Teaching Model Driven Engineering
Through Formal Verification of Process Models

Benoît Combemale[1], Xavier Crégut[2], Arnaud Dieumegard[2], Marc Pantel[2] and Faiez Zalila[2]

[1] *University of Rennes 1*,
TRISKELL team (IRISA & INRIA)
benoit.combemale@irisa.fr – http://www.combemale.fr

[2] *University of Toulouse, IRIT – France*,
ACADIE team (CNRS)
firstname.lastname@enseeiht.fr

October 15, 2011

Context and Motivations

Software Engineering in Critical Embedded Systems

- MDE: Separation of Concerns thanks to tool-supported DSMLs
- Formal Methods: efficient tools (*e.g.,* model-checker) based on verification specific languages

Schizophrenic paradox between *MDE* and *Formal Methods* in Curriculums/Industry

How to bridge the gap between *domain specific* and *verification specific* languages?

Motivations for a new course

- Strong need in the industry of critical embedded systems (*e.g.,* local specificity!)
- Focus on the horizontal axis (V&V) instead of the vertical axis (code generation, platform independence, etc.)
- Voluntary choice of a case study that is not an executable software to show the unification provided by MDE

Context and Motivations

Software Engineering in Critical Embedded Systems

- MDE: Separation of Concerns thanks to tool-supported DSMLs
- Formal Methods: efficient tools (*e.g.,* model-checker) based on verification specific languages

Schizophrenic paradox between *MDE* and *Formal Methods* in Curriculums/Industry

How to bridge the gap between *domain specific* and *verification specific* languages?

Motivations for a new course

- Strong need in the industry of critical embedded systems (*e.g.*, local specificity!)
- Focus on the horizontal axis (V&V) instead of the vertical axis (code generation, platform independence, etc.)
- Voluntary choice of a case study that is not an executable software to show the unification provided by MDE

Context and Motivations

Software Engineering in Critical Embedded Systems

- MDE: Separation of Concerns thanks to tool-supported DSMLs
- Formal Methods: efficient tools (*e.g.,* model-checker) based on verification specific languages

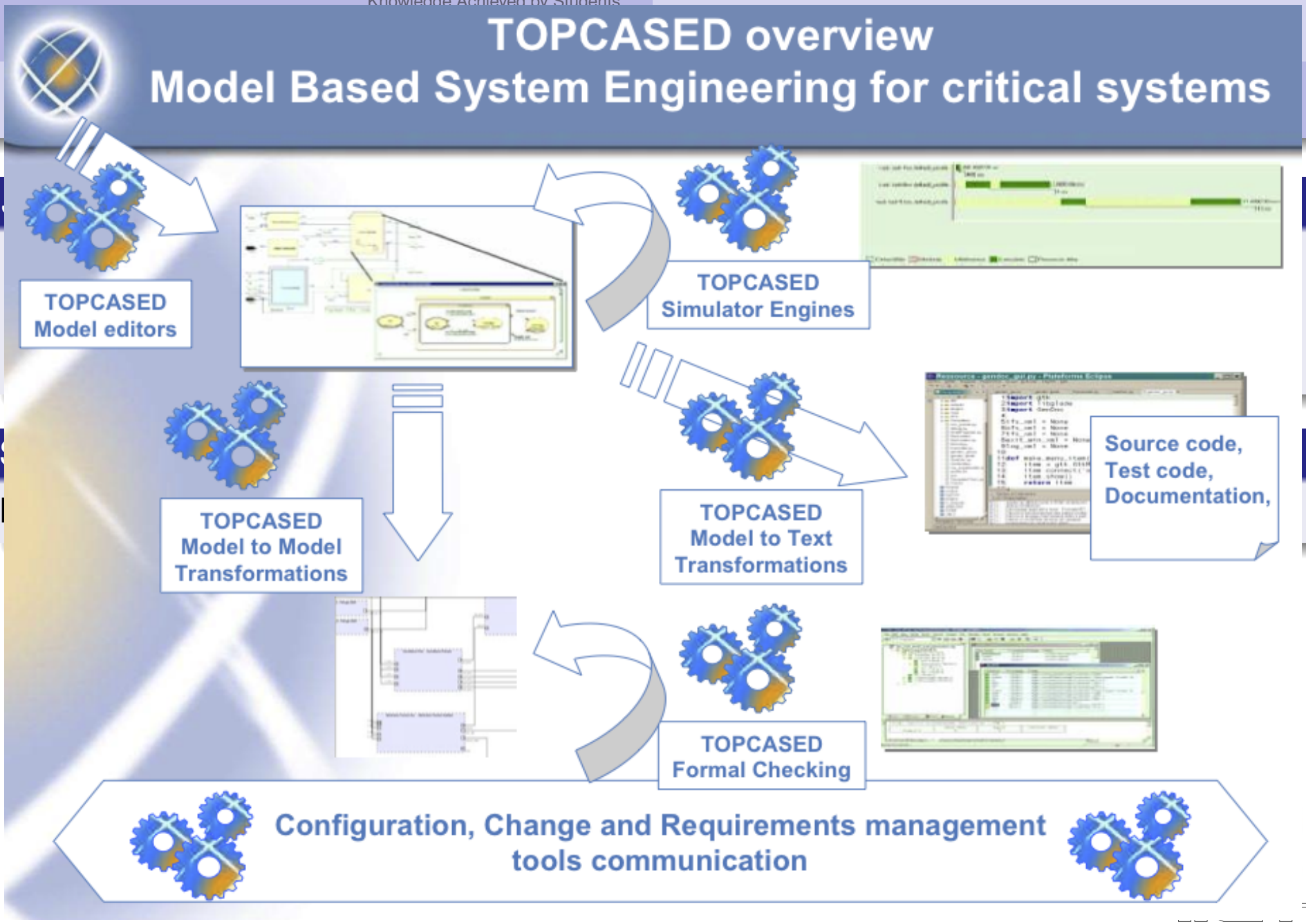
Schizophrenic paradox between *MDE* and *Formal Methods* in Curriculums/Industry

