- Add and remove naturals and relational compartments at runtime
- Add and remove viewtypes at runtime

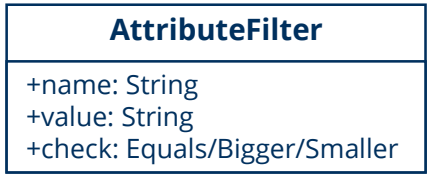
Runtime Adaptation

Fine granular Authorization

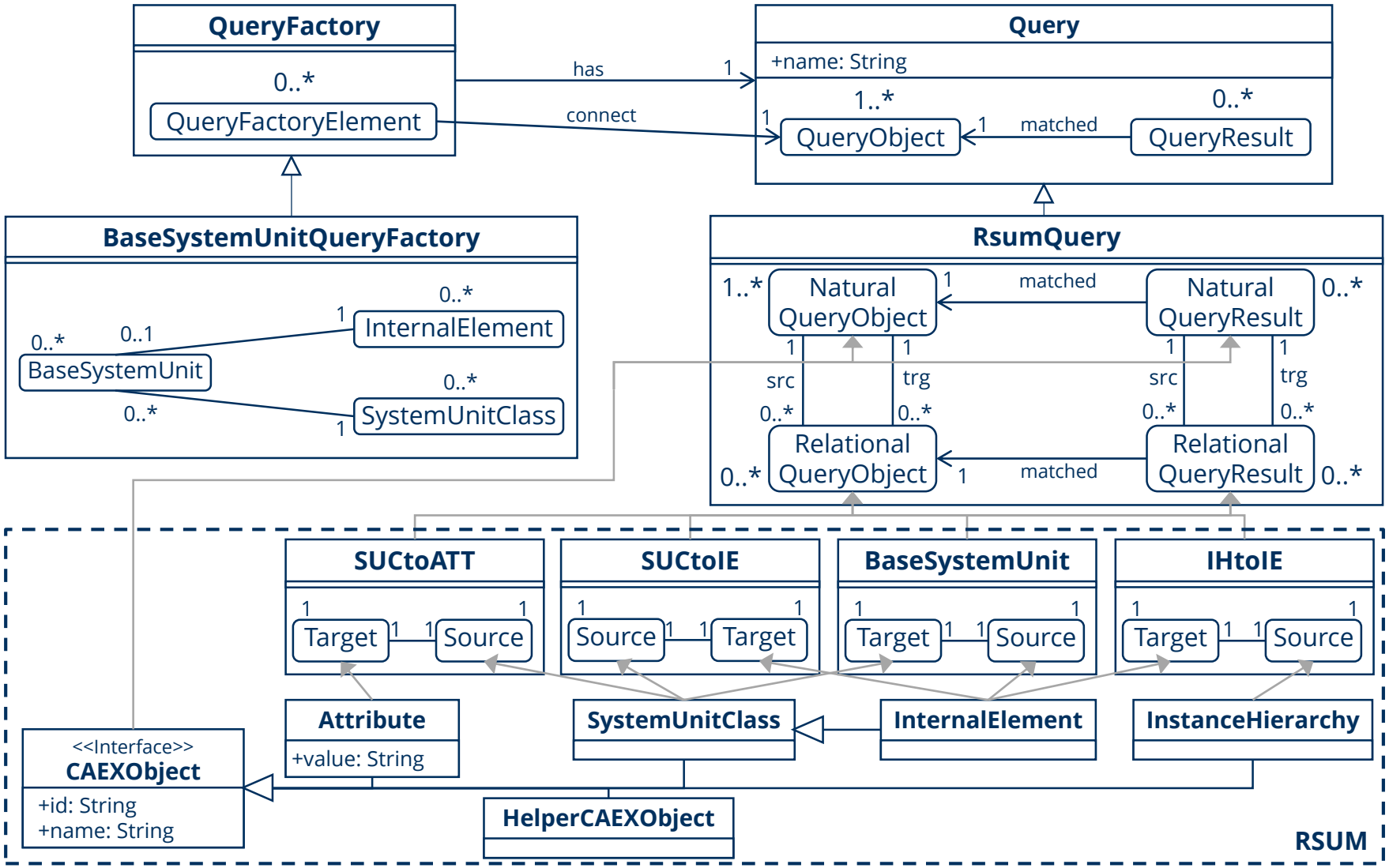
General Concept



- Bind *QueryObject* to each element
- Use one element for multiple queries
- Concept independent from underlying model
- *QueryResult* roles are bound to each query result after running the query



