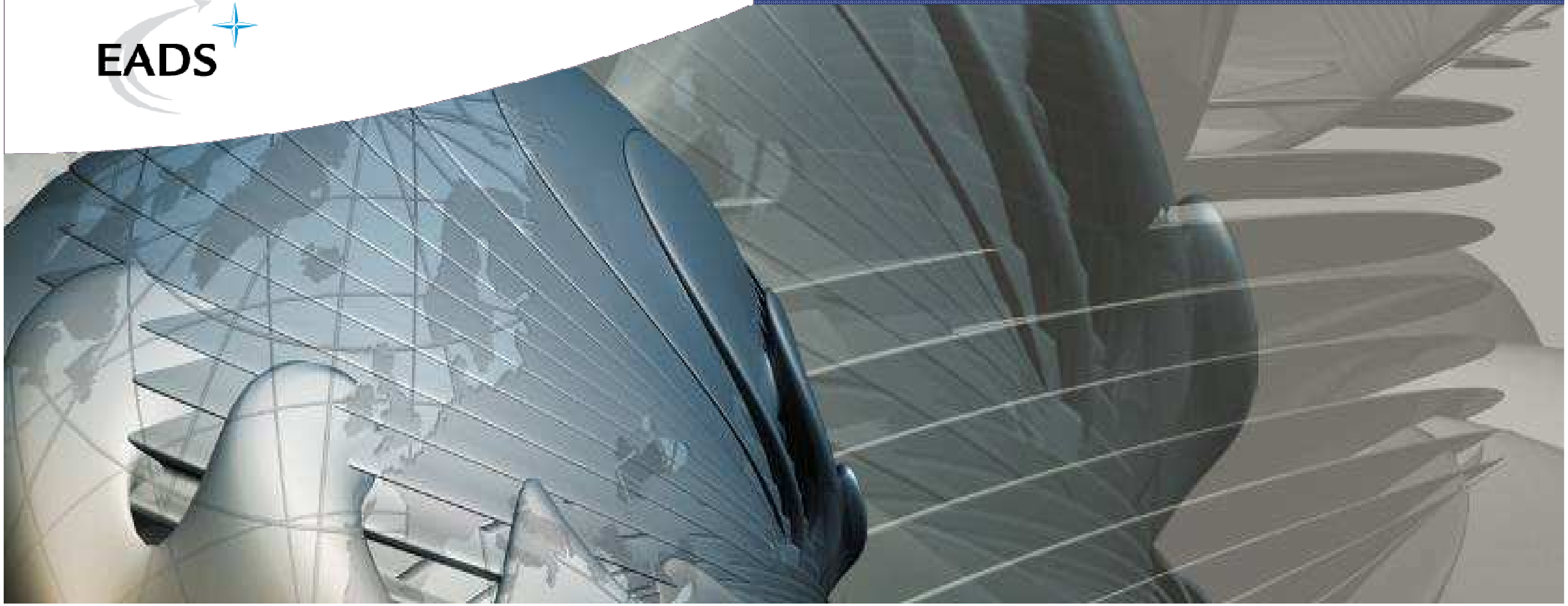
EADS



Knowledge Models to Reduce the Gap between Heterogeneous Models

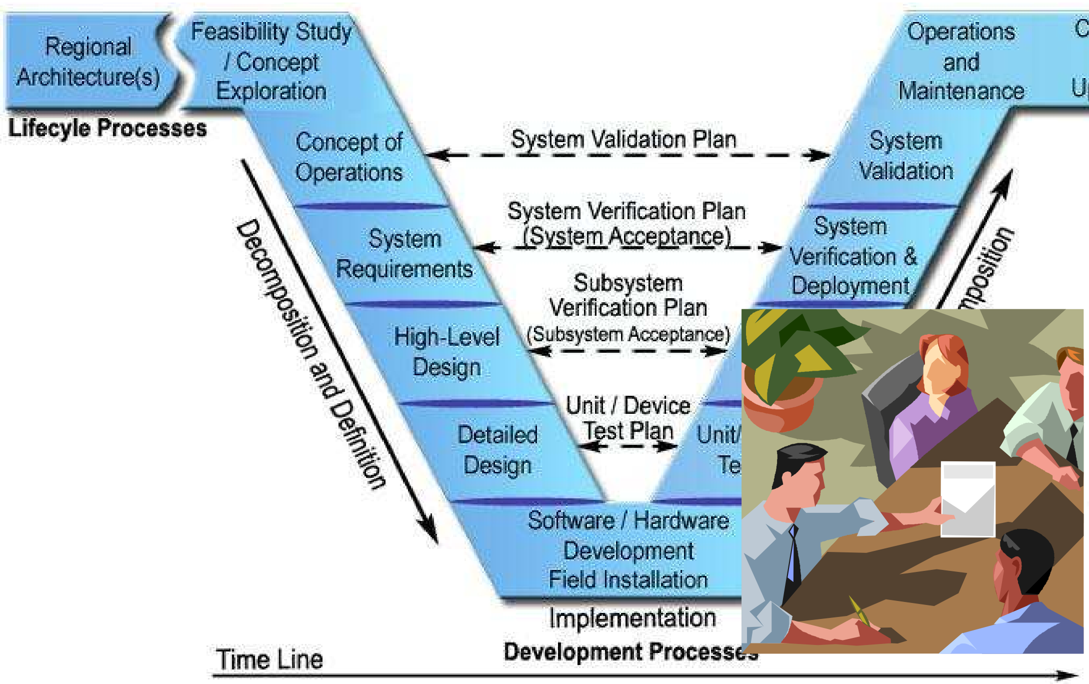
Application to aircraft systems engineering

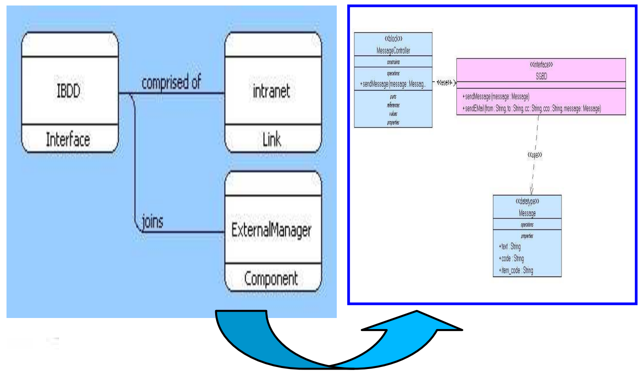
ICECCS 2010 UML & AADL Workshop

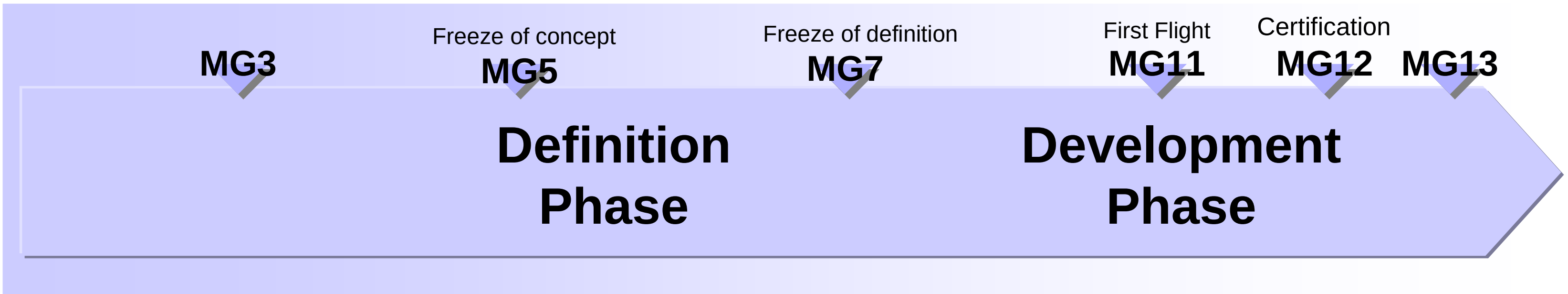
David SIMON ZAYAS (EADS IW)
Dr. Anne MONCEAUX (EADS IW)
Dr. Yamine AIT AMEUR (LISI-ENSMA)