How to bridge the gap between *domain specific* and *verification specific* languages?

Motivations for a new course

- Strong need in the industry of critical embedded systems (*e.g.,* local specificity!)
- Focus on the horizontal axis (V&V) instead of the vertical axis (code generation, platform independence, etc.)
- Voluntary choice of a case study that is not an executable software to show the unification provided by MDE

TOPCASED overview
Model Based System Engineering for critical systems



Context and Motivations

Software Engineering in Critical Embedded Systems

- MDE: Separation of Concerns thanks to tool-supported DSMLs
- Formal Methods: efficient tools (*e.g.,* model-checker) based on verification specific languages

Schizophrenic paradox between *MDE* and *Formal Methods* in Curriculums/Industry

How to bridge the gap between *domain specific* and *verification specific* languages?

Motivations for a new course

- Strong need in the industry of critical embedded systems (*e.g.,* local specificity!)
- Focus on the horizontal axis (V&V) instead of the vertical axis (code generation, platform independence, etc.)
- Voluntary choice of a case study that is not an executable software to show the unification provided by MDE

Outline

1. Context and Motivations
2. Case Study and Content of the Course
   - Case Study: Process Model Verification
   - Schedule of the Course
3. Knowledge Achieved by Students
   - Metamodeling
   - Model Transformation
4. Discussions
5. Conclusion & Perspectives