Adapted Query Concept for RSUM



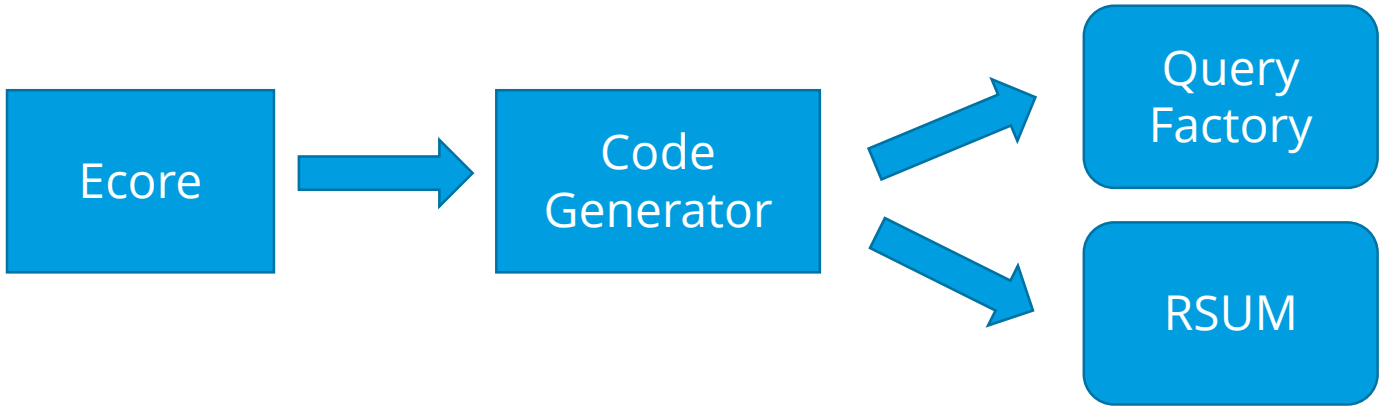
Adapted Query Concept for RSUM

RsumQuery compartment works for all naturals and relational compartments

Helper classes allow formulation of generic queries of interfaces and abstract classes

Code generator:

- Generate naturals and relational compartments in RSUM from Ecore model
- Generate a query factory compartment from the Ecore model
- Add helper classes for all interfaces and abstract classes (can be created in factory)