Context of the problem

Aircraft Systems Engineering processes







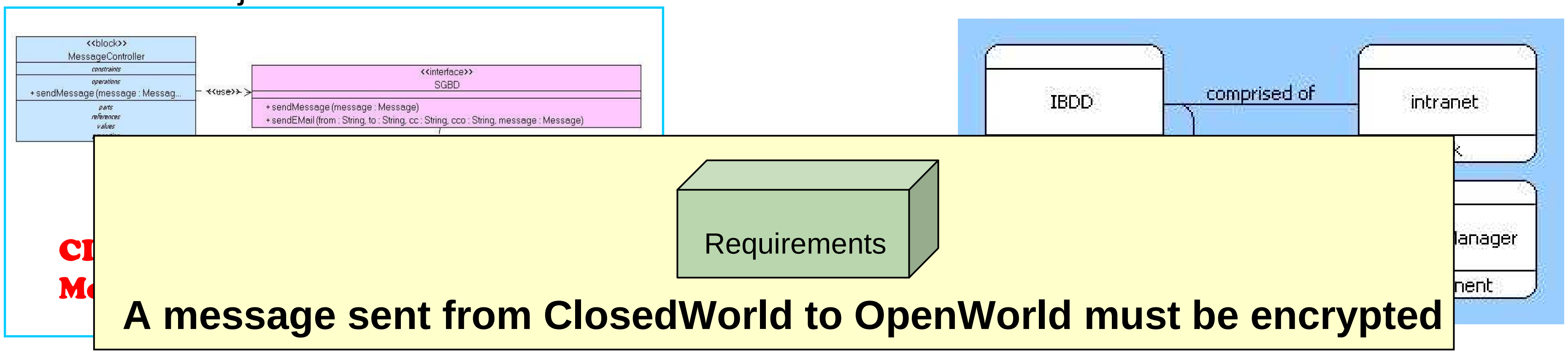
Context of the problem

Heterogeneous multi-modelling

- Two models describing parts of the aircraft Information System

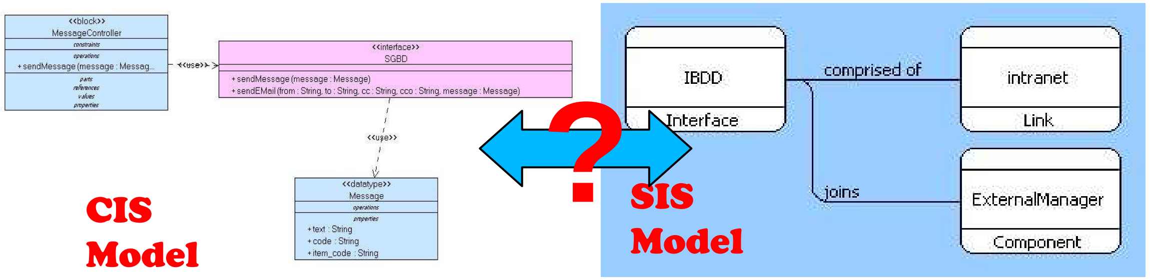
- CIS: Cockpit Information System model
  - CIS sends messages to “external” systems
  - ClosedWorld System
  - SysML modelling language
  - Object-oriented

- SIS: Shared Information System
  - SIS treats messages from CIS
  - OpenWorld System
  - CORE modelling language
  - Functional-oriented



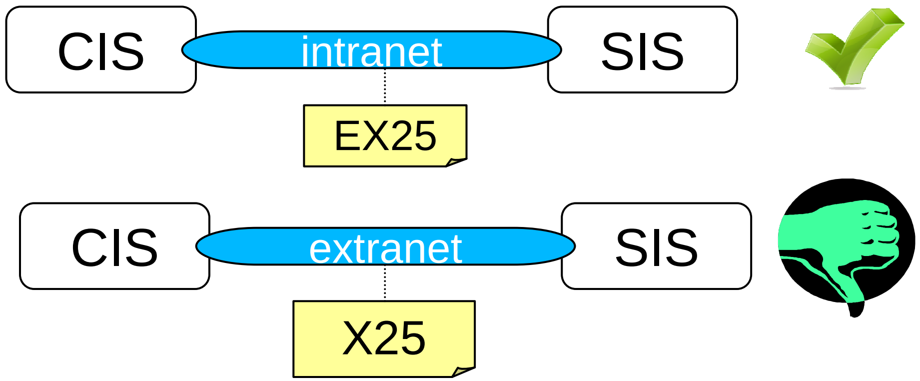
Context of the problem

Implicit knowledge

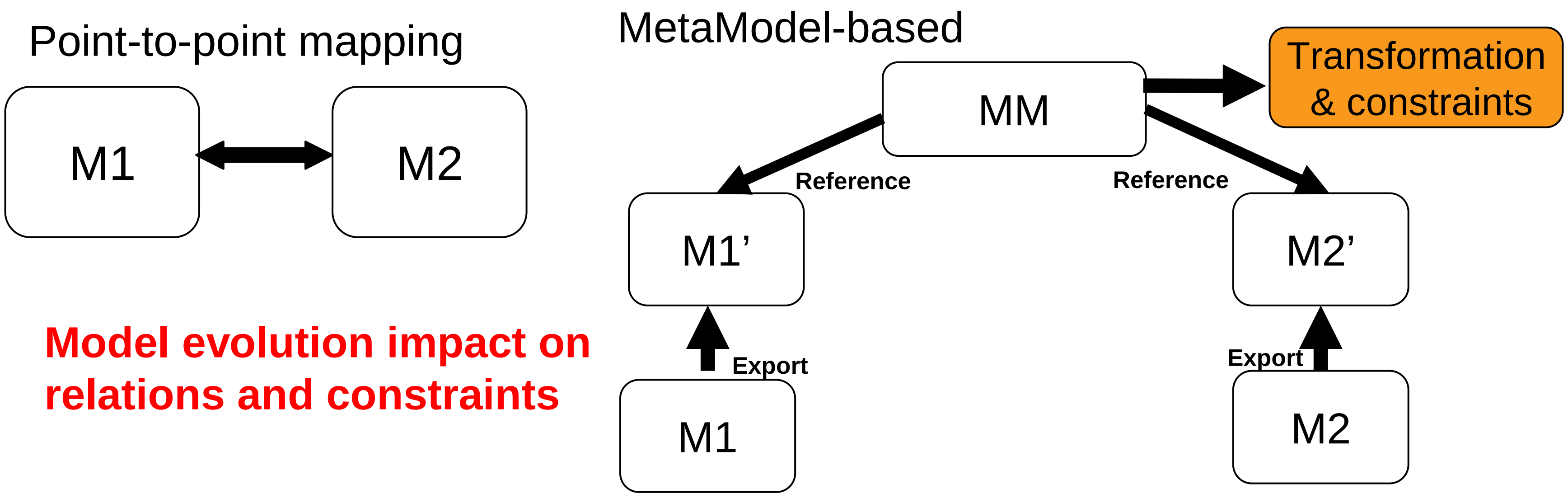


- To relate the two models engineers need to agree on the following concepts:
  - ClosedWorld
  - OpenWorld
  - Message
  - Encrypted protocol

- If this is done we can exploit relations: e.g. to check the requirement fulfillment



