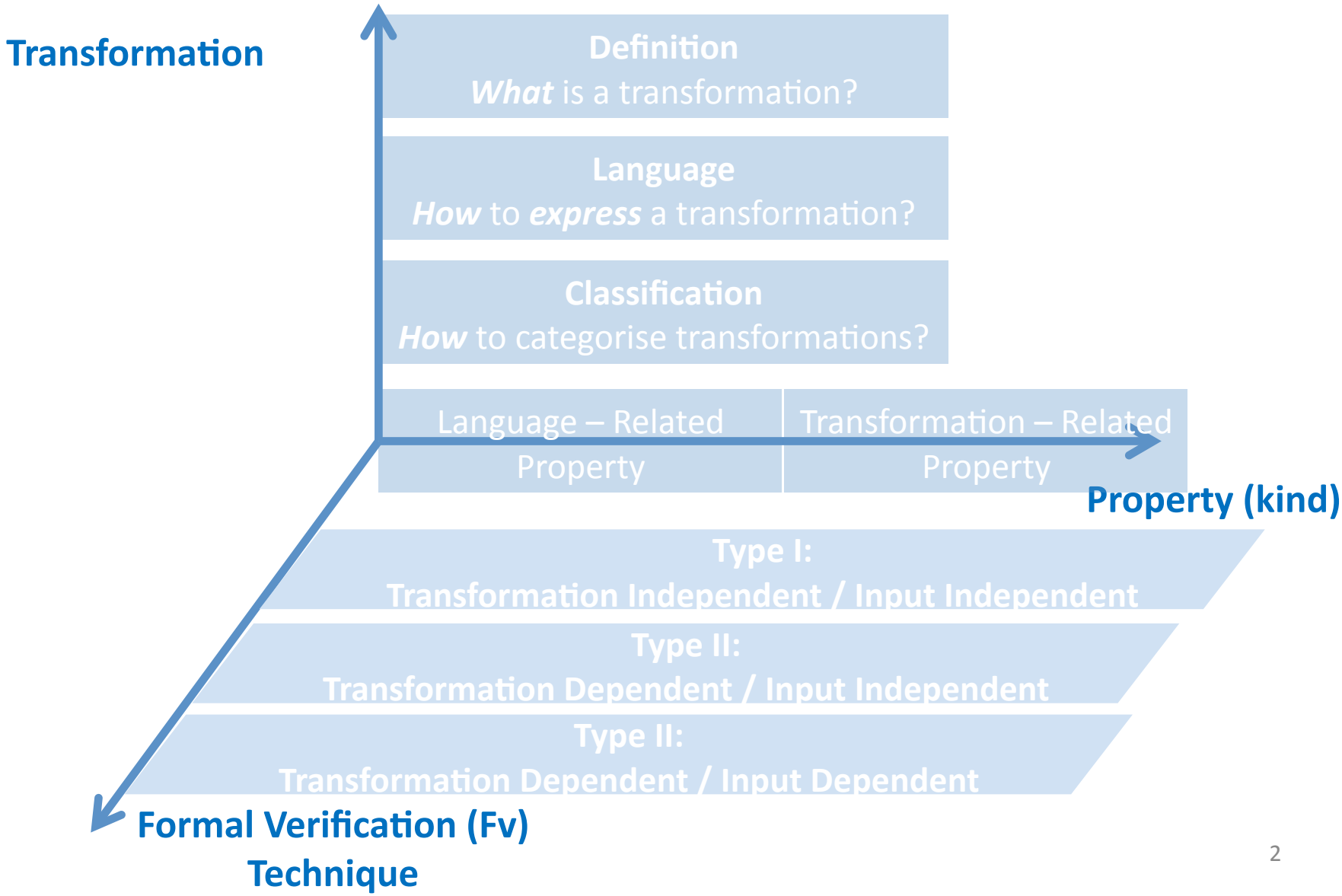
Towards a Model Transformation Intent Catalog

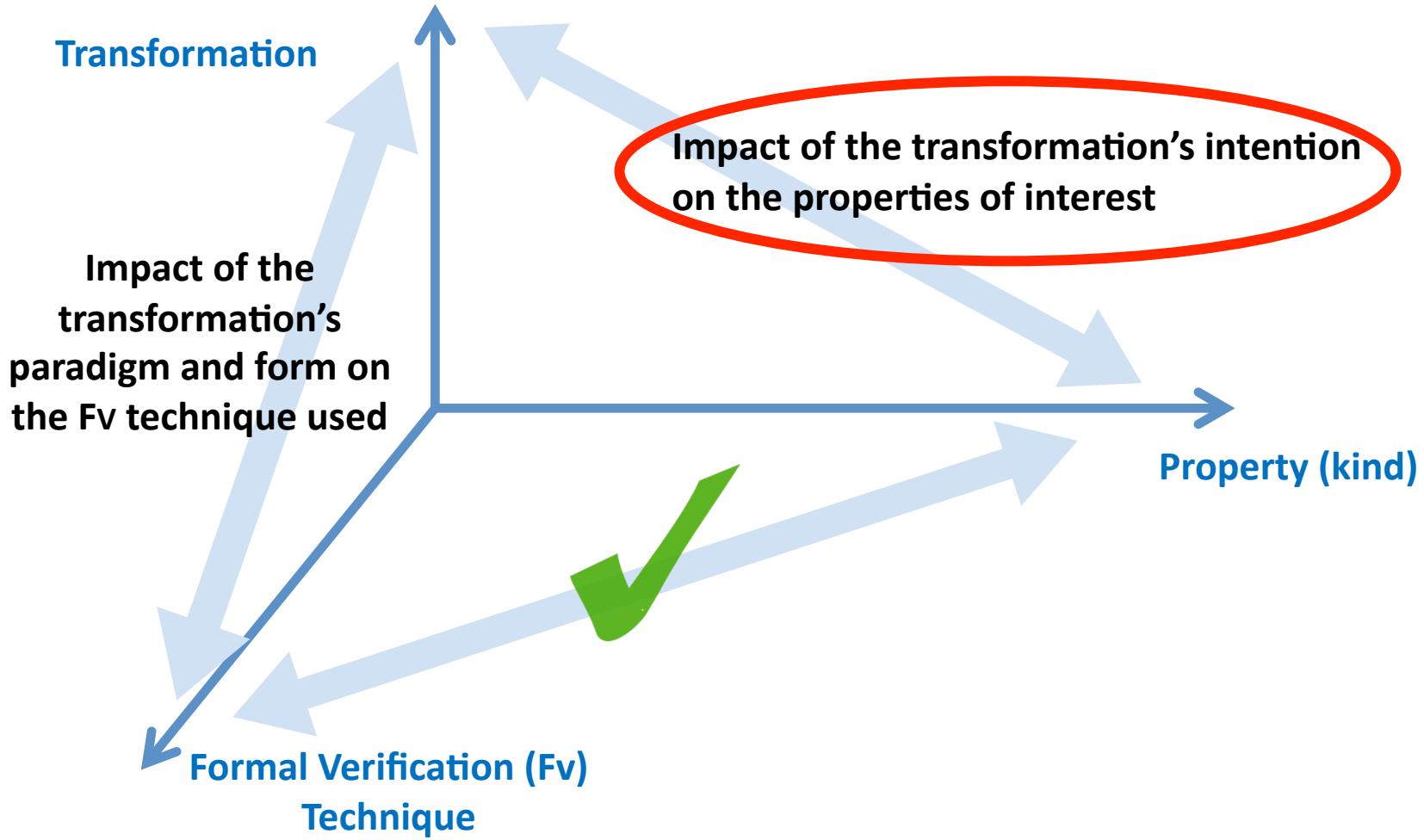
Moussa Amrani (Luxembourg)
Jürgen Dingel (Queens)
Leen Lambers (Potsdam / Hasso Plattner)
Levi Lúcio (McGill)

Rick Salay (Toronto)
Gehan Selim (Queens)
Eugene Syriani (Alabama)
Manuel Wimmer (Málaga)

Verification of Model Transformations



Verification of Model Transformations



M. Amrani, L. Lúcio, G. Selim, B. Combemale, J. Dingel, H. Vangheluwe, Y. Le Traon, and J. R. Cordy, "*A Tridimensional Approach for Studying the Formal Verification of Model Transformations*," in VOLT'2012@ICST

Presentation Outline

- Case study
- Transformation intent catalog
- Formalising transformation intents
- Two examples
  - The *Analysis* intent
  - The *Query* intent
- Conclusion

The NECSIS Project

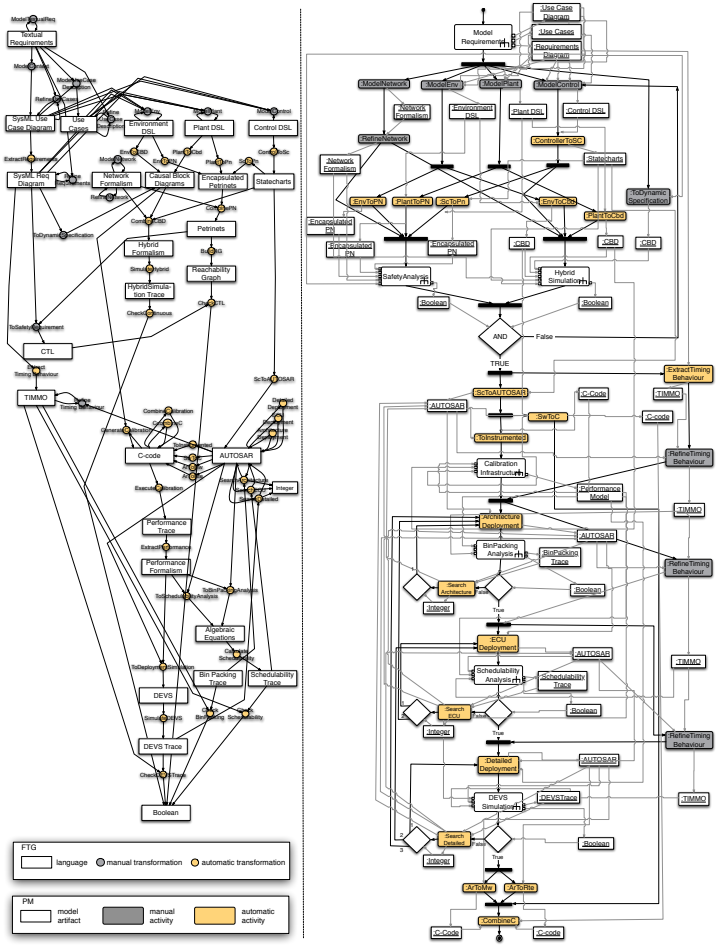
"NECSIS is focused on the advancement of a software methodology, called Model-Driven Engineering (MDE), that can yield dramatic improvements in software-developer productivity and product quality. "

Collaboration between:

McMaster University, University of Waterloo, University of British Columbia, CRIM (Centre de recherche informatique de Montréal), McGill University, Queen's University, University of Toronto, University of Victoria and

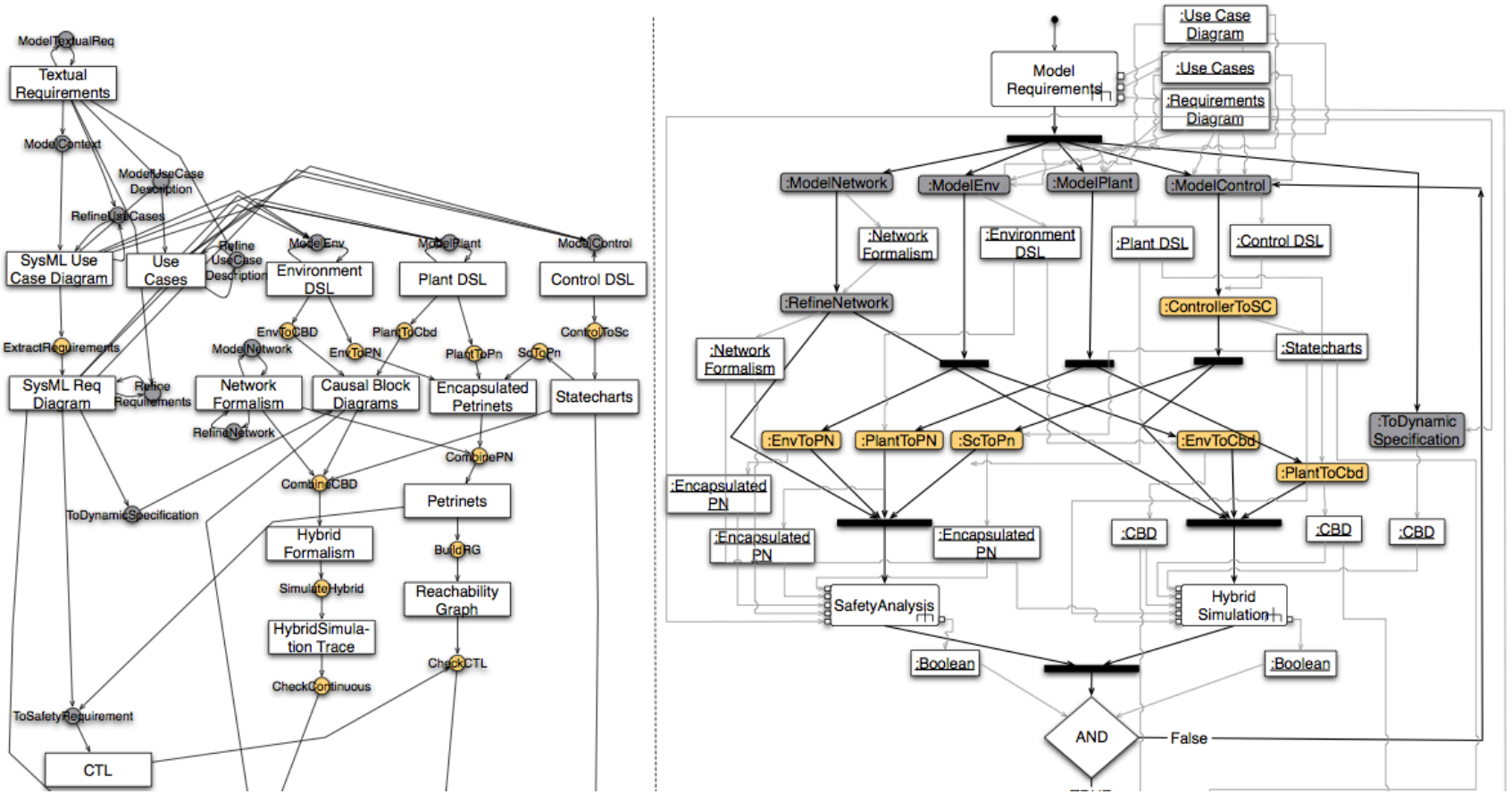
General Motors of Canada, IBM Canada and Malina Software.