Outline

1 Context and Motivations

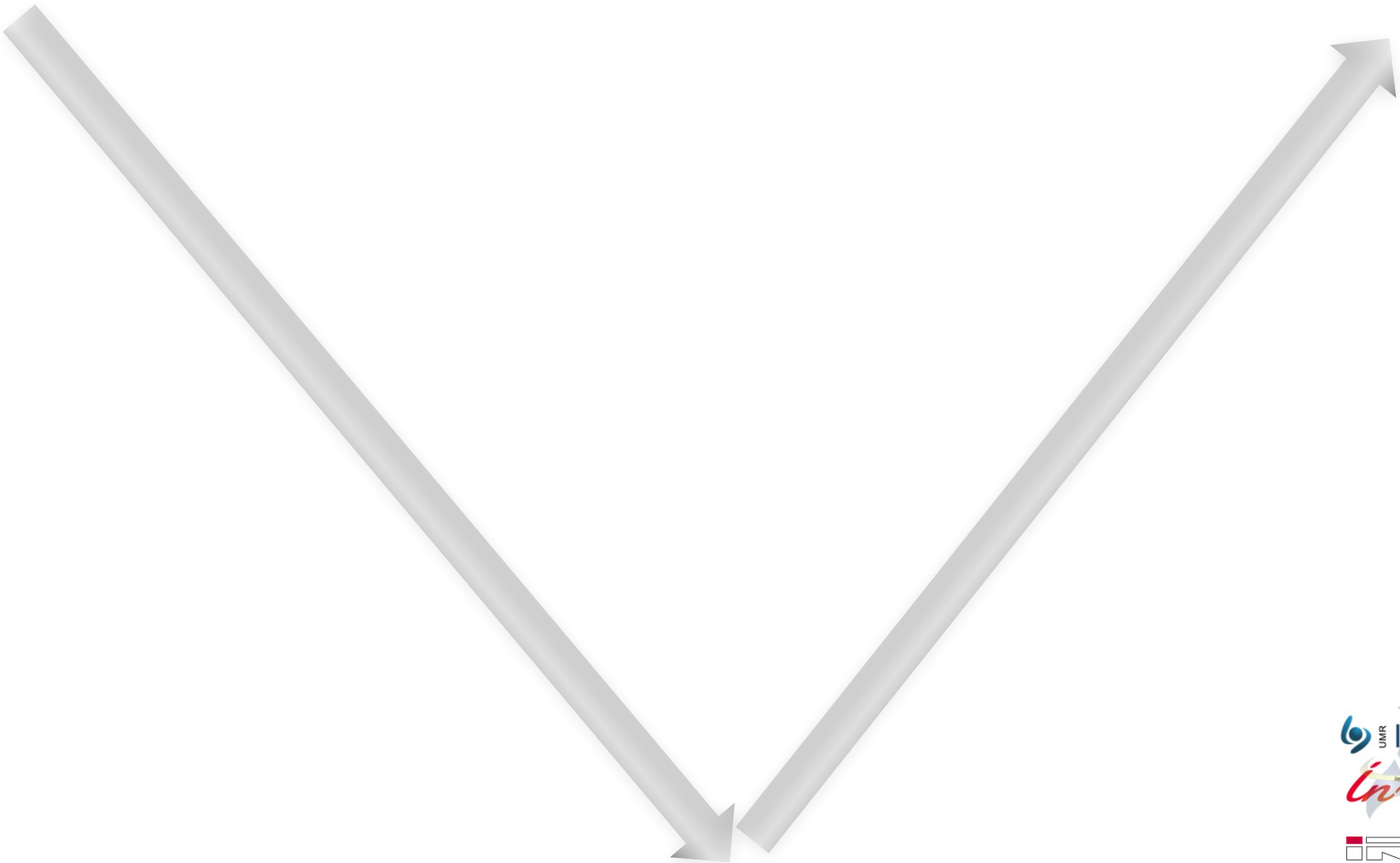
2 Case Study and Content of the Course
- Case Study: Process Model Verification
- Schedule of the Course

3 Knowledge Achieved by Students
- Metamodeling
- Model Transformation

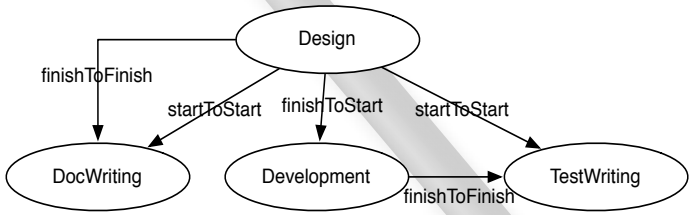
4 Discussions

5 Conclusion & Perspectives

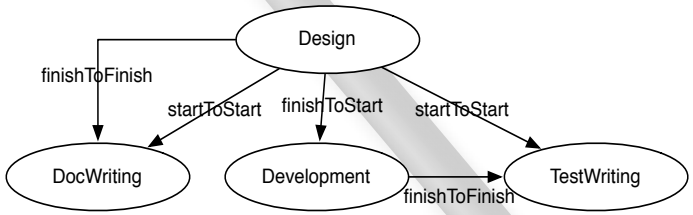
Case Study: (Process) Model Verification