Inter-model Relationships Creation



Model evolution impact on relations and constraints

No explicit additional knowledge

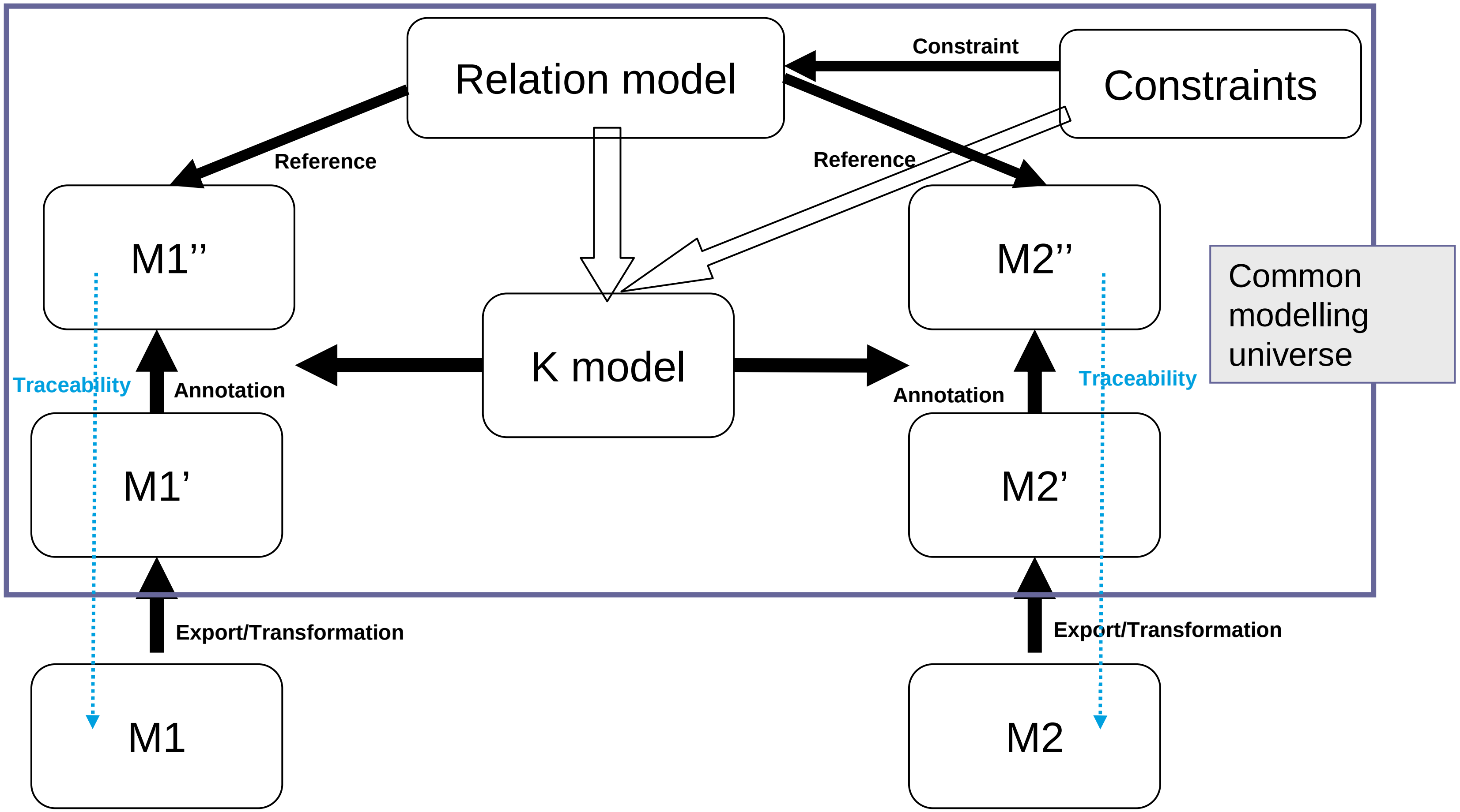
Goals

- Manage inter-models heterogeneity in a system engineering setting

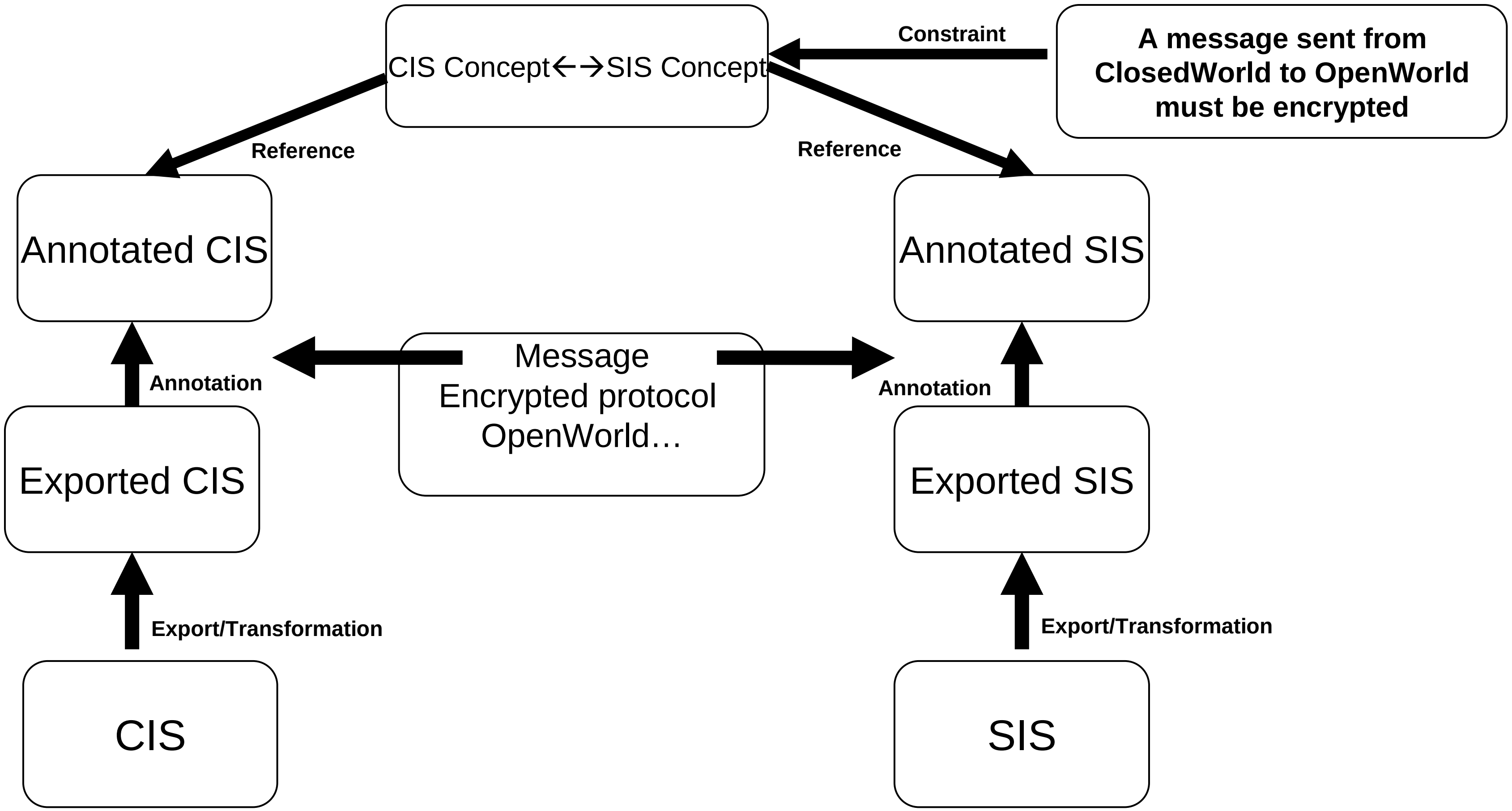
- Our proposal
  - To describe, model and verify
    - some inter-model constraints and relationships
    - between pre-existing heterogeneous models (analysis)
    - used in a System Engineering process

  - To make explicit, formalize and exploit additional knowledge
    - usually not expressed by the engineers
    - to express these constraints and relationships