Case Study: MDE based development of control software for Automobiles' Power Windows



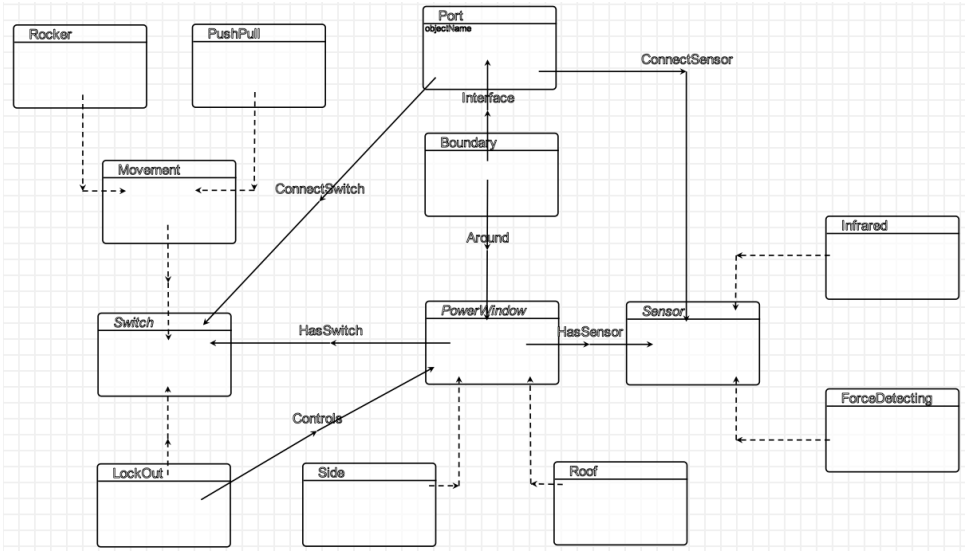
L. Lúcio, J. Denil, and H. Vangheluwe, "*An Overview of Model Transformations for a Simple Automotive Power Window*," McGill University, Tech. Rep. SOCS-TR-2012.1, 2012.

Transformation Chains



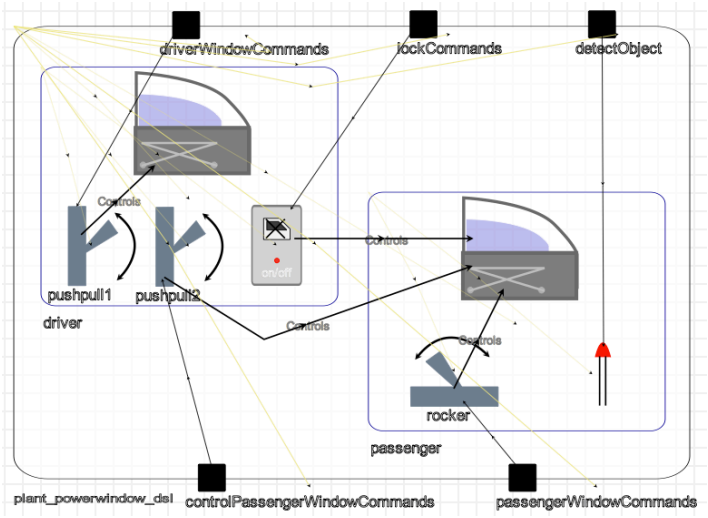
FTG (Formalism Transformation Graph) + **PM (Process Model)**, UML 2.0 Activity Diagrams

Transformation Chains



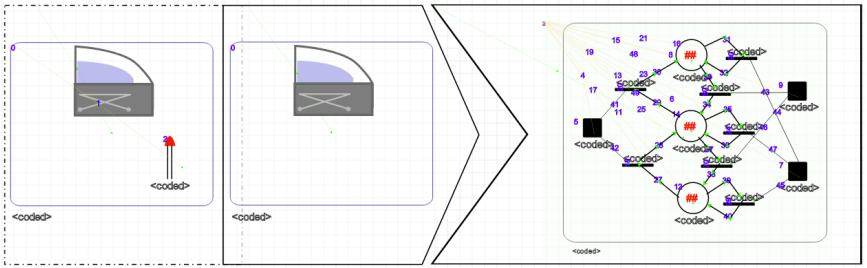
Plant DSL Formalism

Plant DSL Model

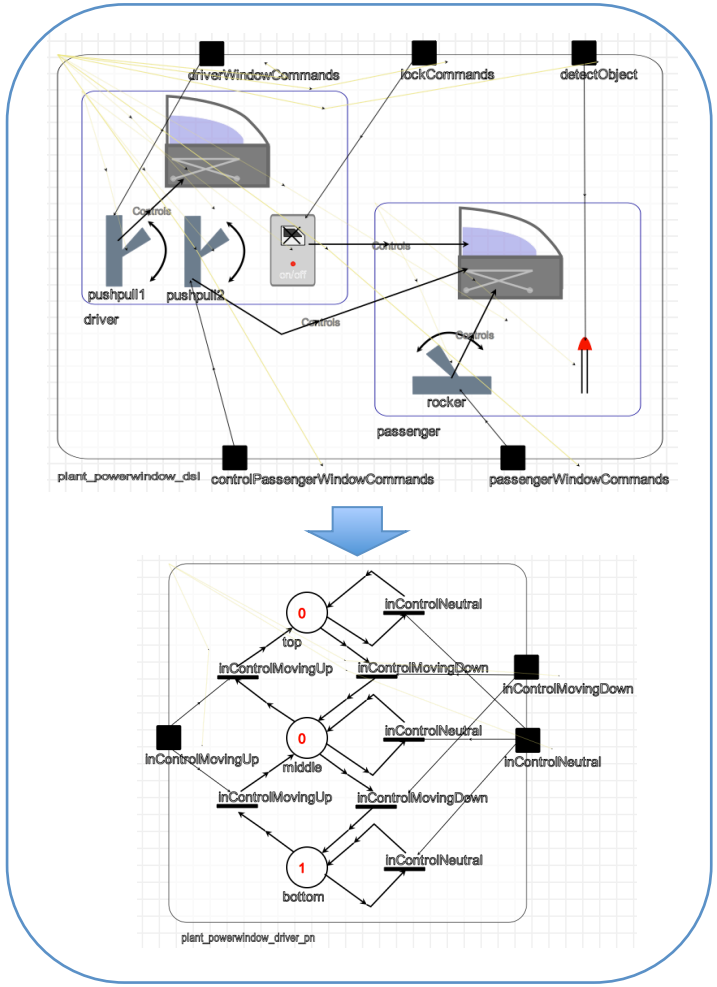


Transformation Chains

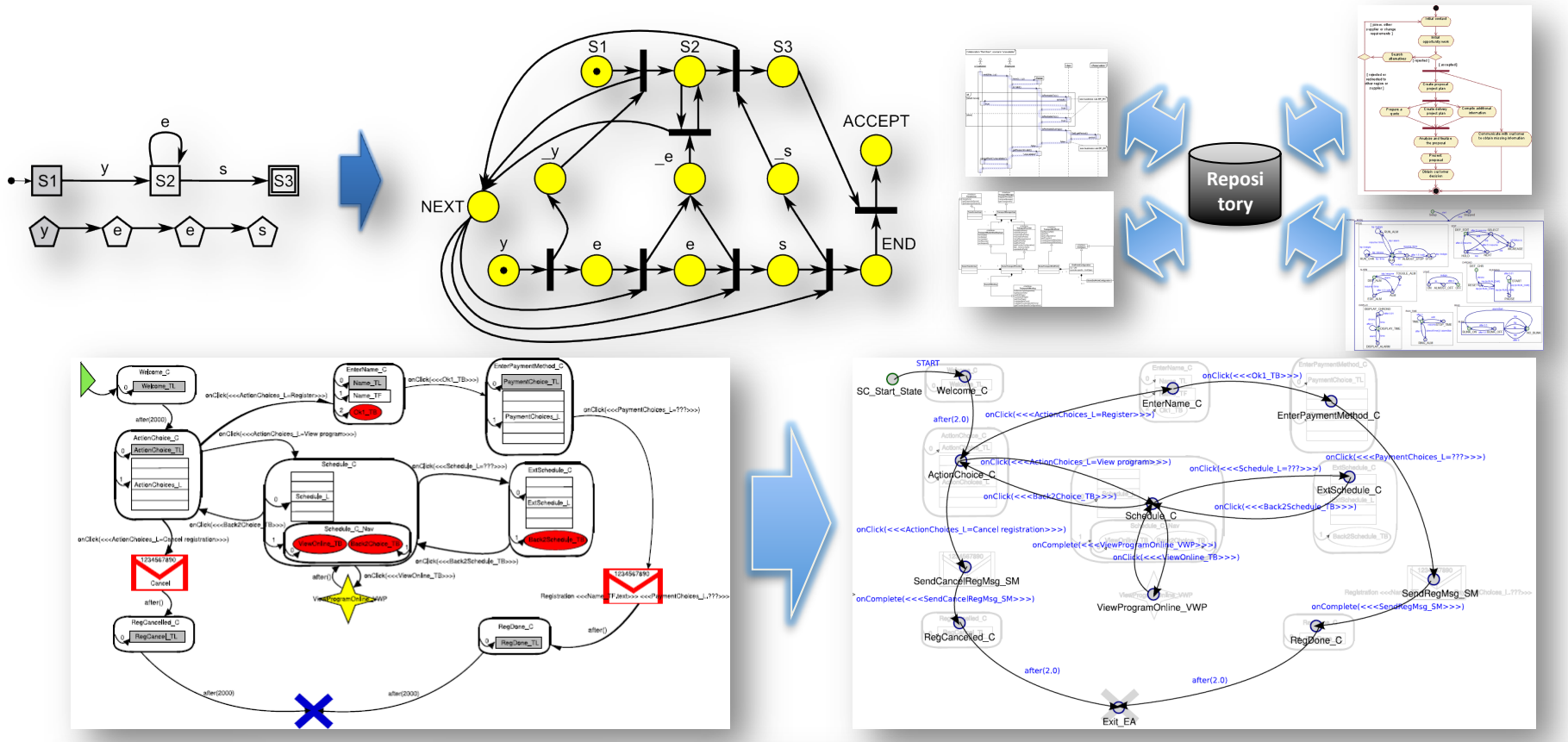
Transformation Definition (1 rule)



Transformation Execution



Model Transformations



Intents of Model Transformations

"A model transformation is *an automated manipulation of models according to a specific intent.*"

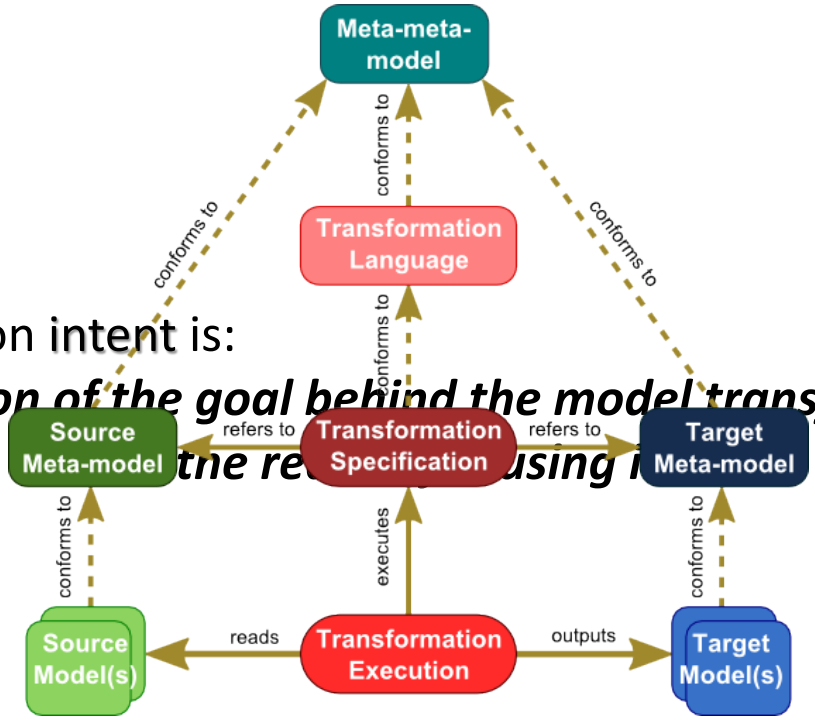
E. Syriani, "A Multi-Paradigm Foundation for Model Transformation Language Engineering," Ph.D. Thesis, McGill University, 2011

Working Definition

A model transformation intent is:

"a description of the goal behind the model transformation

the re using



Some Transformation Intents

- T. Mens and P. Van Gorp, “A Taxonomy Of Model Transformation”. ENTCS: 152, pp. 125–142, 2006.