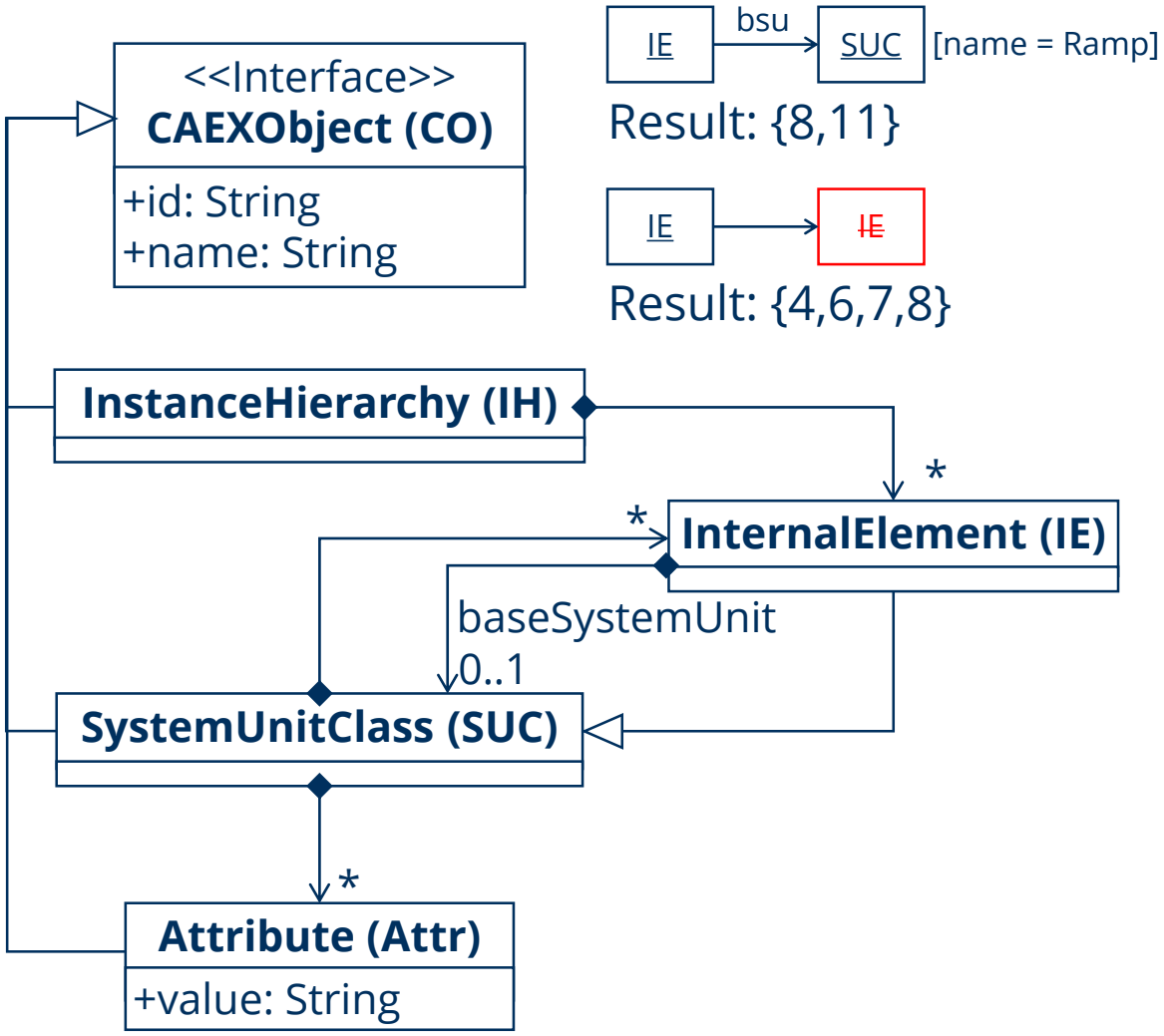


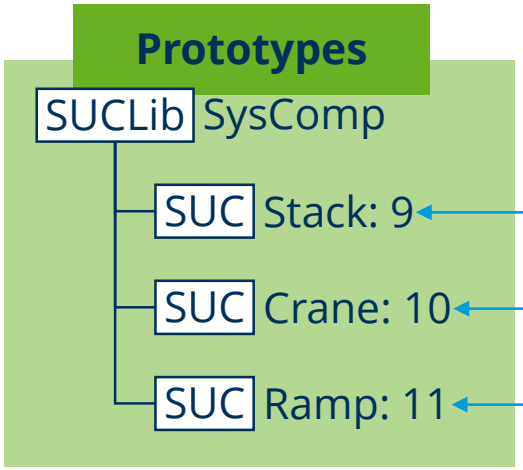
Creation of Queries

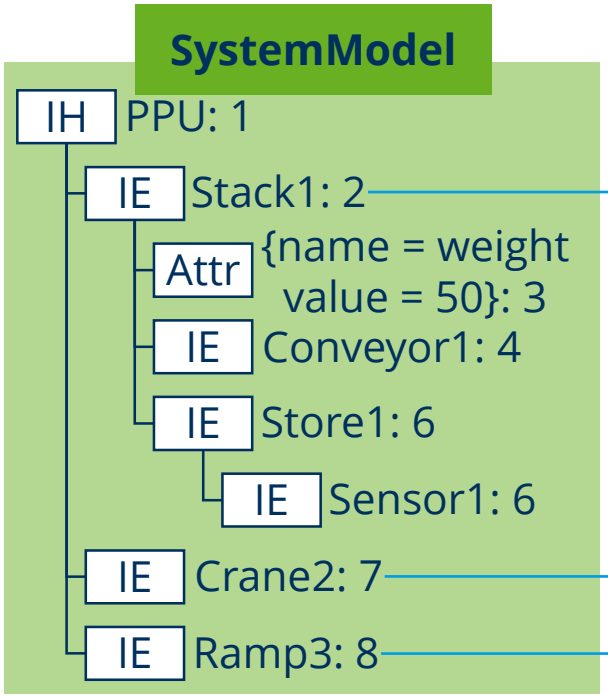
```
RsumQuery q3 = new RsumQuery("Query 3")
InternalElement ie3 = /*InternalElement in RSUM*/
SystemUnitClass suc = /*SystemUnitClass in RSUM*/
BaseSUnit bsu = /*Relational compartment between suc & ie3*/
QueryObject r0 = q3.addQueryRole(ie3)                    //Add Query Roles
QueryObject r1 = q3.addQueryRole(suc)
QueryObject r2 = q3.addQueryRole(bsu)
r1.addAttributeFilter("name", “Ramp", CheckOption.equals)   //Set Properties
Set[Set[Objects]] result = q3.run()                      //Run Query
```


```
AmlQueryFactory q5 = new AmlQueryFactory
InternalElement ie1 = q5.createInternalElement()
InternalElement ie2 = q5.createInternalElement()
ie1.addInternalElements(ie2)
ie2.getQueryObject.negated = true                        //Set Properties
Set[Set[Objects]] result = q5.getQuery().run()           //Run Query
```