Knowledge-based inter-model relationships management

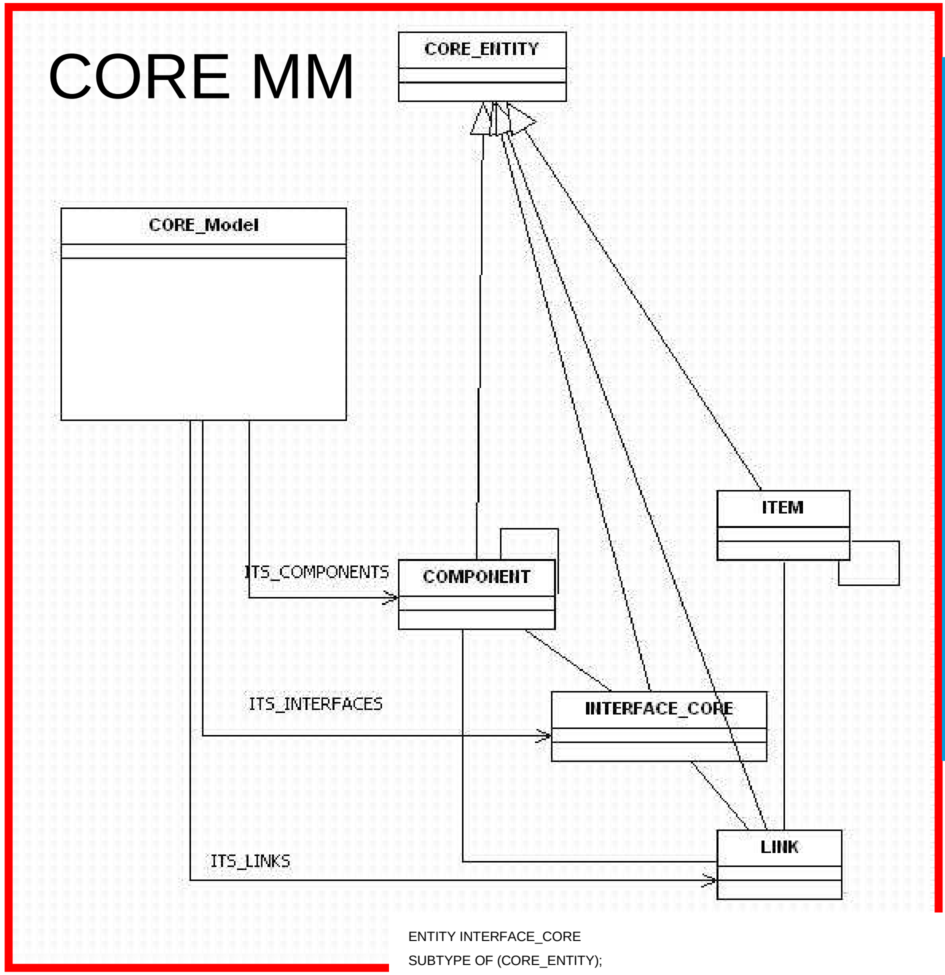


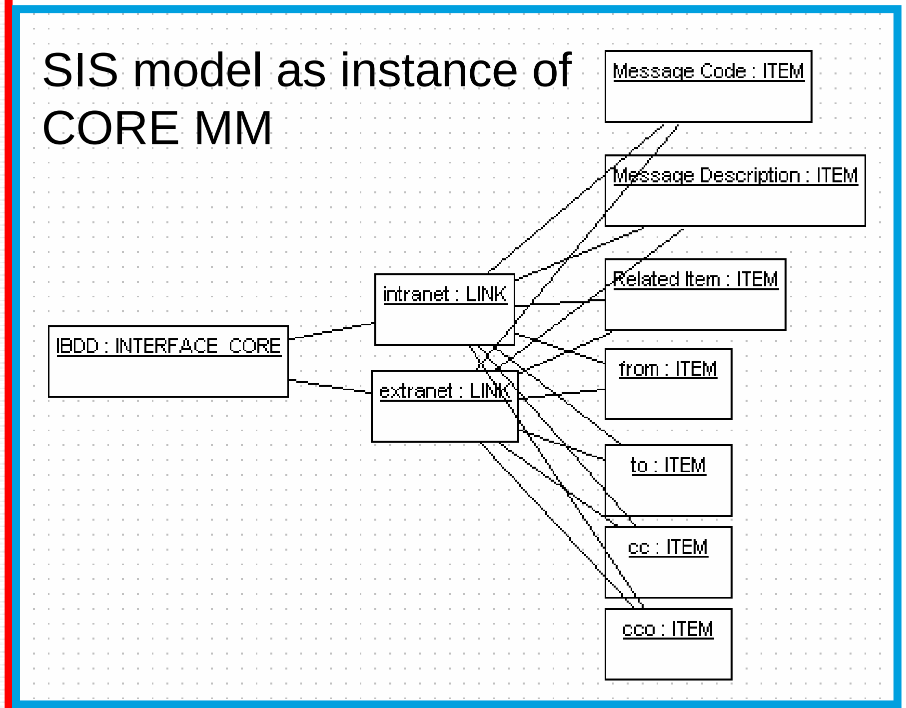
Knowledge-based inter-model relationships management



Step by Step

Step 1 (Export to a common modelling formal language)





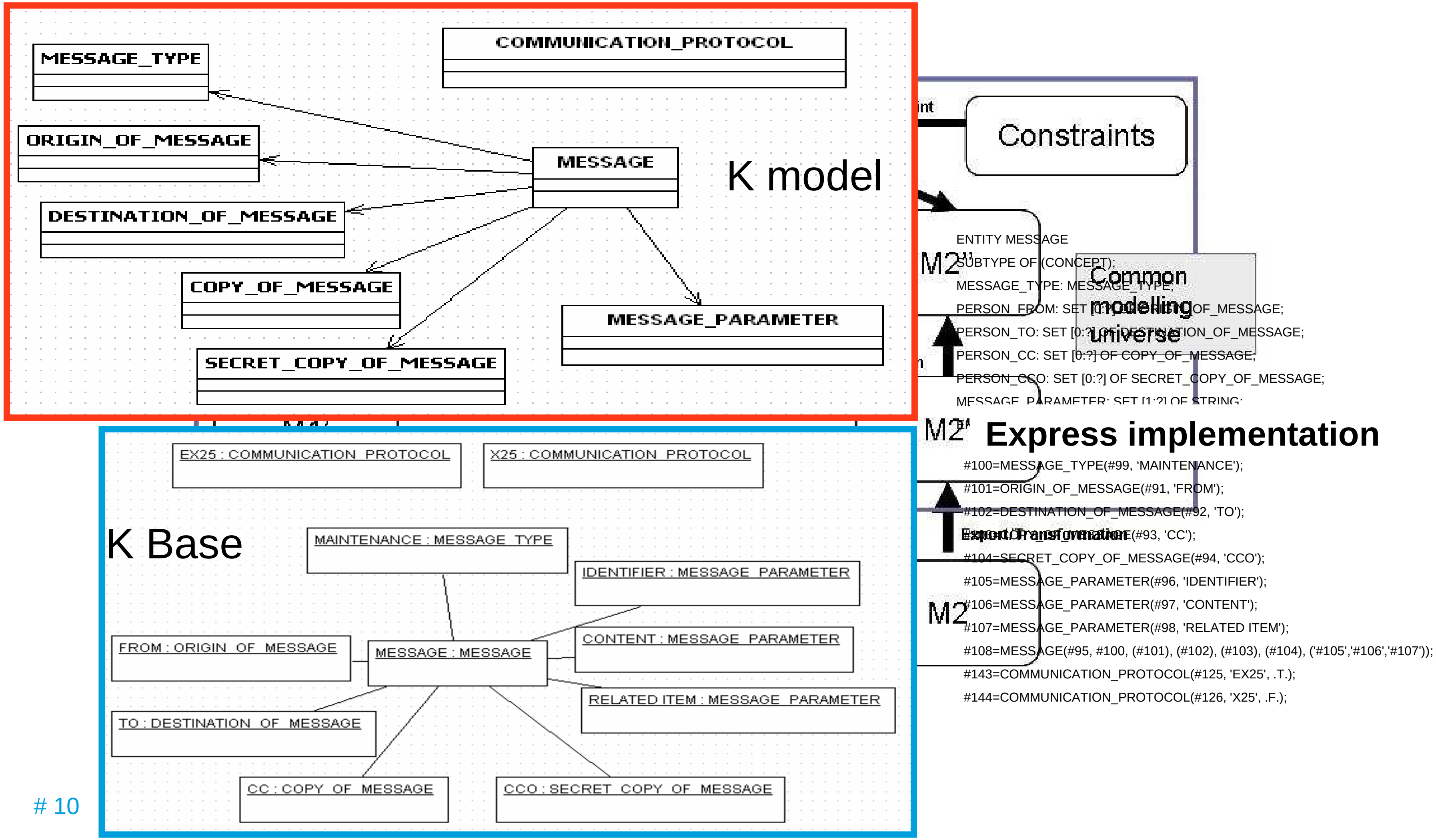
M2

```
ENTITY INTERFACE_CORE
SUBTYPE OF (CORE_ENTITY);
COMPRISED_OF: SET[0:?] OF LINK;
SPECIFIED_BY: SET[0:?] OF REQUIREMENT;
INVERSE
JOINS_THROUGH: SET[0:?] OF COMPONENT FOR JOINED_THROUGH;
JOINS_TO: SET[0:?] OF COMPONENT FOR JOINED_TO;
END_ENTITY;
```