- K. Czarnecki and S. Helsen, “Feature-Based Survey of Model Transformation Approaches”. IBM *Systems Journal:* 45(3), pp. 621–645, 2006.

- E. Syriani, “A Multi-Paradigm Foundation for Model Transformation Language Engineering”. Ph.D. Thesis, McGill University, 2011.

- M. Tisi, F. Jouault, P. Fraternali, S. Ceri, and J. Bézivin, “On the Use of Higher-Order Model Transformations”. ECMDA-FA, 2009, pp. 18–33.

Manipulation

- A model transformation performs a **manipulation** on a model.
- Simple atomic or bulk operations on a model:
  - **Add** an element to the model;
  - **Remove** an element from the model;
  - **Update** an element's properties;
  - **Access** an element or its properties.
- These primitive operations are know as the **CRUD operations** (Create, Read, Update, Delete)

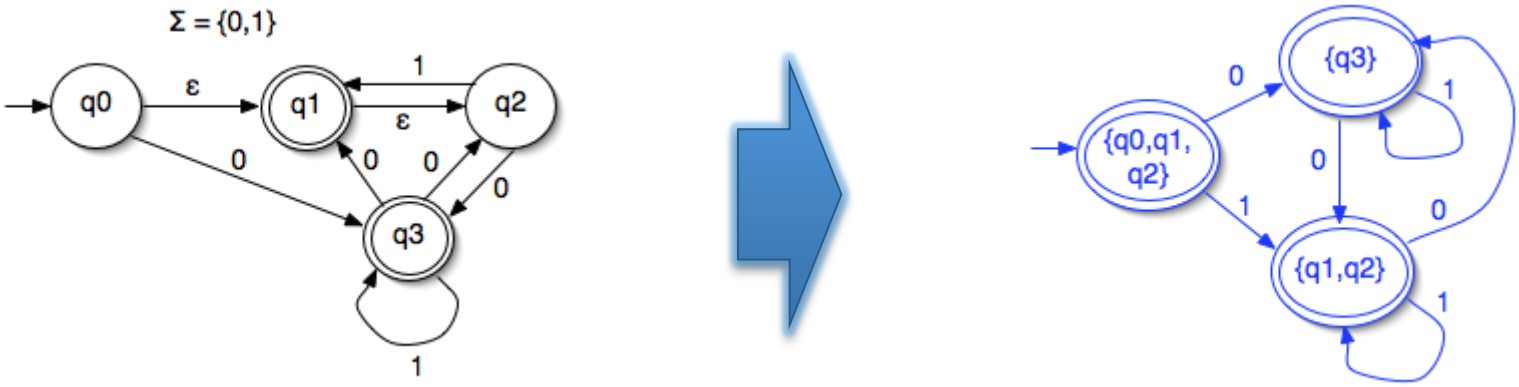
Query

- A query requests some information about a model M
- It produces a view of M
- Restrictive view: Reveal a proper subset of M (all, none, some)
  - Retrieve all cycles in a causal block diagram
  - Show only classes/associations of a class diagram
- Aggregated view: Restriction of M modifying some of its properties
  - Get the average of all costs per catalogue product in a relational database schema
  - In a hierarchical model, show top-level elements only, with an extra attribute denoting the number of sub-elements

Refinement

- Transform from a higher level specification to a lower level description
  - M1 refines M2 if M1 can answer all questions that M2 can

NFA to DFA



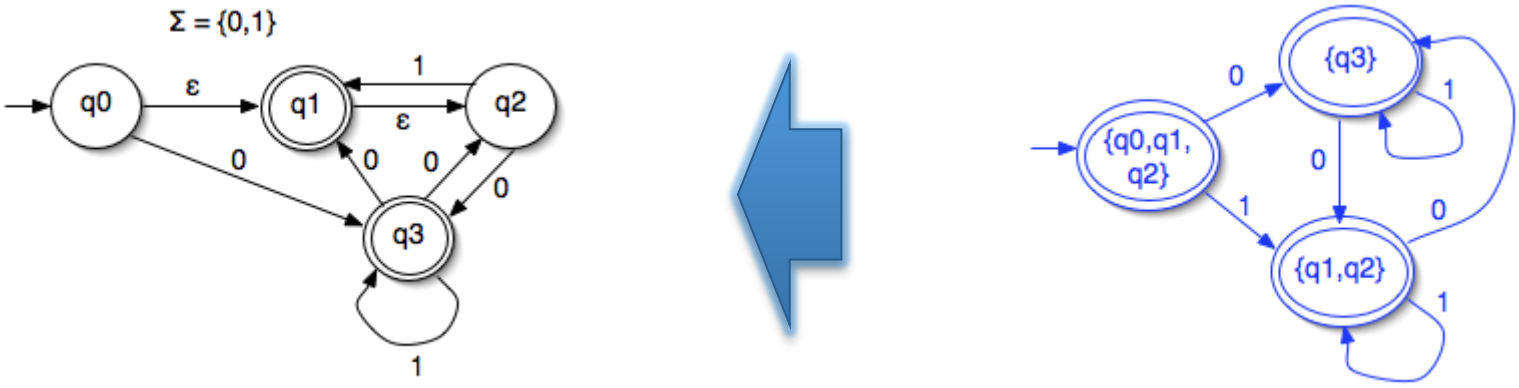
Non-deterministic state automata (NFA)

Deterministic state automata (DFA)

Abstraction

- Inverse of refinement
  - M1 refines M2 then M2 is an abstraction of M1

DFA to NFA



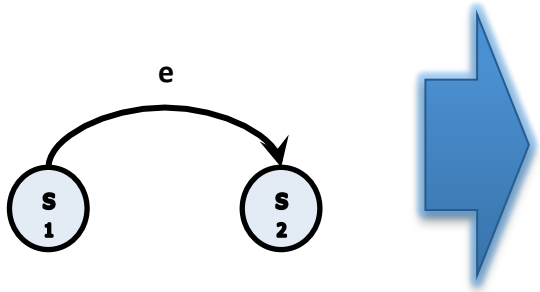
Non-deterministic state automata (NFA)

Deterministic state automata (DFA)

Synthesis

- Model is synthesized into a well-defined language format that can be stored, such as in serialization
- Model-to-code generation
  - Case where the target language is source code in a programming language

Statecharts to Python Compiler



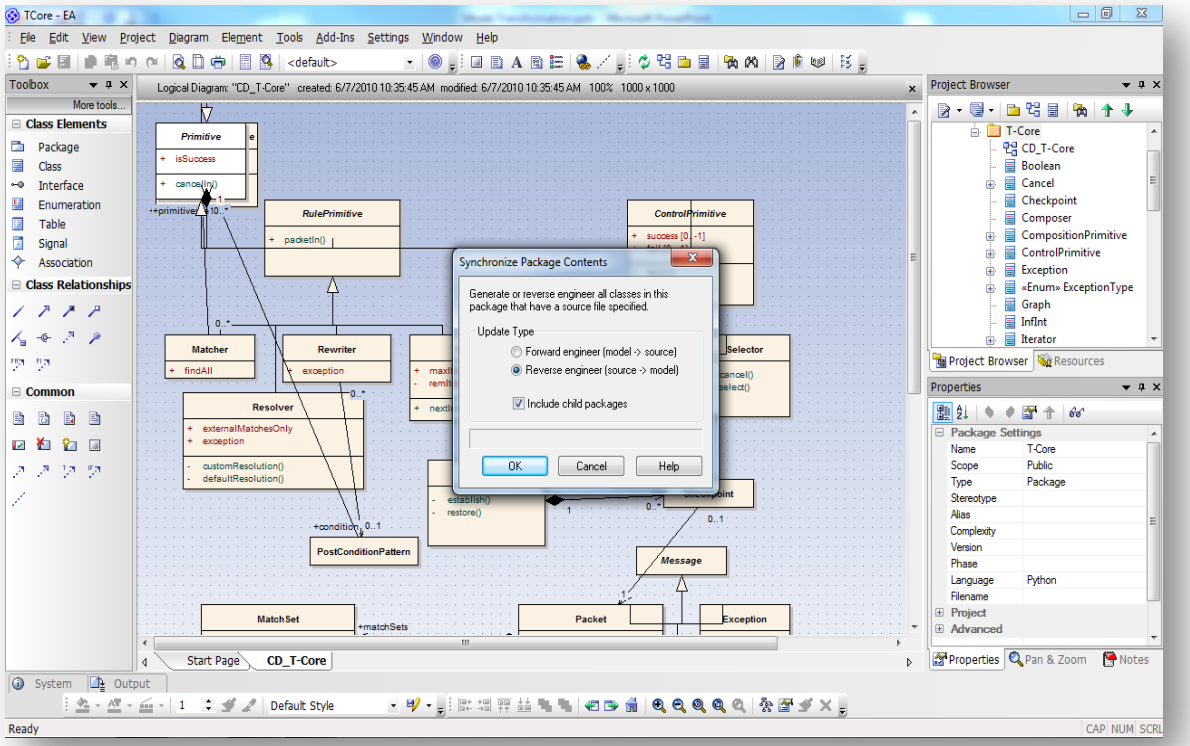
Statecharts model

```
if e == 0:    # event "e"
    if table[1] and self.isInState(1) and self.testCondition(3):
    if (scheduler == self or scheduler == None) and table[1]:
        self.runActionCode(4)            # output action(s1)
        self.runExitActionsForStates(-1)
        self.clearEnteredStates()
        self.changeState(1, 0)
        self.runEnterActionsForStates(self.StatesEntered, 1)
        self.applyMask(DigitalWatchStatechart.OrthogonalTable[1], table)
        handled = 1
    if table[0] and self.isInState(0) and self.testCondition(4):
        if (scheduler == self or scheduler == None) and table[0]:
            self.runActionCode(5)        # output action(s2)
            self.runExitActionsForStates(-1)
            self.clearEnteredStates()
            self.changeState(0, 0)
            self.runEnterActionsForStates(self.StatesEntered, 1)
            self.applyMask(DigitalWatchStatechart.OrthogonalTable[0], table)
            handled = 1
```

Generated Python code

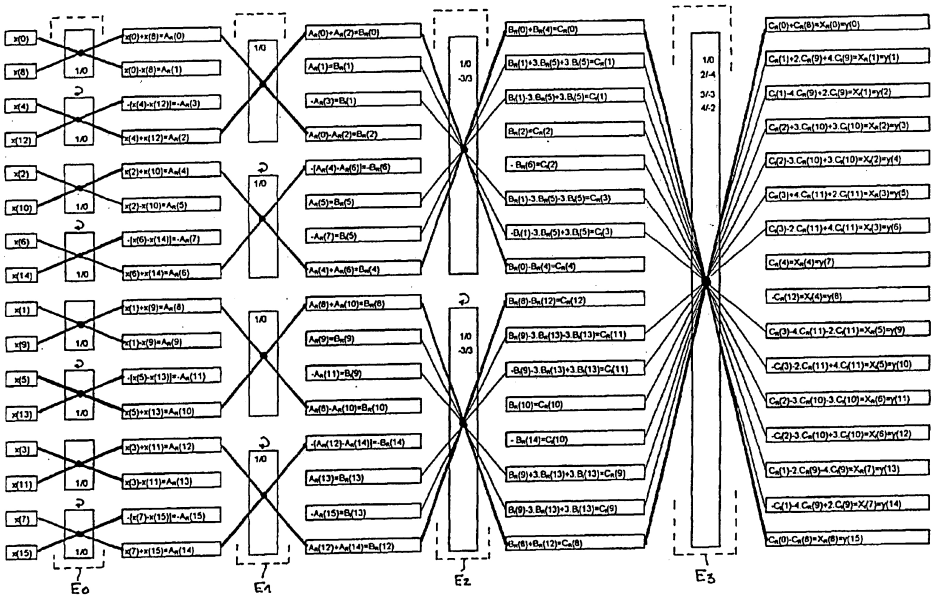
Reverse Engineering

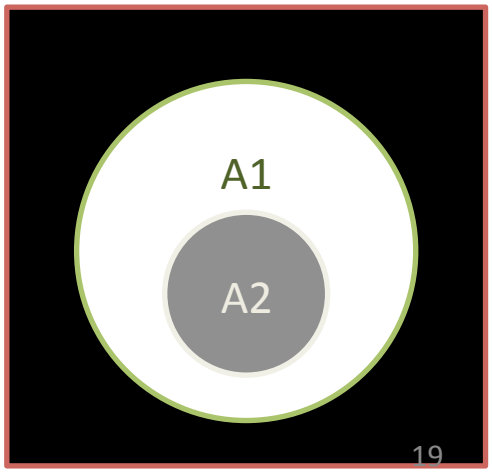
- Inverse of synthesis: extracts higher level specifications from lower level ones.
  - UML class diagrams can be generated from Java with Fujaba