Results of Queries







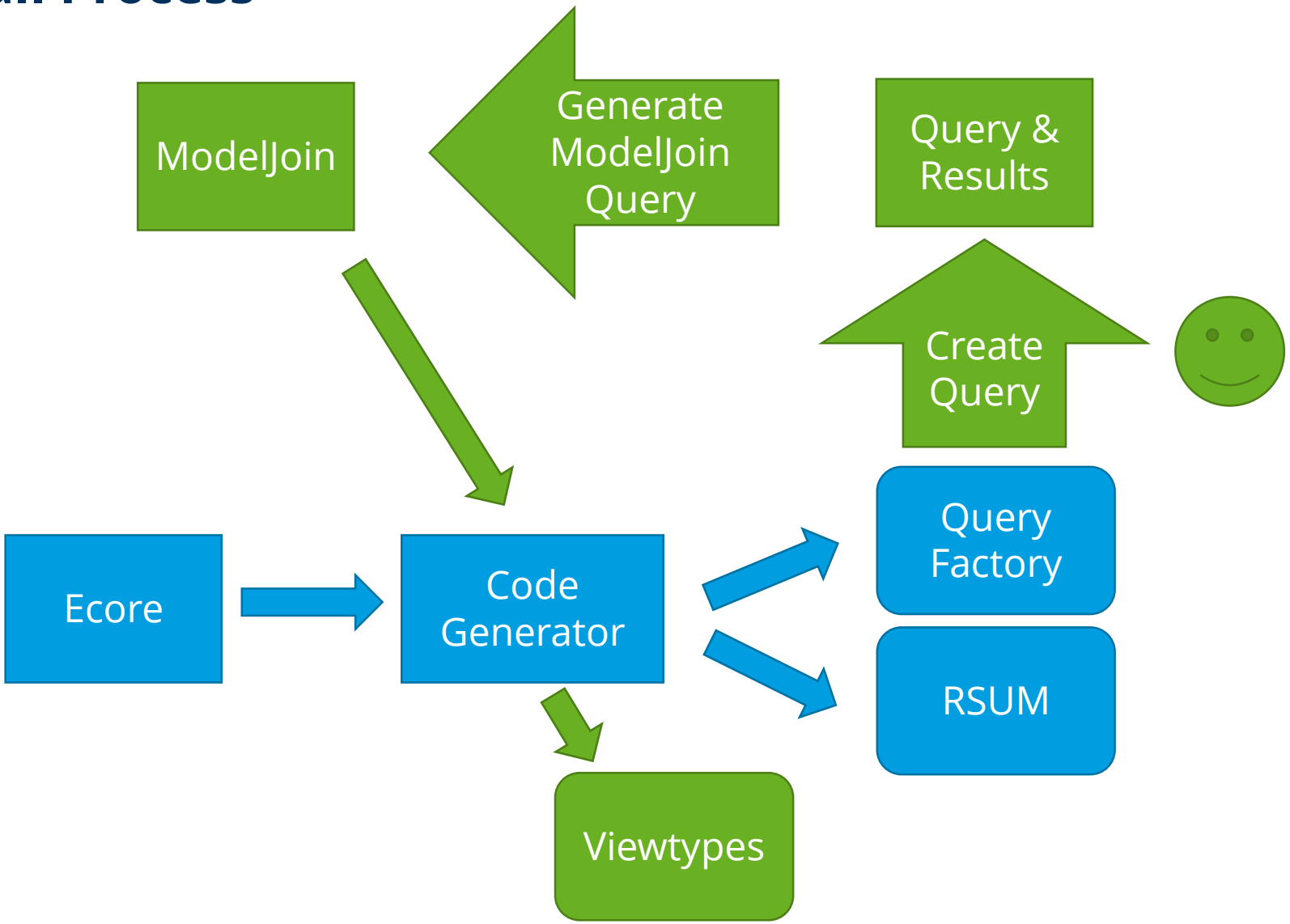
ModelJoin Representation

ModelJoin [Burger2016]: SQL like syntax, approach to create flexible viewtypes


```
natural join aml.InternalElement with aml.InternalElement as aml.InternalElement {
   keep attributes aml.InternalElement.name
   keep attributes aml.InternalElement.id
   keep outgoing aml.InternalElement.baseSystemUnit as type aml.SystemUnitClass {
      keep attributes aml.SystemUnitClass.name
      keep attributes aml.SystemUnitClass.id
   }
}
```


```
natural join aml.InternalElement with aml.InternalElement as aml.InternalElement {
   keep attributes aml.InternalElement.name
   keep attributes aml.InternalElement.id
}
```

Overall Process



Conclusion & Future Work

- Presented a generic approach for a by-example query language
- Showing the applicability of this approach using the example of AML
- Implemented upon the RSUM approach
- Create ModelJoin queries from the by-example queries
- Generate viewtypes with the ModelJoin queries

Future Work

- Extend the expressive power of the query language
  - Allow more complex queries
  - Provide more ModelJoin features
- User interface support for the creation of the queries
- Optimization of the queries

References

- **[Burger2016]** E. Burger, J. Henss, M. Küster, S. Kruse, and L. Happe, “View-based model-driven software development with ModelJoin,” Software & Systems Modeling, vol. 15, no. 2, pp. 473–496, 2016.
- **[Werner2018]** C. Werner and U. Aßmann, “Model Synchronization with the Role-oriented Single Underlying Model,” MART Workshops, pp. 62–71, 2018.
- **[Wimmer2018]** M. Wimmer and A. Mazak, “From AutomationML to AutomationQL: A By-Example Query Language for CPPS Engineering Models,” in IEEE 14th International Conference on Automation Science and Engineering (CASE), 2018, pp. 1394–1399