Express implementation

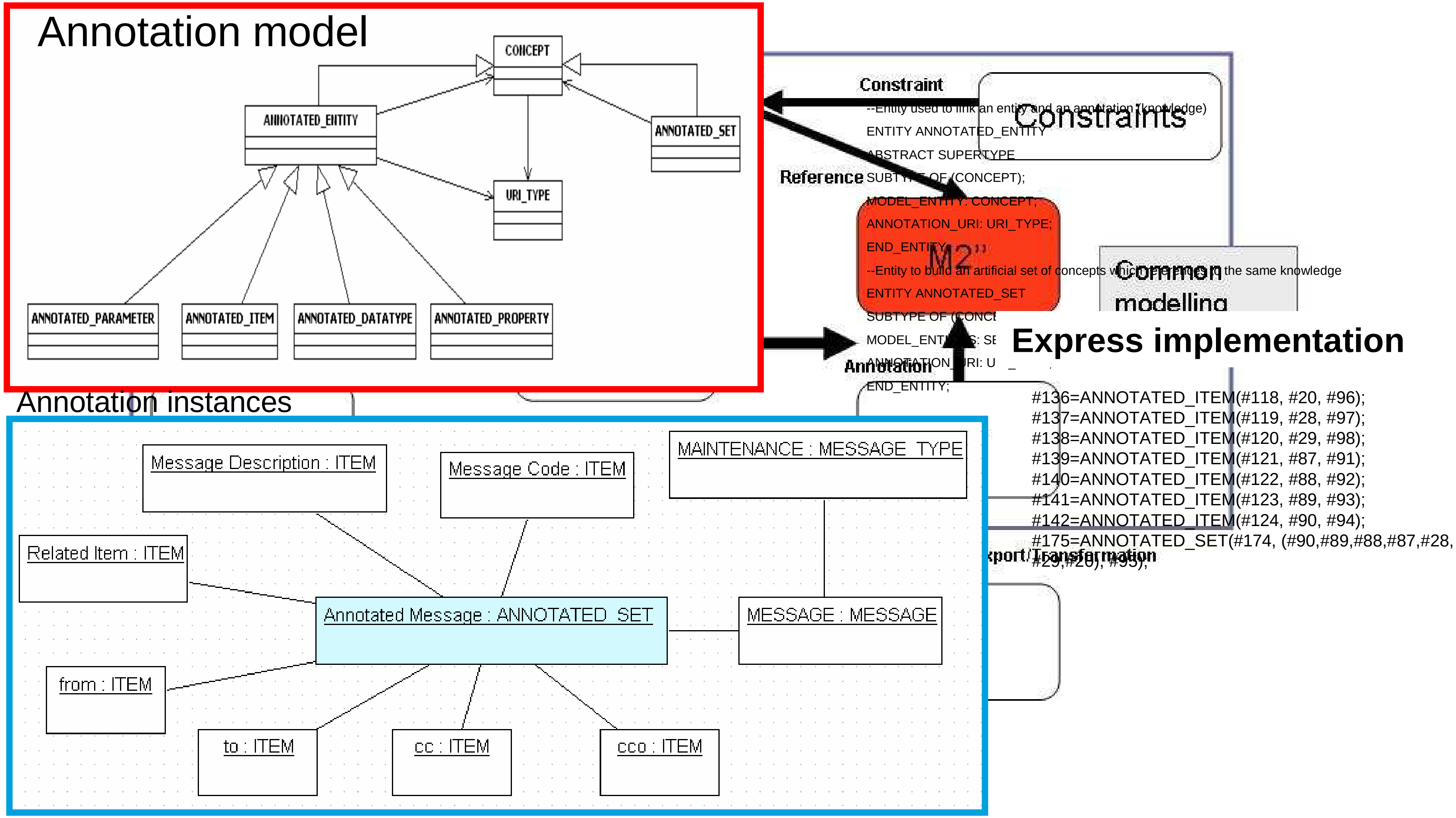
```
tt#16=LINK(#48, #15, 'dsz', $, #15, 'intranet', $, $, $, $, $, 'EX25', (),
(#90,#89,#88,#87,#28,#29,#20));
#20=ITEM(#52, #15, 'dsz', $, #15, 'Message Code', $, $, $, $, $, $, $, $, $, (), (), (),
());
#28=ITEM(#60, #15, 'dsz', $, #15, 'Message Description', $, $, $, $, $, $, $, $, $, (),
(), (), ());
z', $, #15, 'Related Item', $, $, $, $, $, $, $, $, $, (), (), (), ());
z', $, #15, 'from', $, $, $, $, $, $, $, $, $, (), (), (), ());
z', $, #15, 'to', $, $, $, $, $, $, $, $, $, (), (), (), ());
#89=ITEM(#85, #15, 'dsz', $, #15, 'cc', $, $, $, $, $, $, $, $, $, (), (), (), ());
#90=ITEM(#86, #15, 'dsz', $, #15, 'cco', $, $, $, $, $, $, $, $, $, (), (), (), ());
```

Step 2 (Formal side knowledge model)