Approximation

- Refinement with respect to negated properties
  - M1 approximates M2 if M1 negates the answer to all questions that M1 negates
- In practice, M2 is an idealization of M1 where an approximation is only extremely likely

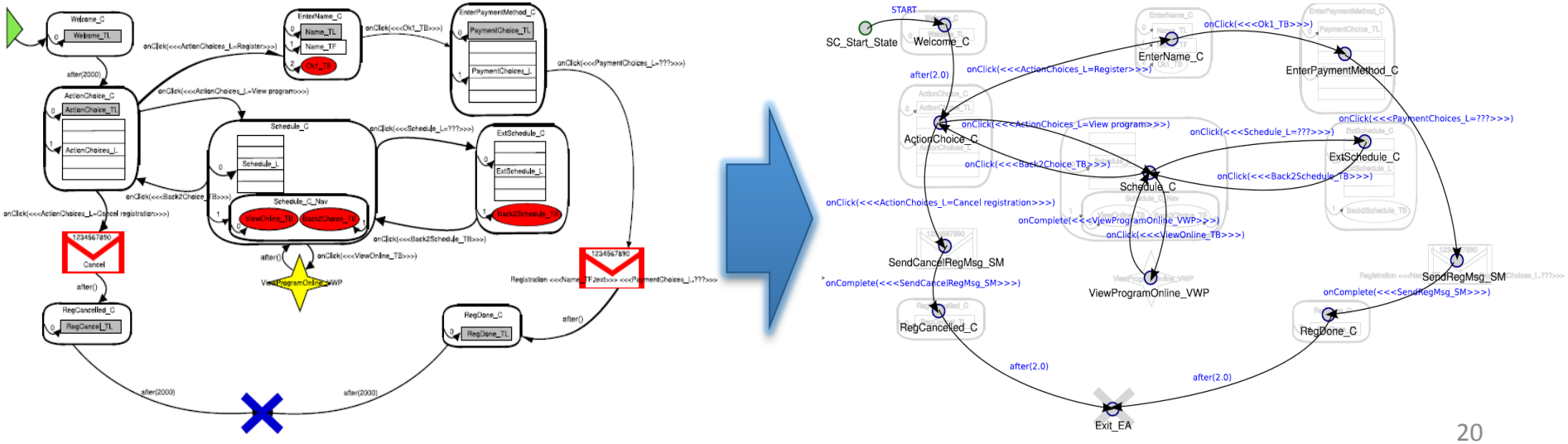




Translational semantics

- The semantics of the source formalism is given in terms of the semantics of the target formalism.
- Semantic mapping function of the original language defined by a MT that translates any of its instances to a valid instance of the reference formalism with well-defined semantics.
- Inter-formalism transformation (a.k.a. m2m transformation)

PhoneApps To Statecharts

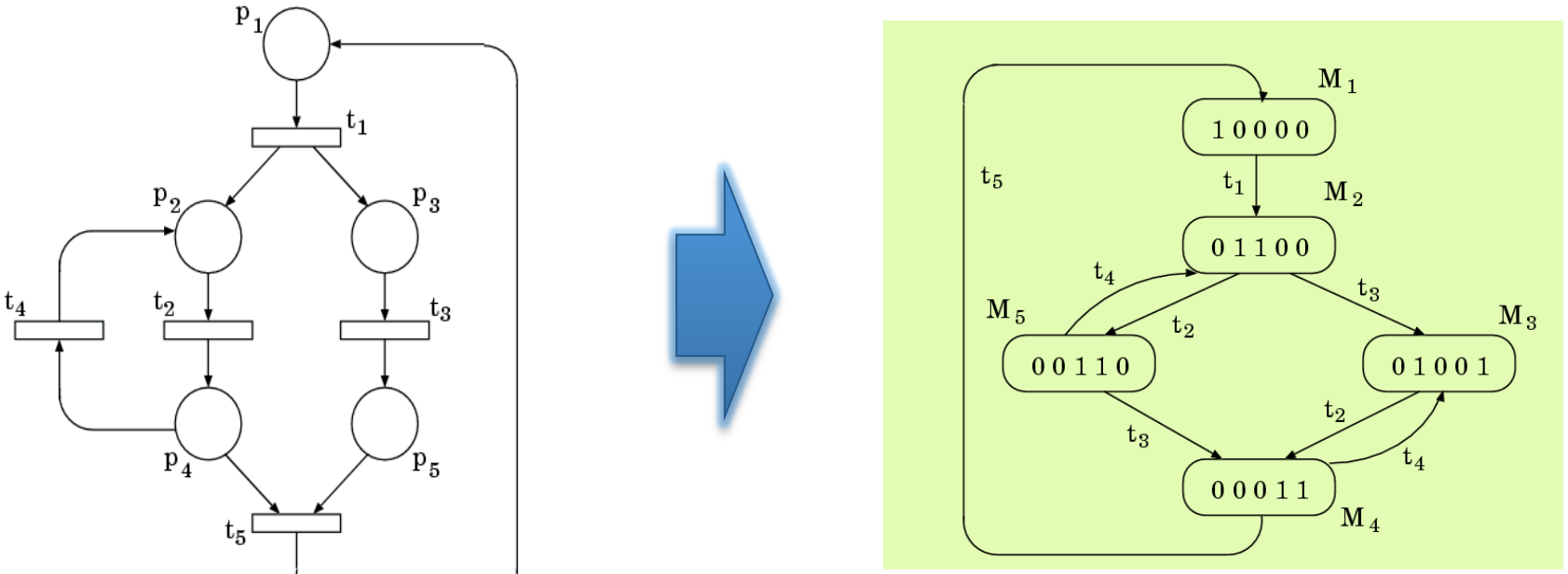


PhoneApps DSM of a conference registration mobile application

Representation of the model in Statecharts

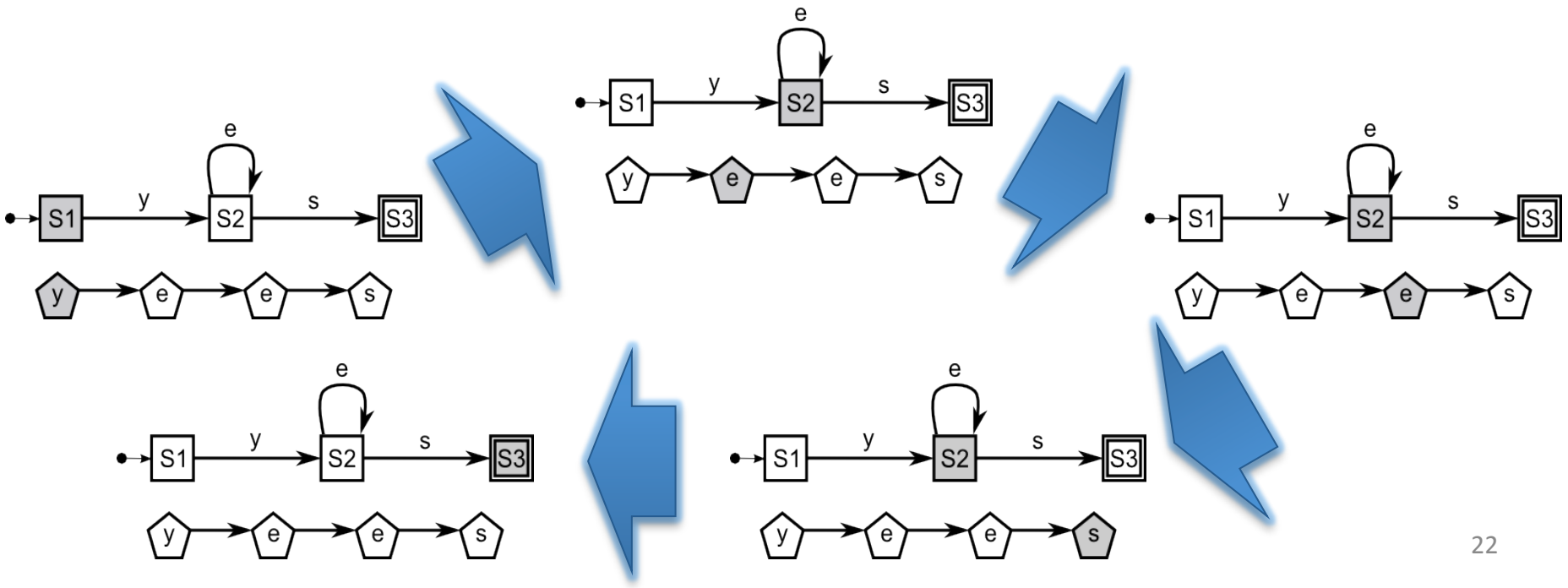
Analysis

- Map a modeling language to a formalism that can be analysed more appropriately than the original language
  - The target language is typically a formal language with known analysis techniques



Simulation

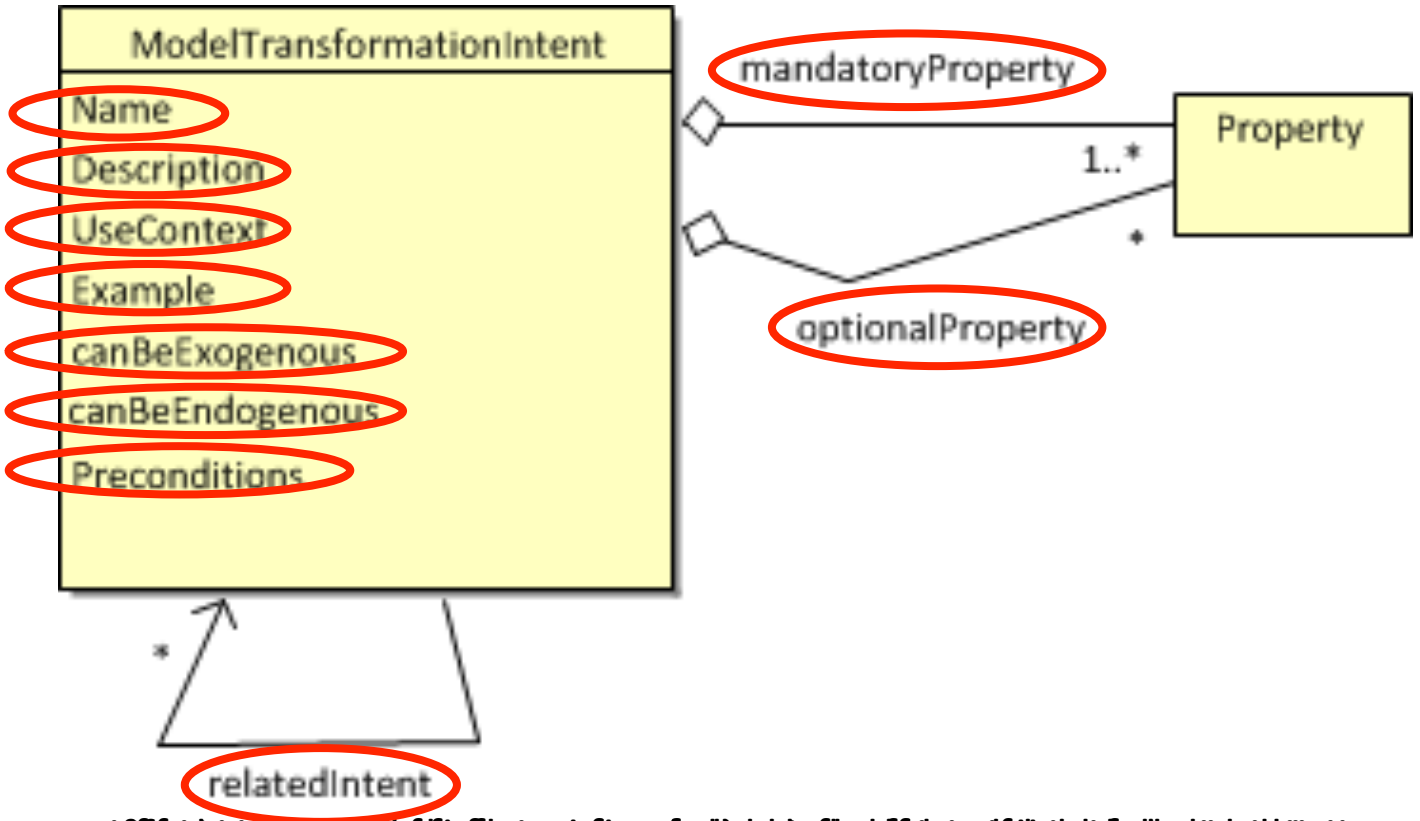
- Update the state of the model
- In this case, the source and target meta-models are identical.
- Moreover, the target model is an “updated” version of the source model: no new model is created



More Intents

Normalization	Aims to decrease the syntactic complexity of models by translating complex language constructs into more primitive constructs, which results in a canonical form of a model.
Rendering	Assignment of a concrete representation to each abstract syntax element or group of elements, as long as a meta-model of the concrete syntax is defined explicitly. Furthermore, multiple concrete syntaxes can be assigned to a single abstract syntax.
Model Generation	The meta-model of a language can be defined by a graph grammar. The grammar execution leads to model transformations able to generate all possible instances of the language.
Migration	transformation from a software model written in one language or framework into another language, keeping the models at the same level of abstraction.
Optimization	aims at improving operational qualities of models such as scalability and efficiency.
Refactoring	Restructuring that changes the internal structure of the model to improve certain quality characteristics without changing its observable behavior.
Composition-Merging	Integrates models that have been produced in isolation into a compound model. Typically, each isolated model represents a concern which may overlap. It creates a new model such that every element from each model is present exactly once in the merged model.
Composition-Weaving	Composition that creates correspondence links between overlapping entities.
Synchronization	Integrates models that have evolved in isolation but that are subject to global consistency constraints. It requires that changes are propagated to the models that are being integrated.