Case Study: (Process) Model Verification

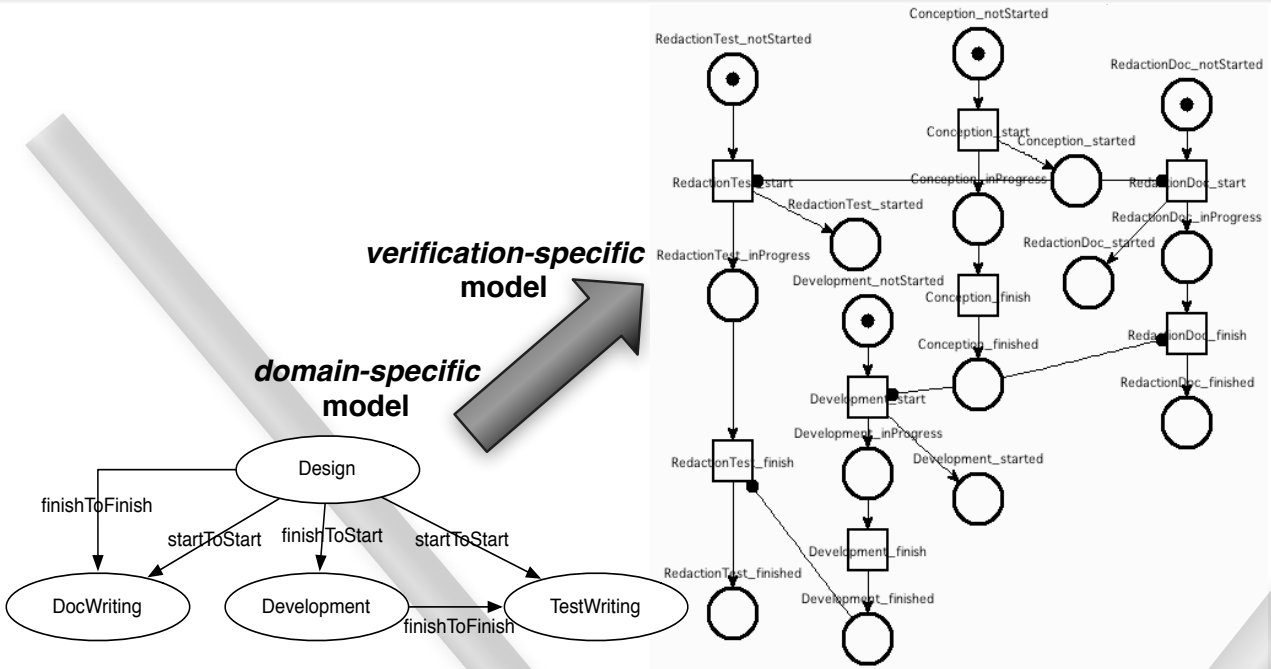


Case Study: (Process) Model Verification



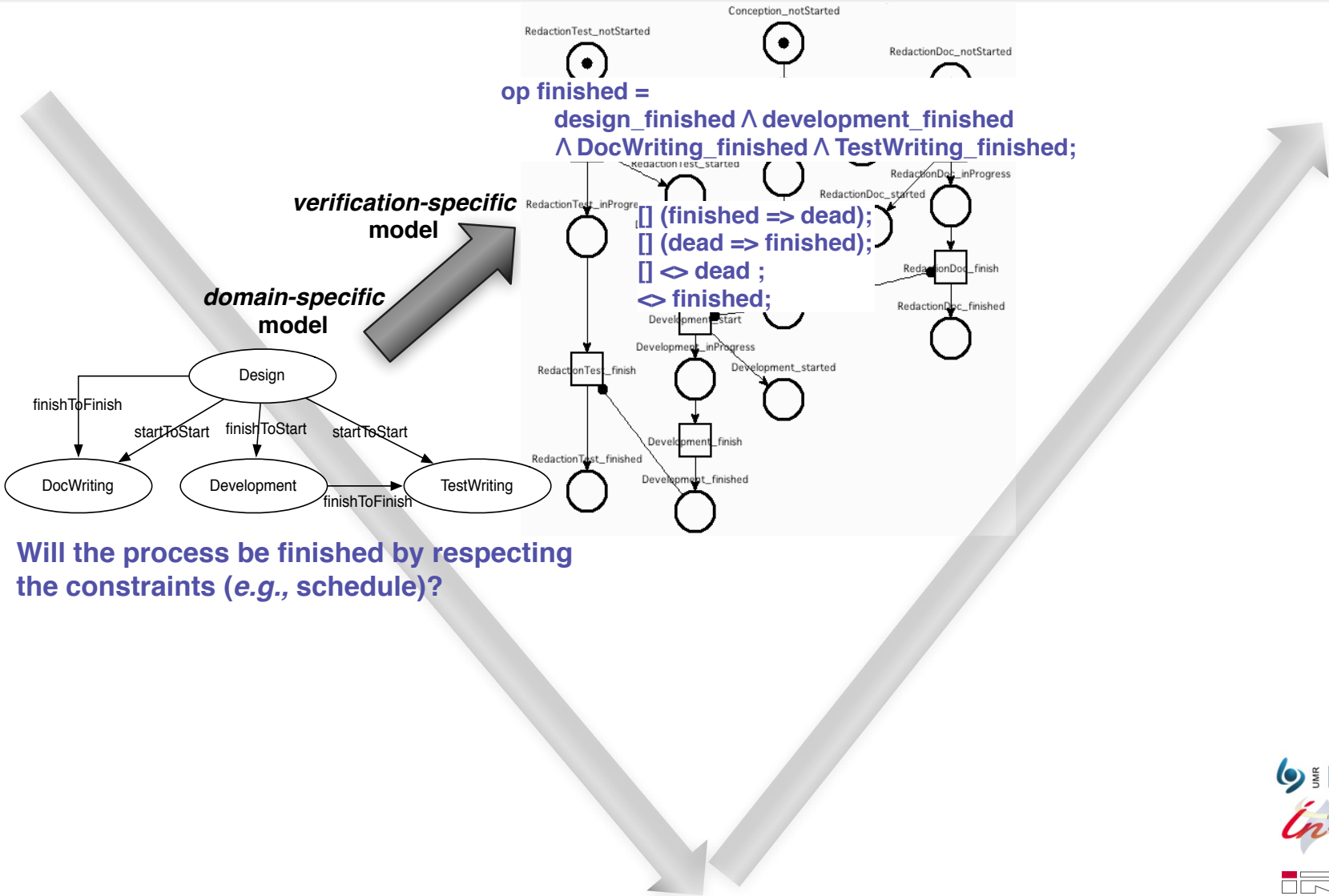
Will the process be finished by respecting the constraints (*e.g.*, schedule)?