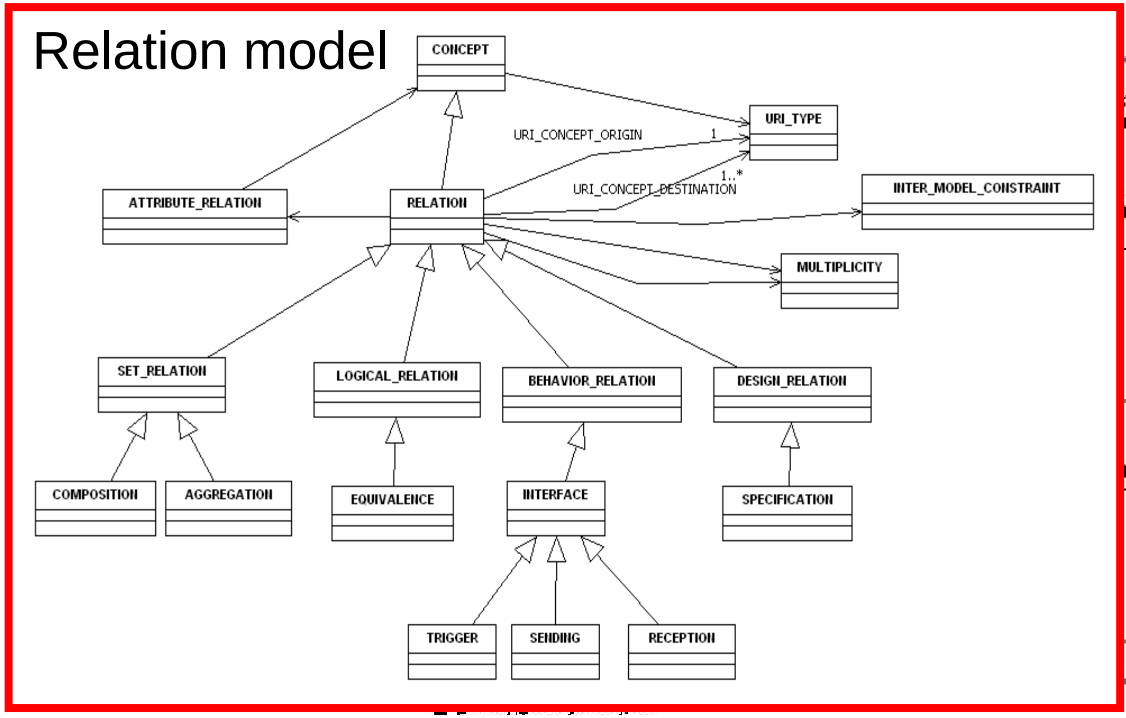
Step by Step

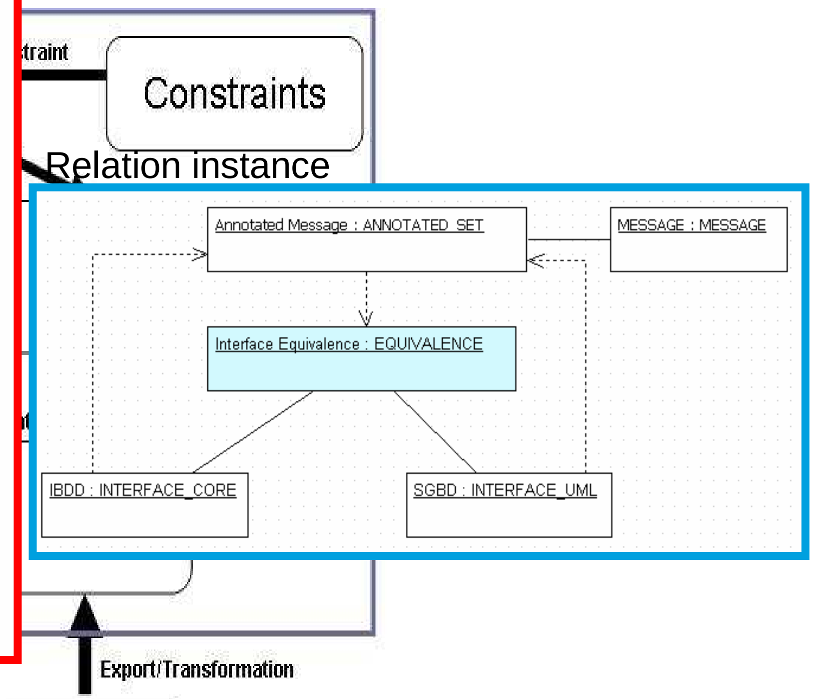
Step 3 (Knowledge based model annotation)



Step by Step

Step 4 (Inter-models relations)





--This entity represents a relation between 2 or more elements of different models
ENTITY RELATION
ABSTRACT SUPERTYPE
SUBTYPE OF (CONCEPT);
NAME : STRING;
URI_CONCEPT_ORIGIN: URI_TYPE;
URI_CONCEPT_DESTINATION: SET [1:?] OF URI_TYF
ATTRIBUTS: SET[0:?] OF ATTRIBUTE_RELATION;
MULTIPLICITY_ORIGIN: SET[0:2] OF MULTIPLICITY;
MULTIPLICITY_DESTINATION: SET[0:2] OF MULTIPLICITY;
CONSTRAINTS: SET [0:?] OF INTER_MODEL_CONSTRAINT;
END_ENTITY;

Express implementation

#33=EQUIVALENCE(#65, 'Interface Equivalence', #39, (#54), (),
#67), (#253));
RI_TYPE('EQUIV001');
#66=MULTIPLICITY(1, 1);
#67=MULTIPLICITY(1, 1);

Step by Step

Step 5 (Express constraints over the relations)

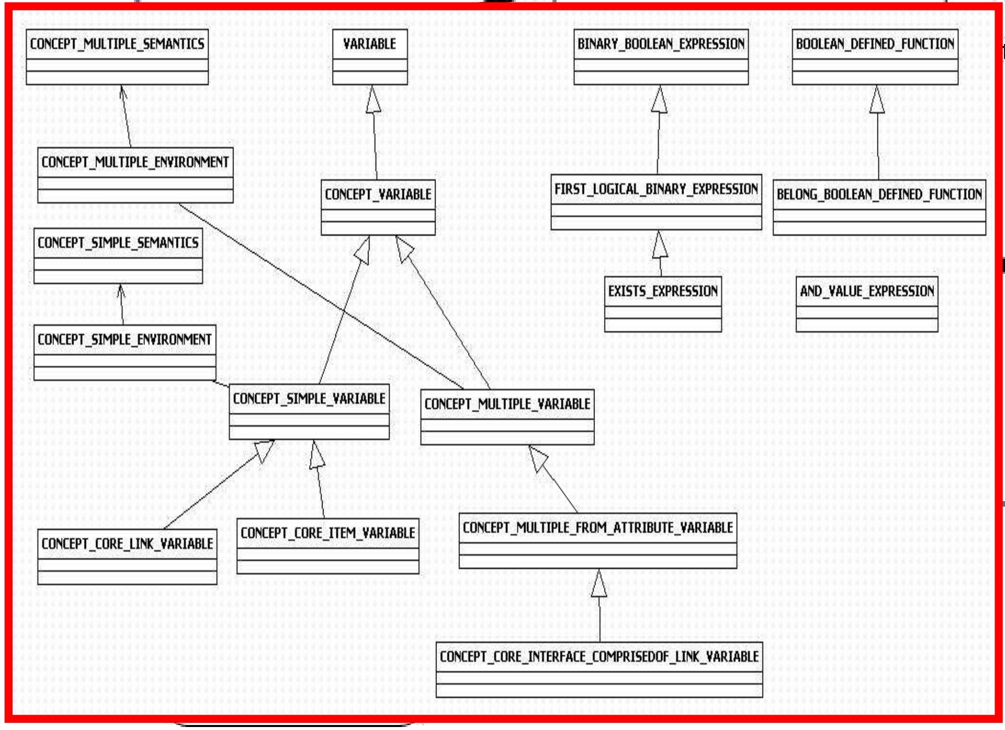
Constraints model based on
PLIB expressions model

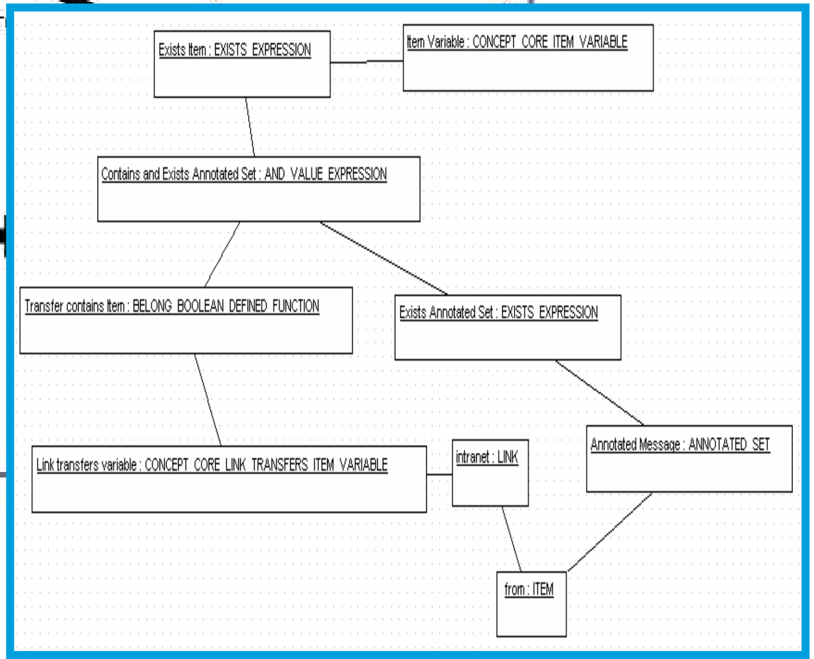
Relation model

Constraint

Constraints

"It exists at least one Link which transfers a maintenance message"

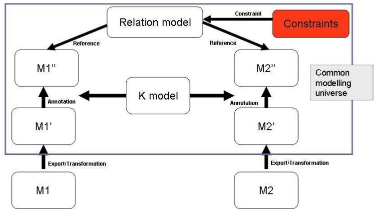





```
--This entity represents the EXISTS FOL assertion
ENTITY EXISTS_EXPRESSION
SUBTYPE OF (FIRST_LOGICAL_BINARY_EXPRESSION);
DERIVE
SELF\BOOLEAN_VALUE_EXPRESSION.BOOLEAN_VALUE. BOOLEAN := EXISTS_FUNCTION(SELF);
END_ENTITY;
```