Formalising Intent



A description of the intent

i.e., what problems can it be used to solve?

The Analysis intent

T. Kühne, G. Mezei, E. Syriani, H. Vangheluwe, and M. Wimmer, "Systematic Transformation Development," ECEASST, vol. 21, 2009

J. de Lara and G. Taentzer, "Automated Model Transformation and its Validation Using AToM3 and AGG," in *Diagrams,* 2004, pp. 182–198

B. König and V. Kozioura, "Augur 2–A New Version of a Tool for the Analysis of Graph Transformation Systems," *Electronic Notes in Theoretical Computer Science (ENTCS),* vol. 211, pp. 201–210, 2008

D. Varro , S. Varro-Gyapay, H. Ehrig, U. Prange, and G. Taentzer, "Termination Analysis of Model Transformations by Petri Nets," *Int. Conference on Graph Transformations,* pp. 260–274, 2006

A. Narayanan and G. Karsai, "Verifying Model Transformations by Structural Correspondence," ECEASST, vol. 10, 2008

A. Narayanan and G. Karsai, "Towards Verifying Model Transformations," ENTCS, vol. 211, pp. 191–200, 2008

J. Rivera, E. Guerra, J. de Lara, and A. Vallecillo, "Analyzing Rule- Based Behavioral Semantics of Visual Modeling Languages with Maude," *Software Language Engineering,* pp. 54–73, 2009

B. Schätz, F. Hölzl, and T. Lundkvist, "Design-Space Exploration Through Constraint-Based Model-Transformation," in *Engineering of Computer Based Systems Workshop (ECBS),* 2010, pp. 173–182

The Analysis intent: attributes

Name	Analysis

The Analysis intent: attributes

Name	Analysis
Description	To indirectly analyse a property of the input model by running the analysis algorithm on the transformation's output model

The Analysis intent: attributes

Name	Analysis
Description	To indirectly analyse a property of the input model by running the analysis algorithm on the transformation's output model
UseContext	Need to analyse models that are not analysable in the transformation's input language, or are more efficiently analysable in the transformation's output language

The Analysis intent: attributes

Name	Analysis
Description	To indirectly analyse a property of the input model by running the analysis algorithm on the transformation's output model
UseContext	Need to analyse models that are not analysable in the transformation's input language, or are more efficiently analysable in the transformation's output language
Example	Transforming graph rewriting systems into Petri Nets to analyse them for termination (*e.g. Varró et al, ICGT 2006*)

The Analysis intent: attributes

Name	Analysis
Description	To indirectly analyse a property of the input model by running the analysis algorithm on the transformation's output model
UseContext	Need to analyse models that are not analysable in the transformation's input language, or are more efficiently analysable in the transformation's output language
Example	Transforming graph rewriting systems into Petri Nets to analyse them for termination (*e.g. Varró et al, ICGT 2006*)
canBeExogenous	True

The Analysis intent: attributes

Name	Analysis
Description	To indirectly analyse a property of the input model by running the analysis algorithm on the transformation's output model
UseContext	Need to analyse models that are not analysable in the transformation's input language, or are more efficiently analysable in the transformation's output language
Example	Transforming graph rewriting systems into Petri Nets to analyse them for termination (*e.g. Varró et al, ICGT 2006*)
canBeExogenous	True
canBeEndogenous	True (if transforming to a profile of the original language)

The Analysis intent: attributes

Name	**Analysis**
Description	To indirectly analyse a property of the input model by running the analysis algorithm on the transformation's output model
UseContext	Need to analyse models that are not analysable in the transformation's input language, or are more efficiently analysable in the transformation's output language
Example	Transforming graph rewriting systems into Petri Nets to analyse them for termination (*e.g. Varró et al, ICGT 2006*)
canBeExogenous	True
canBeEndogenous	True (if transforming to a profile of the original language)
Preconditions	1. Access to intended semantics, 2. The property of interest (that should be analysed) is defined 3. There exists an exhaustive (up to a given abstraction) automated method to analyse the property of interest on the transformation's output language 4. There exists a method to translate the property of interest onto the transformation's output language (if the transformation is exogenous)

The Analysis intent: associations

mandatoryProperty	1. Termination 2. Type correctness 3. Preservation of the property of interest (specialises *Property preservation)* 4. Analysis result can be mapped back onto the input model (specialises *Traceability)*

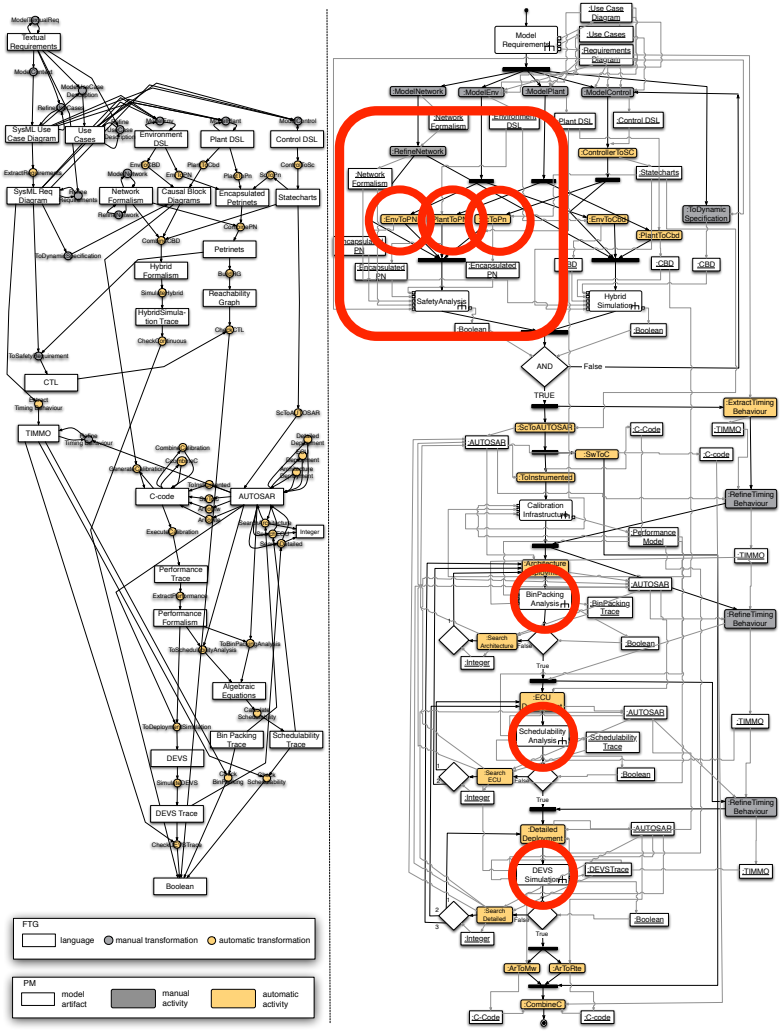
The Analysis intent: associations

mandatoryProperty	1. Termination 2. Type correctness 3. Preservation of the property of interest (specialises *Property preservation)* 4. Analysis result can be mapped back onto the input model (specialises *Traceability)*
optionalProperty	1. Readability of the transformation's output for debugging purposes 2. Semantics of the input language is formally defined (specialises *Mathematical underpinning)*

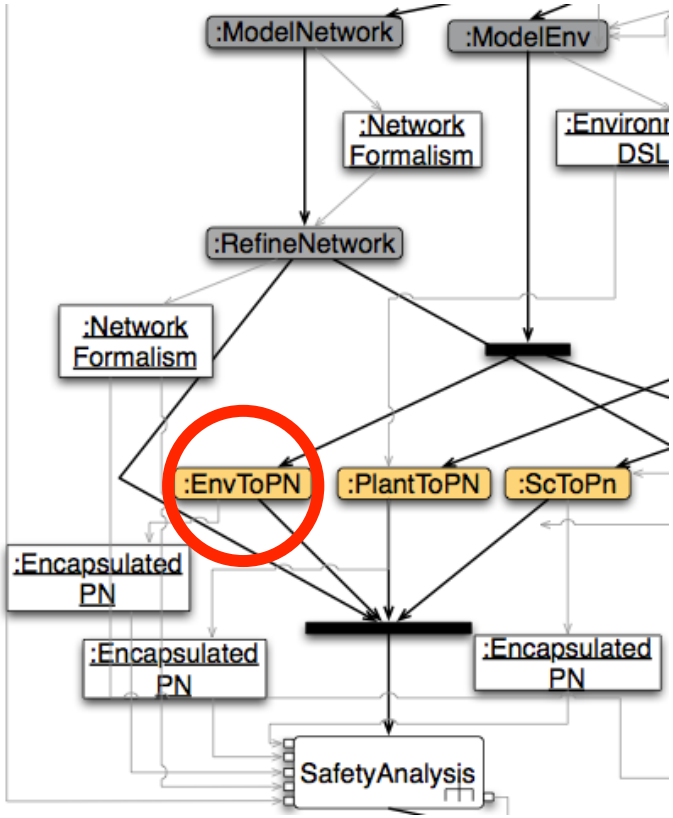
The Analysis intent: associations

mandatoryProperty	1. Termination 2. Type correctness 3. Preservation of the property of interest (specialises *Property preservation)* 4. Analysis result can be mapped back onto the input model (specialises *Traceability)*
optionalProperty	1. Readability of the transformation's output for debugging purposes 2. Semantics of the input language is formally defined (specialises *Mathematical underpinning)*
relatedIntent	Translational Semantics, Simulation

The Analysis Intent in the Power Window transformation chain



The Analysis Intent in the Power Window transformation chain



Build a Petri net representation of a specialised model of the passenger's interactions with the power window. Allows checking power window security requirements.

Satisfies preconditions 1,2,3, missing 4

"There exists a method to translate the property of Interest onto the transformation's output language"

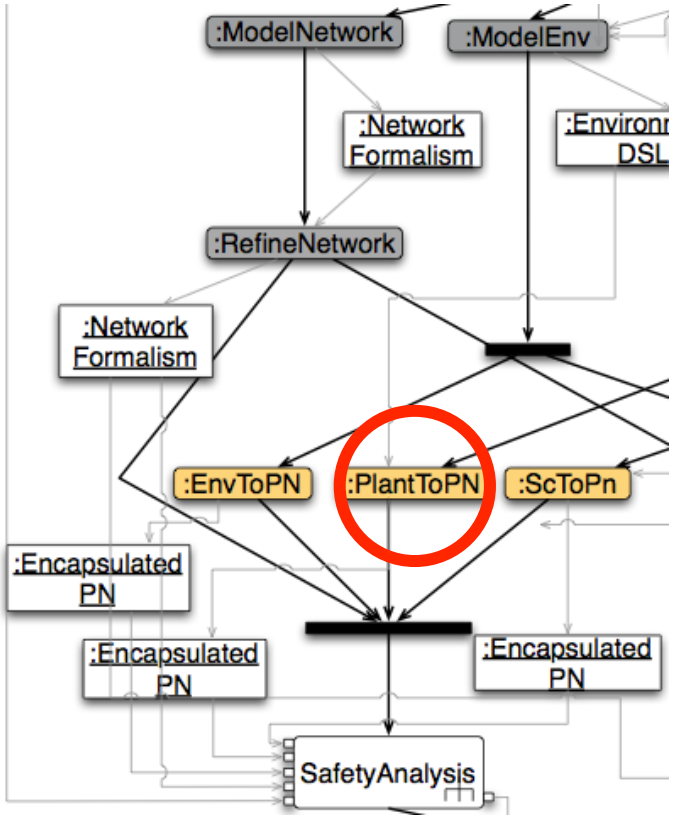
Satisfies properties 1,2,3, missing 4

"Analysis result can be mapped back onto the input model"

Satisfies no optional properties

The intent is *analysis*!

The Analysis Intent in the Power Window transformation chain



Build a Petri net representation of a specialised model of the power window physical configuration. Allows checking power window security requirements.

Satisfies preconditions 1,2,3, missing 4

"There exists a method to translate the property of Interest onto the transformation's output language"

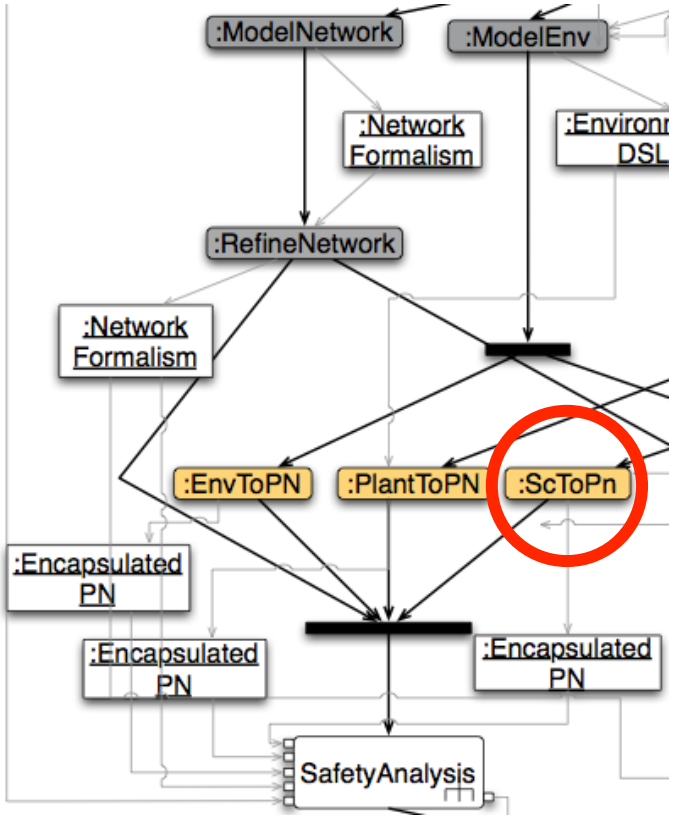
Satisfies properties 1,2,3, missing 4

"Analysis result can be mapped back onto the input model"

Satisfies no optional properties

The intent is *analysis*!

The Analysis Intent in the Power Window transformation chain



Build a Petri net representation of a specialised model of the power window control software. Allows checking power window security requirements.

Satisfies preconditions 1,2,3, missing 4

"There exists a method to translate the property of Interest onto the transformation's output language"

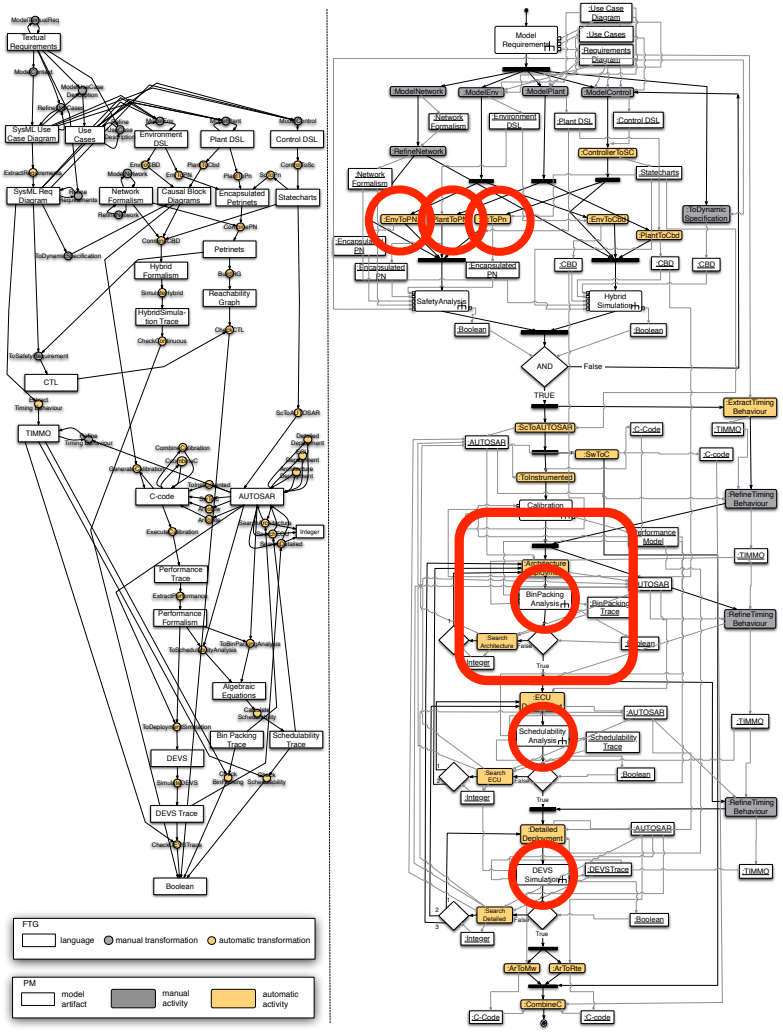
Satisfies properties 1,2,3, missing 4

"Analysis result can be mapped back onto the input model"

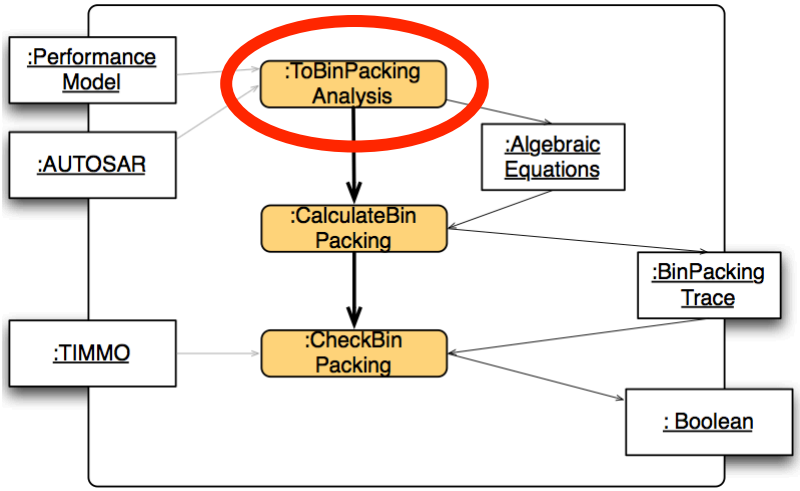
Satisfies optional properties 1,2

The intent is *analysis*!

The Analysis Intent in the Power Window transformation chain



The Analysis Intent in the Power Window transformation chain

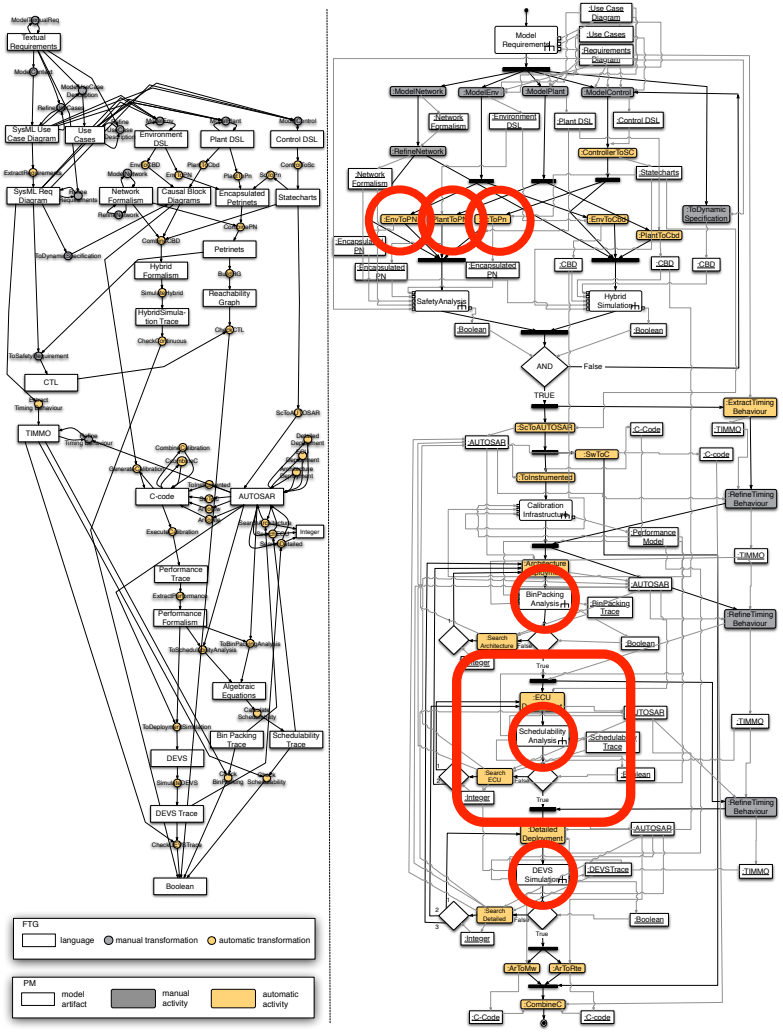


Build an equational algebraic representation of the dynamic behavior of the involved hardware components from an AUTOSAR specification. Allows checking processor load distribution.

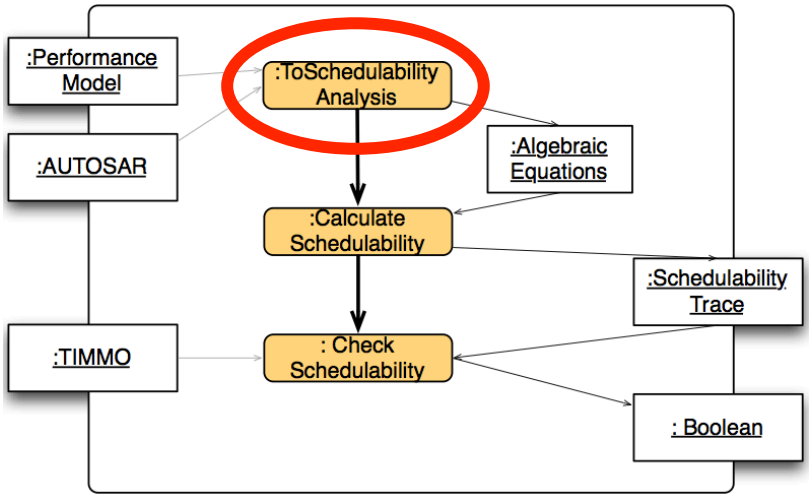
Satisfies preconditions 1,2,3,4
Satisfies properties 1,2,3,4
Satisfies optional properties 1

The intent is *analysis*!

The Analysis Intent in the Power Window transformation chain



The Analysis Intent in the Power Window transformation chain

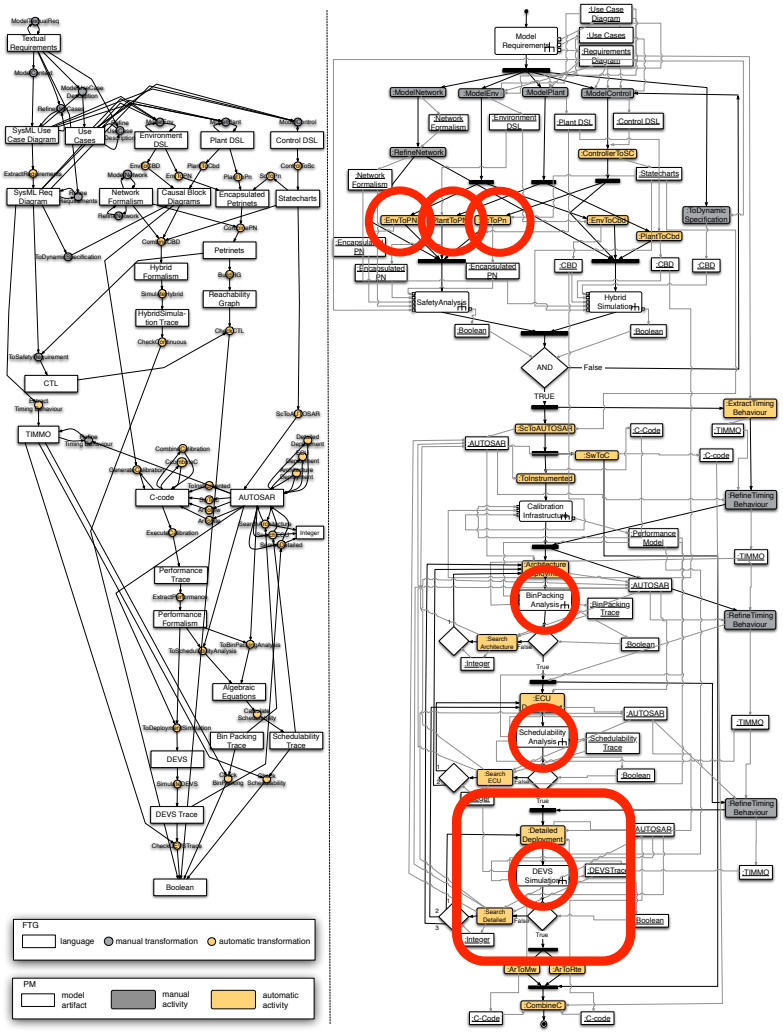


Build an equational algebraic representation of the dynamic behavior of the involved hardware and software components from an AUTOSAR specification. Allows checking software response times.

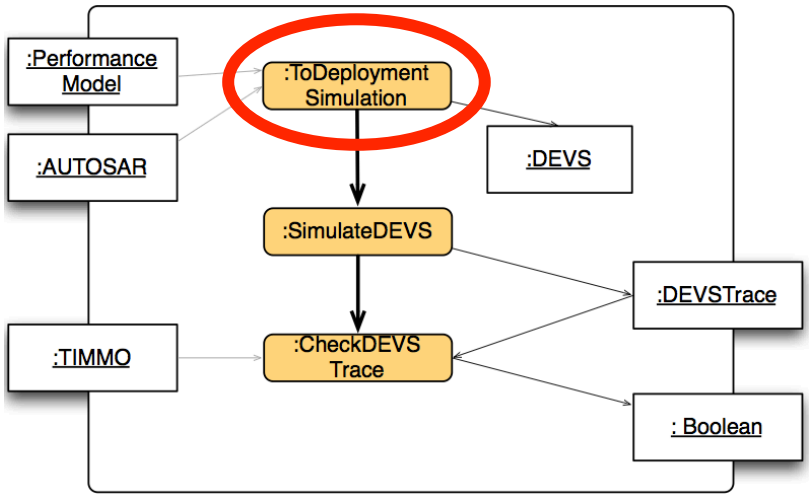
Satisfies preconditions 1,2,3,4
Satisfies properties 1,2,3,4
Satisfies optional properties 1

The intent is *analysis*!