Express implementation

```
#146=CONCEPT_SIMPLE_SEMANTICS(#90);
#147=CONCEPT_SIMPLE_ENVIRONMENT(#152, #146);
#151=BELONG_BOOLEAN_DEFINED_FUNCTION(#152, #153);
#152=CONCEPT_CORE_ITEM_VARIABLE(*, *);
#155=EXISTS_EXPRESSION((#152,#186), (#160), (#90,#89,#88,#87,#28,
  #29));
#156=CONCEPT_SIMPLE_SEMANTICS(#165);
_ENVIRONMENT(#160, #156);
_E_SEMANTICS($, (#16,#165));
_E_ENVIRONMENT(#161, #158);
_INK_VARIABLE(*, *);
#161=CONCEPT_CORE_INTERFACE_COMPRISEDOF_LINK_VARIABLE(*, *,
*, ());
#186=AND_VALUE_EXPRESSION((#151,#187));
#187=EXISTS_EXPRESSION((#180,#188), (#160,#152), (#175));
```



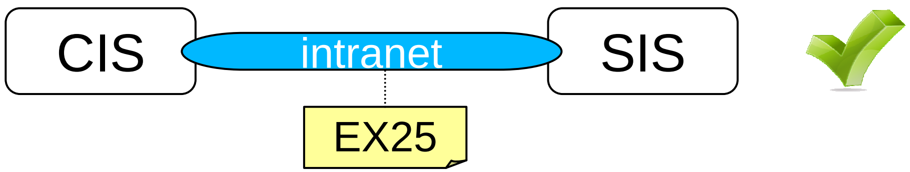
Step by Step

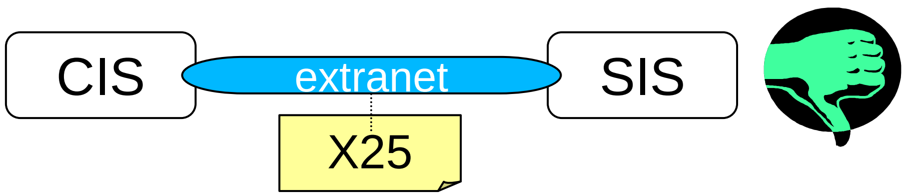
Step 5 (Check of constraint)

For each LINK transferring a maintenance message
from a Cockpit INTERFACE (CIS SysML model)
to a Shared System INTERFACE (SIS CORE model),
the used communication protocol must be secure

X: RELATION.URI_CONCEPT_ORIGIN
Y: RELATION.URI_CONCEPT_DESTINATION
Z: MAINTENANCE.URI

$$\forall l: Link \left(\left(\begin{array}{l} contains(Y.comprised\_of, l) \wedge \left(\exists i: Item \left(\begin{array}{l} contains(l.transfer, i) \wedge \\ \left(\exists as: AnnotatedSet \left(\begin{array}{l} as.annotation\_uri = Z \wedge \\ contains(as.model\_entities) \end{array} \right) \right) \end{array} \right) \right) \\ \wedge \exists o: Operation\_UML \left(\begin{array}{l} contains(X.owned\_operation, o) \wedge \\ \exists p: Parameter\_UML \left(\begin{array}{l} contains(o.owned\_parameter, p) \wedge \\ \exists adt: Annotated\_Data\_Type \left(\begin{array}{l} adt.concept\_uri = p.type\_uml \\ \wedge adt.annotated\_uri = Z \end{array} \right) \end{array} \right) \end{array} \right) \end{array} \right) \Rightarrow \left(\exists cp: Communication\_Protocol \left(cp.securised = TRUE \wedge l.protocol = cp.name \right) \right) \right)$$





Operational validation with Express modeling language

- Set of entities grouped in a schema
  - Entity = set of typed attributes
    - Derived attributes
    - Inverse attributes
  - Types
    - Base Types: INTEGER, REAL, BOOLEAN, STRING
    - Complex Types: Abstract Types LIST, SET, BAG, ARRAY with corresponding operations.
    - Entities used as attribute types.

```
SCHEMA Foo1;

ENTITY A;
  att_A: INTEGER;
INVERSE
  att_I: B FOR att_3;
END_ENTITY;
```

```
ENTITY B;
 att_1: REAL;
 att_2: LIST [0:?] OF STRING;
 att_3: A;
END_ENTITY;
```

```
ENTITY B2;
 att_1: REAL;
 att_2: LIST[0:?] OF STRING;
 att_3: A;
DERIVE
 att_4: BOOLEAN:=(SELF.att_3\A.att_A=
                 SIZEOF(SELF.att_2));
END_ENTITY;
```

```
#1=A(3);
#2=B(4.0, ('hello','bye'), #1);
```

Operational Validation

Express

- Schemas and entities can be constrained
- Constraints = First order logical expression.
    - Local Constraint: applied to an entity (WHERE Clause)
    - Global Constraint: applied to entities in a schema (RULE Clause).

Local Constraints

```
ENTITY A;
          att_A: INTEGER;
WHERE
 WR1 : (SELF.att_A >= 1) AND (SELF.att_A <= 10);

 WR2 : SELF.att_A = f(SELF);

END_ENTITY;
```

Global Constraint

```
RULE Card FOR A;
 WHERE
   SIZEOF(QUERY(inst <* A|(inst.att_A=1))) = 2;
END_RULE;
```

Conclusion

- Approach integrating Systems Engineering models
- Specificity of the approach
  - Implicit knowledge is made explicit
  - Side models for relations and constraints
- Operational approach
    - Use of EXPRESS

Perspectives

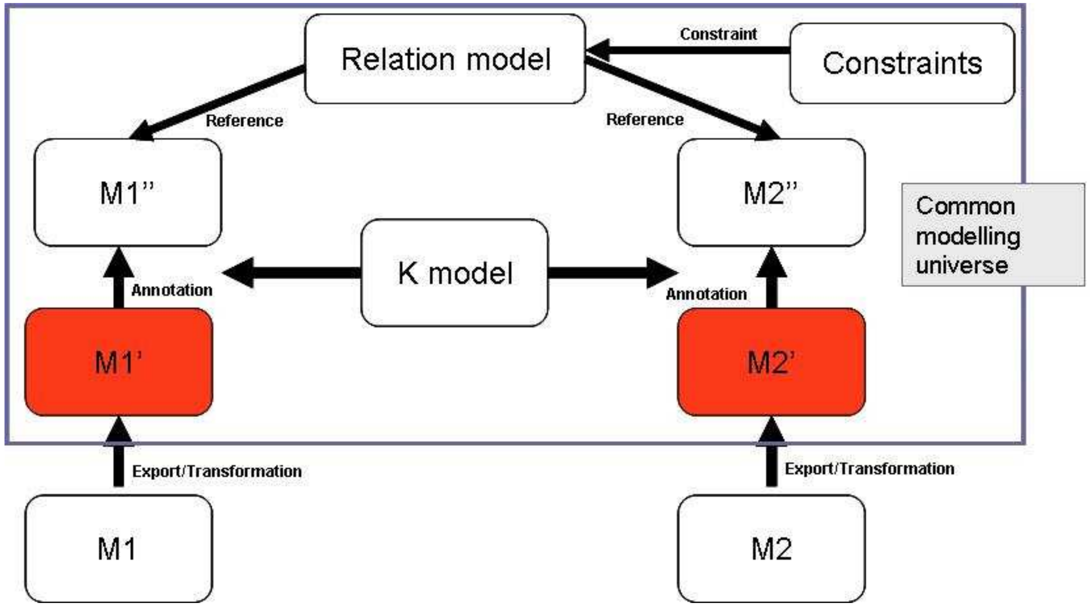
- Illustrate other cases and kind of relations
- Demonstrate other uses of formal semantic relation
  - To develop services exploiting formal relation
    - Discover relation using K annotation
    - Search into models
    - Check models consistency
- Study the impact of source models evolution
- Scalability
  - More than two models?
  - More than two languages?