The Analysis Intent in the Power Window transformation chain



The Analysis Intent in the Power Window transformation chain



Build a DEVS representation of the deployment solution to check for latency times, deadlocks and lost messages.

Satisfies preconditions 1,2,4 (missing 3)

"A verification method exists for analyzing the property of interest on the target language"

Satisfies properties 1,2,3,4

Satisfies optional properties 1

The intent is *NOT* analysis!

The Query intent: attributes

Name	Query (restrictive view)

The Query intent: attributes

Name	**Query (restrictive view)**
Description	To extract the unique submodel (the view) from a model that satisfies some criterion (the query)

The Query intent: attributes

Name	**Query (restrictive view)**
Description	To extract the unique submodel (the view) from a model that satisfies some criterion (the query)
UseContext	1. Want to extract the relevant part (view) of a model for a task or 2. Want to decompose a model to manage complexity

The Query intent: attributes

Name	**Query (restrictive view)**
Description	To extract the unique submodel (the view) from a model that satisfies some criterion (the query)
UseContext	1. Want to extract the relevant part (view) of a model for a task or 2. Want to decompose a model to manage complexity
Example	1. Extract the submodel that are immediate neighbours of a particular element 2. Extract the submodel of structural elements from a UML model 3. Model slicing

The Query intent: attributes

Name	**Query (restrictive view)**
Description	To extract the unique submodel (the view) from a model that satisfies some criterion (the query)
UseContext	1. Want to extract the relevant part (view) of a model for a task or 2. Want to decompose a model to manage complexity
Example	1. Extract the submodel that are immediate neighbours of a particular element 2. Extract the submodel of structural elements from a UML model 3. Model slicing
canBeExogenous	True

The Query intent: attributes

Name	**Query (restrictive view)**
Description	To extract the unique submodel (the view) from a model that satisfies some criterion (the query)
UseContext	1. Want to extract the relevant part (view) of a model for a task or 2. Want to decompose a model to manage complexity
Example	1. Extract the submodel that are immediate neighbours of a particular element 2. Extract the submodel of structural elements from a UML model 3. Model slicing
canBeExogenous	True
canBeEndogenous	True

The Query intent: attributes

Name	**Query (restrictive view)**
Description	To extract the unique submodel (the view) from a model that satisfies some criterion (the query)
UseContext	1. Want to extract the relevant part (view) of a model for a task or 2. Want to decompose a model to manage complexity
Example	1. Extract the submodel that are immediate neighbours of a particular element 2. Extract the submodel of structural elements from a UML model 3. Model slicing
canBeExogenous	True
canBeEndogenous	True
Preconditions	The metamodel of the view must be a submetamodel of the base model. That is, the view can only contain model element types that can occur in the base model.

The Query intent: associations

mandatoryProperty	1. Termination 2. The view must be a submodel of the base (injective graph homomorphism)

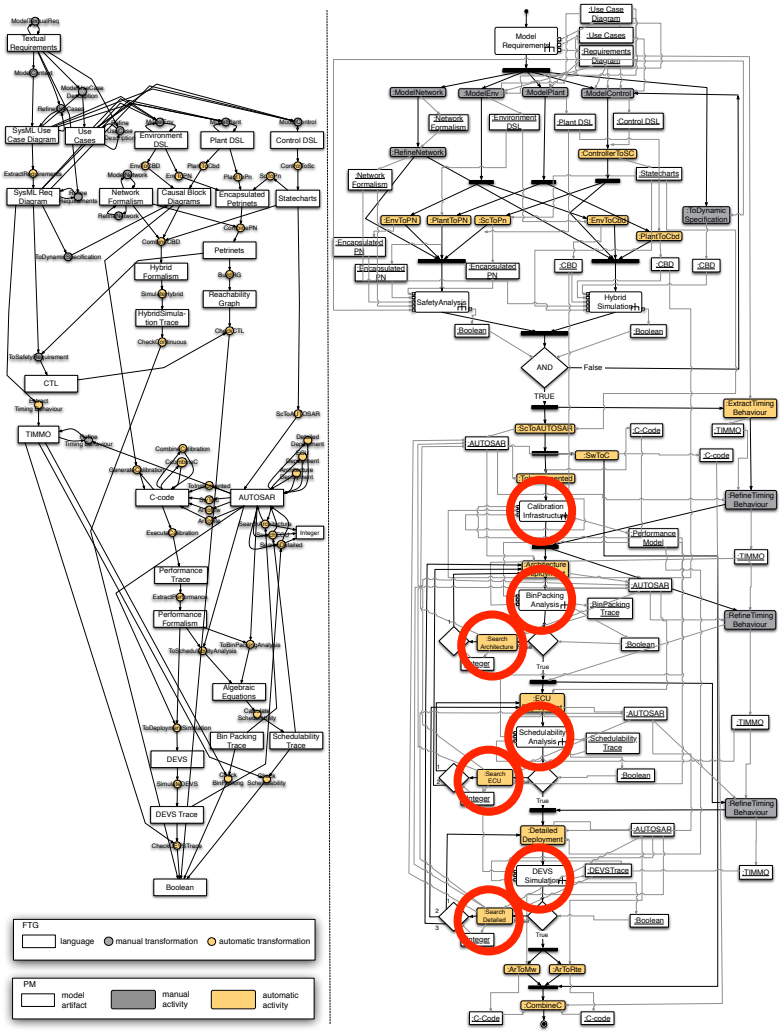
The Query intent: associations

mandatoryProperty	1. Termination 2. The view must be a submodel of the base (injective graph homomorphism)
optionalProperty	1. Semantics preservation - this should hold in cases when the query is used to extract a submodel for use by a human consumer since the view should not say anything that the base does not 2. Deterministic

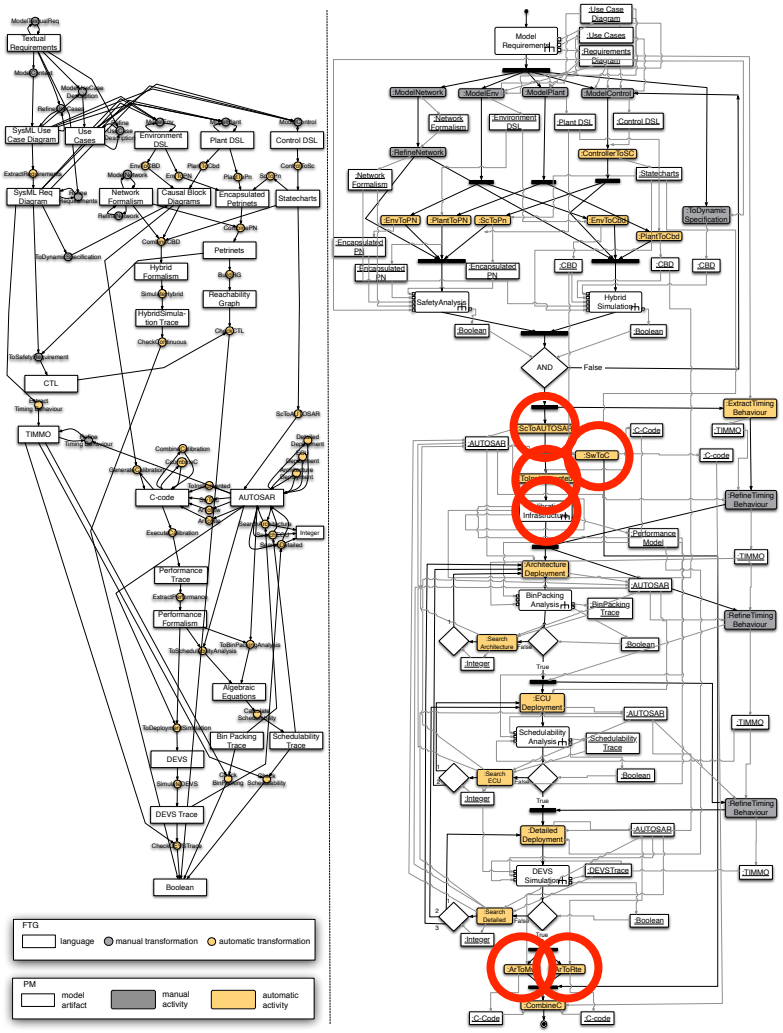
The Query intent: associations

mandatoryProperty	1. Termination 2. The view must be a submodel of the base (injective graph homomorphism)
optionalProperty	1. Semantics preservation - this should hold in cases when the query is used to extract a submodel for use by a human consumer since the view should not say anything that the base does not 2. Deterministic
relatedIntent	Metamodel Instance Generation, Abstraction

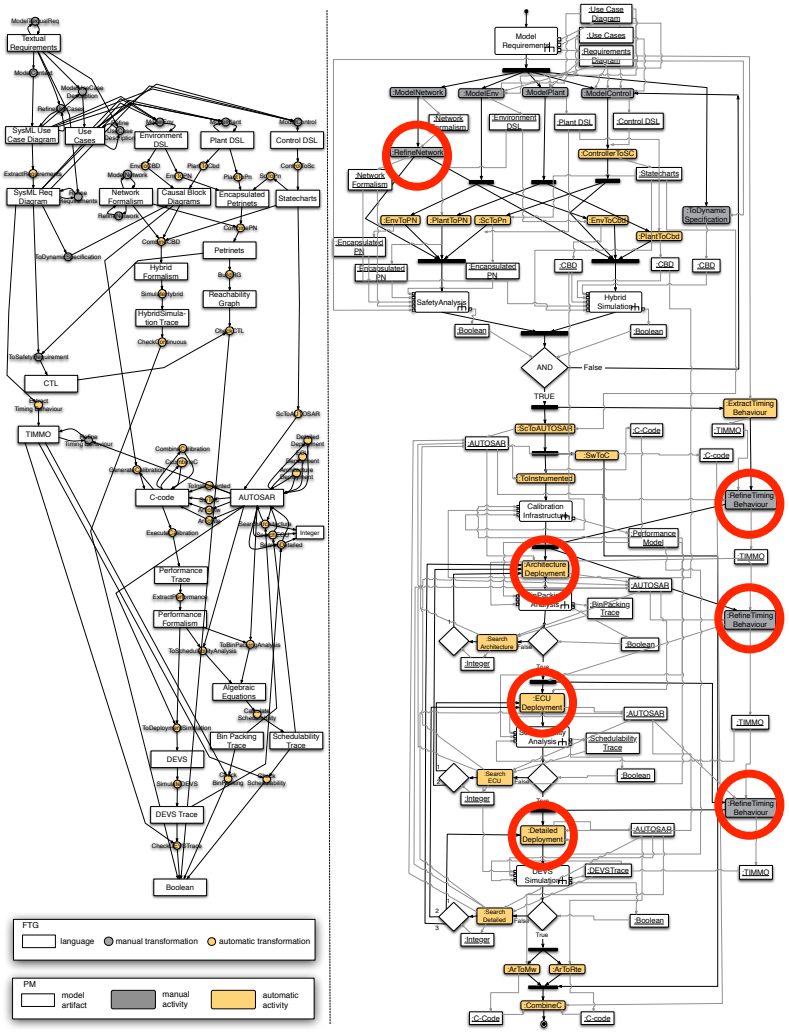
Other intentions: **Query**



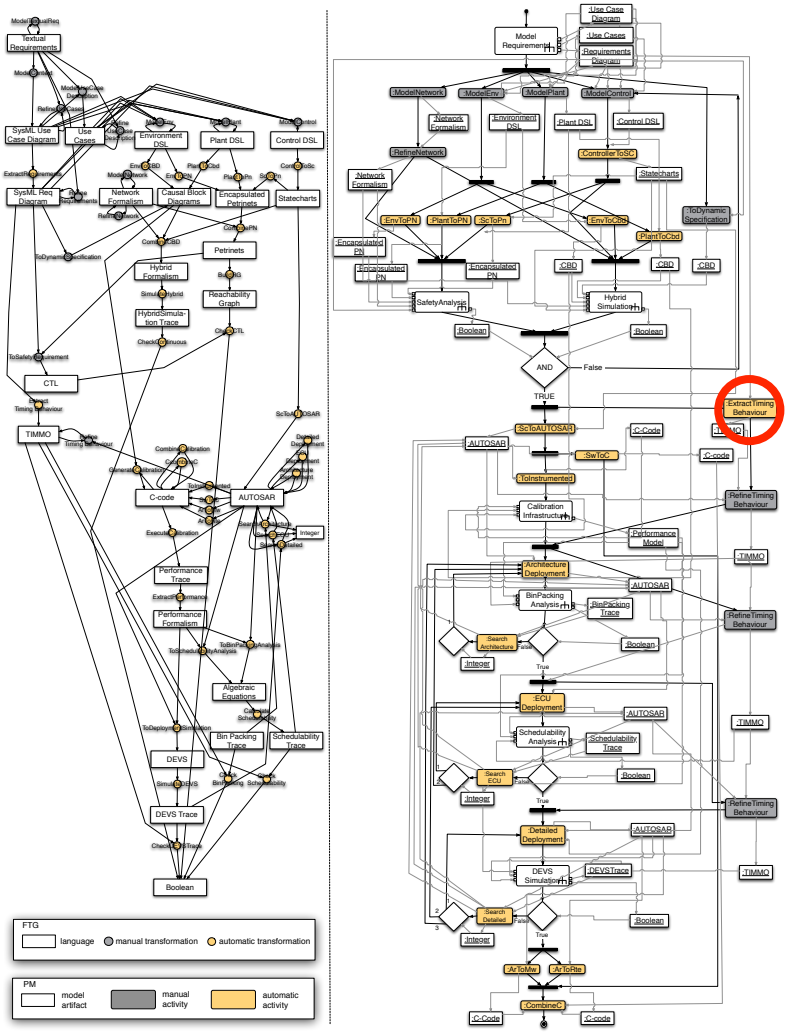
Other intentions: **Synthesis**



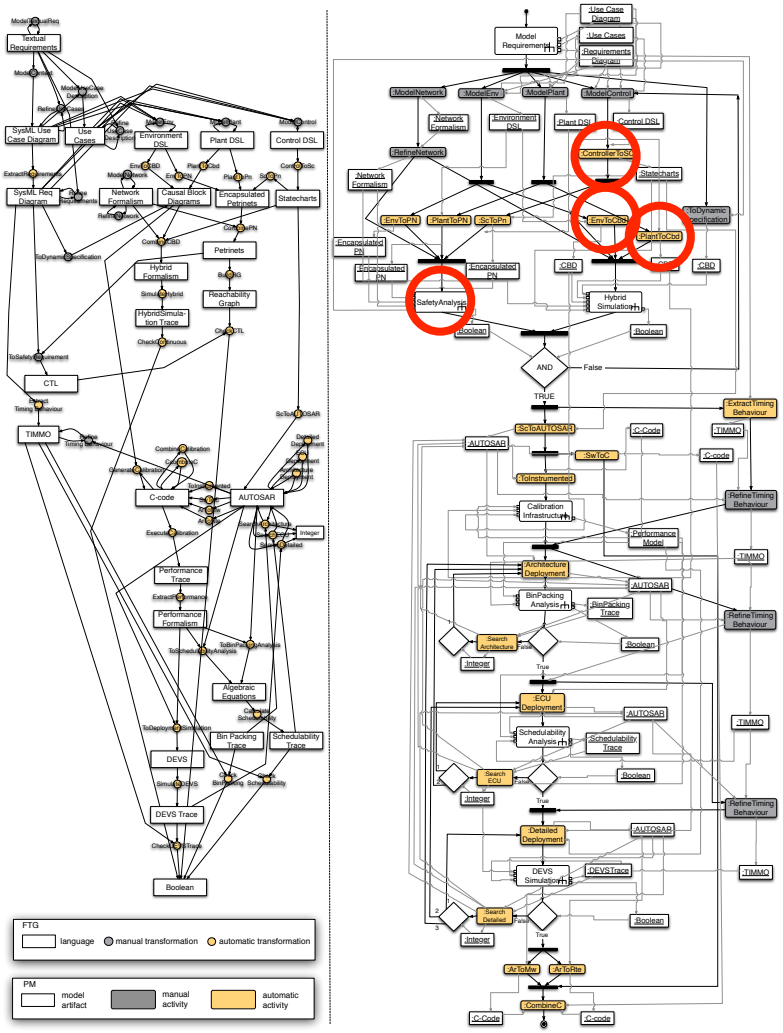
Other intentions: **Refinement**



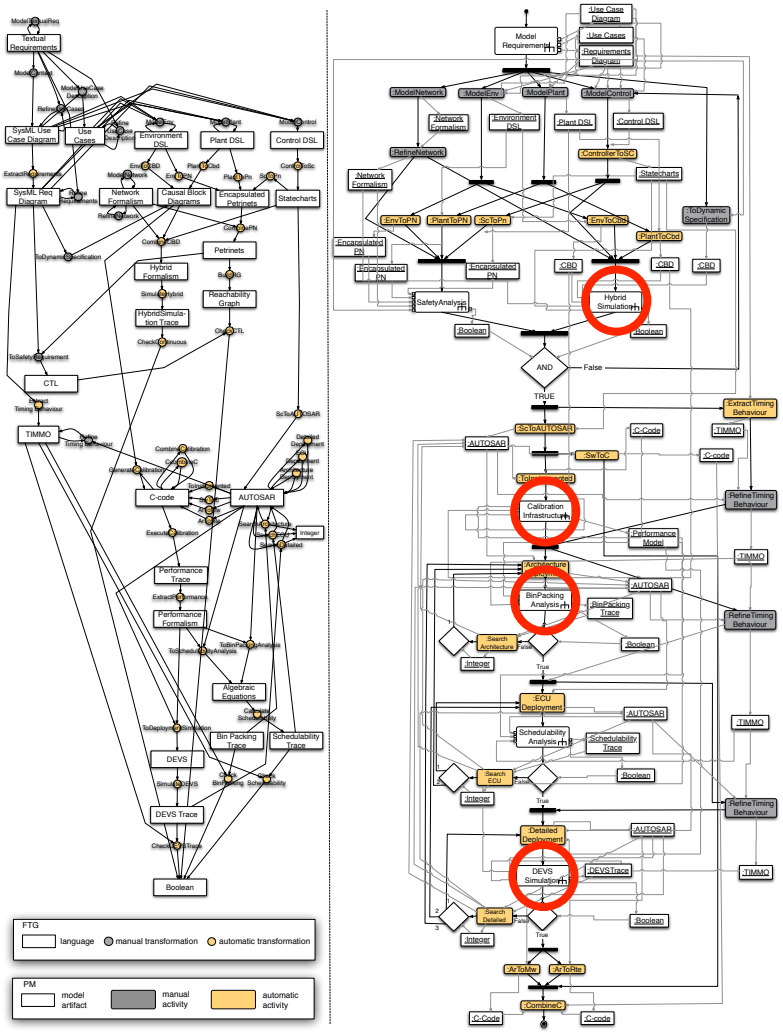
Other intentions: **Abstraction**



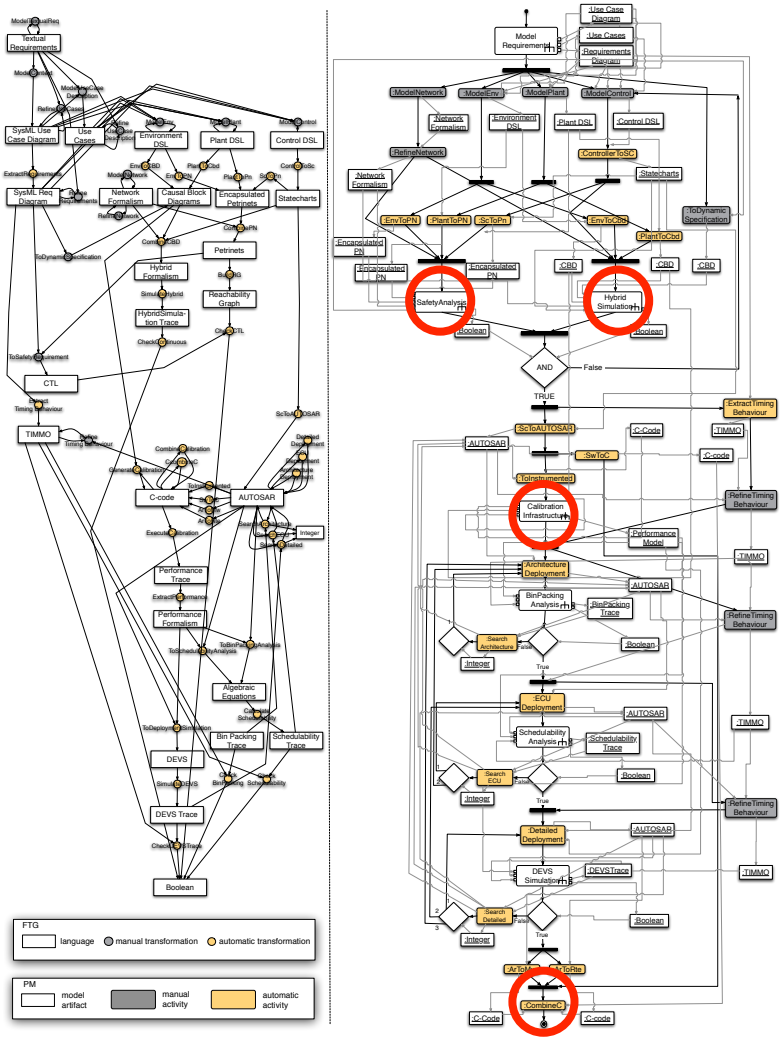
Other intentions:
Translational Semantics



Other intentions: **Simulation**



Other intentions: **Composition**



Contributions

1. Listed some common transformation intents
2. Introduced a metamodel to describe intents
3. Illustrated two intentions on a case study

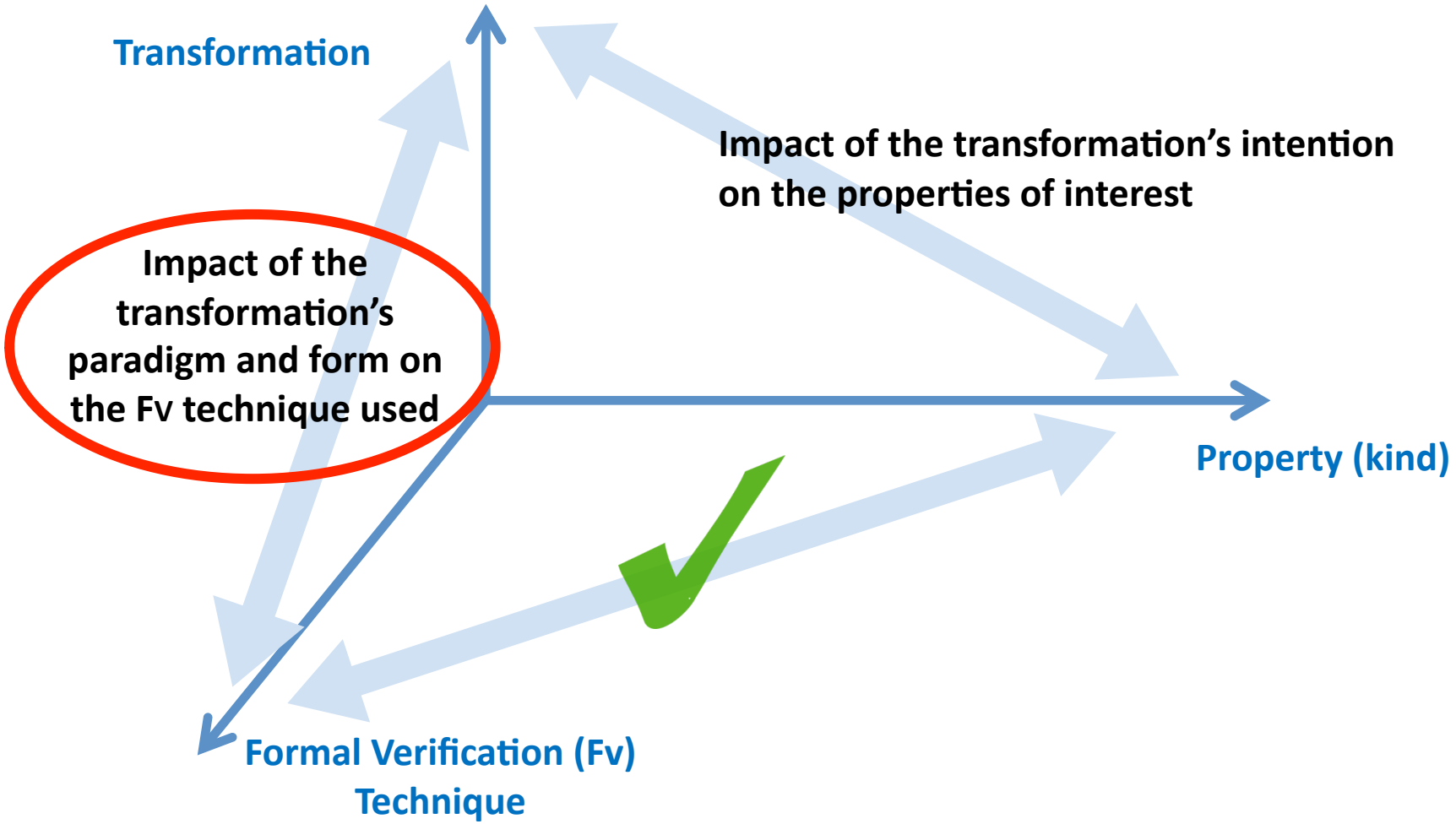
Uses of an Intent Catalog

- Requirements analysis for transformations
- Identification of properties, certification methods, and languages
- Model transformation language design

Future Work

- Are these all possible intents?
- Complete the power window case study with transformation intention information
- Understand the usefulness of our catalog:
  - Are intents “requirement patterns” for transformations?
  - Can we go one step further and mathematically formalise intents?
- Reasoning over transformation chains

Future Work


Transformation
Impact of the transformation's intention on the properties of interest
Impact of the transformation's paradigm and form on the Fv technique used
Property (kind)
Formal Verification (Fv) Technique