Thank you

Step by Step

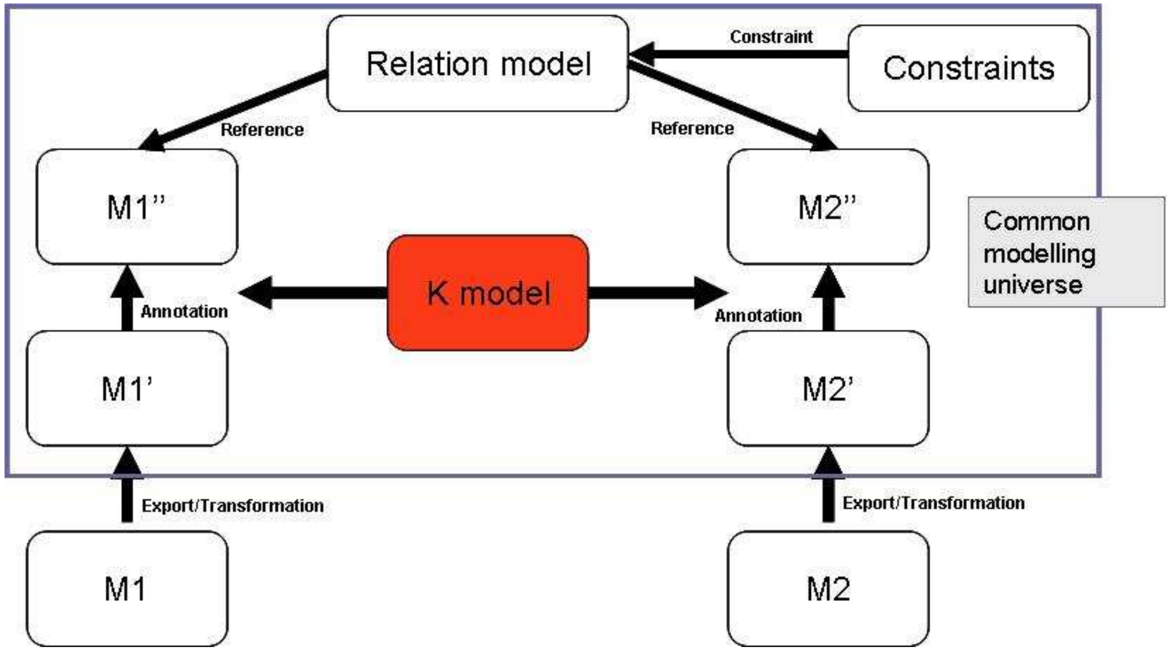
Step 1 (Export to a common modeling formal language)

-Need of meta-model of MMi for the relevant concepts involved in heterogeneity reduction
-Models Mi become instances of MMi



Step 2 (Formal side knowledge model)

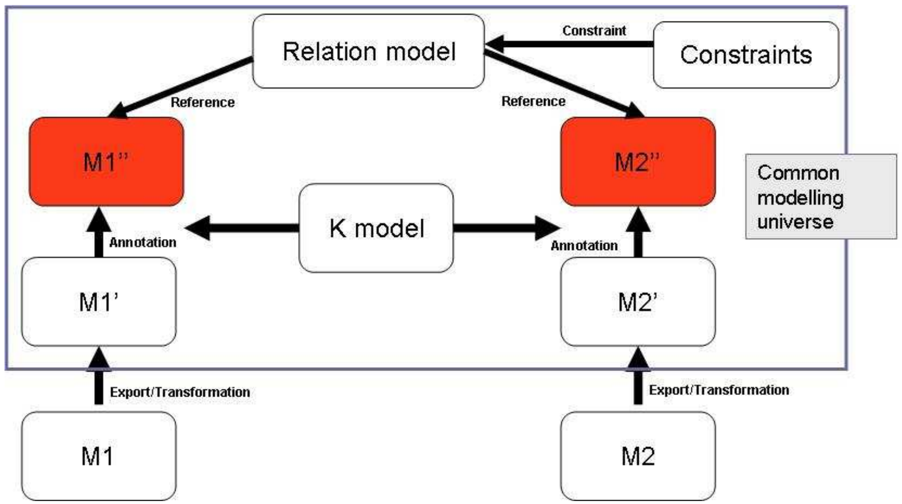
-Domain knowledge is explicited
-Each concept is uniquely identified (URI)

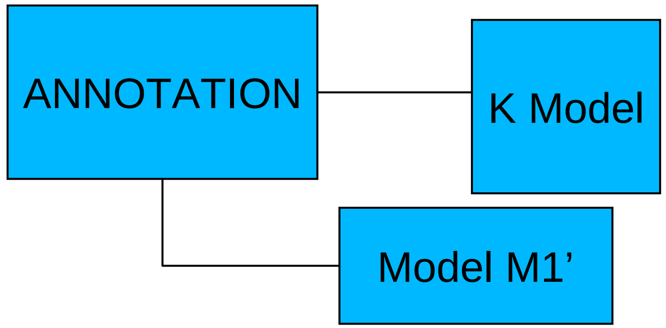


Step 3 (Knowledge based model annotation)

- Define semantic links between Models elements (Classes And/Or Instances) and Knowledge concepts

- Each relevant model element is annotated with an URI from K Model (1-1, 1-n, n-1, n-n)



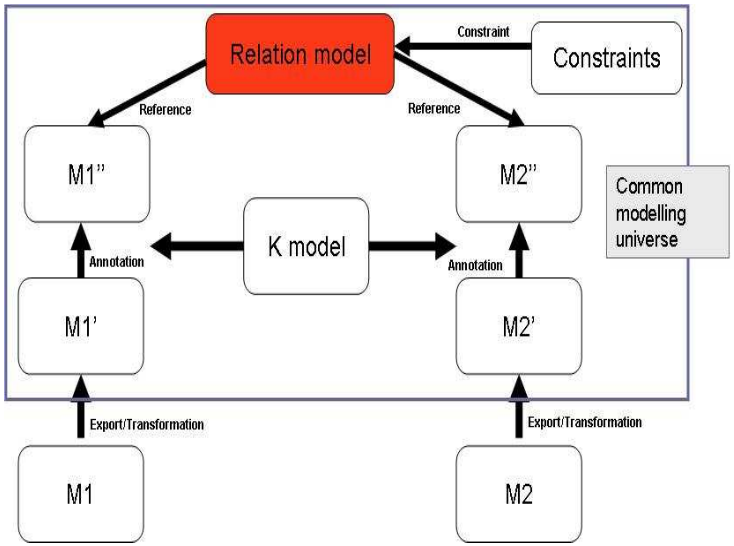


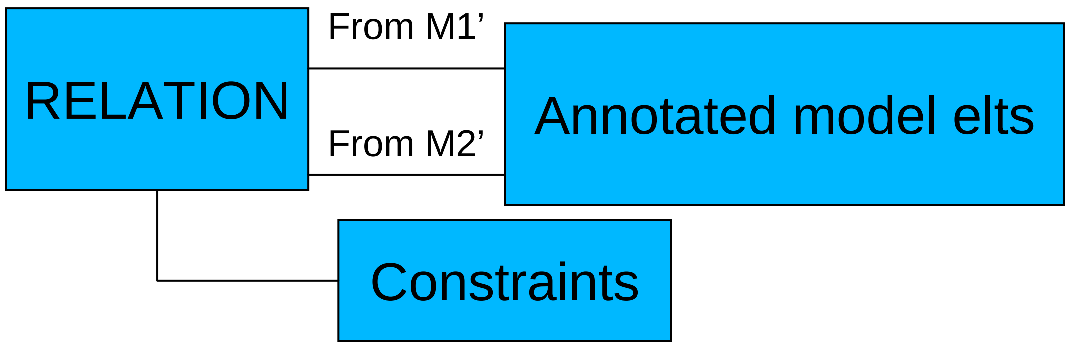
Step by Step

Step 4 (Inter-models relations)

- Need for relationship model
- Structural relations
- Constraint on these relations

} Exploit K Model





Step 5 (express constraints over the relations)

-This is goal oriented
- -To guarantee inter-model consistency
- -To check inter-model properties