Case Study: (Process) Model Verification



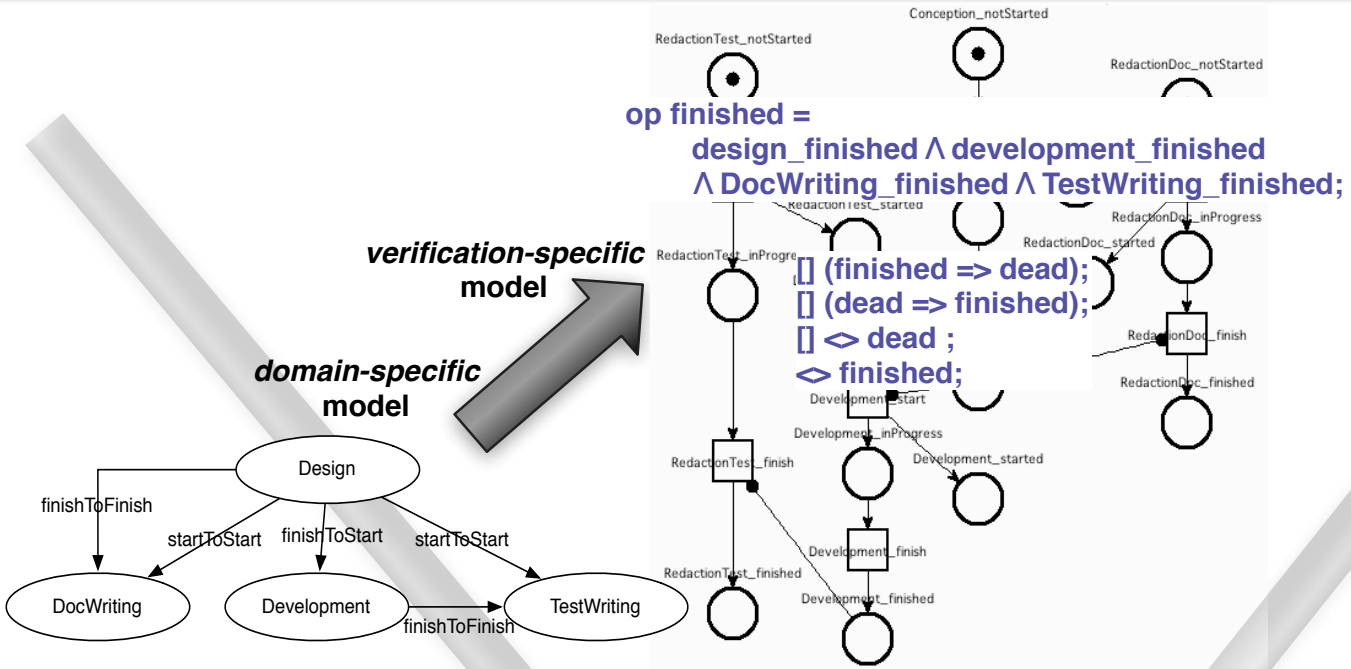
Will the process be finished by respecting the constraints (*e.g.*, schedule)?

Case Study: (Process) Model Verification



Will the process be finished by respecting the constraints (*e.g.*, schedule)?

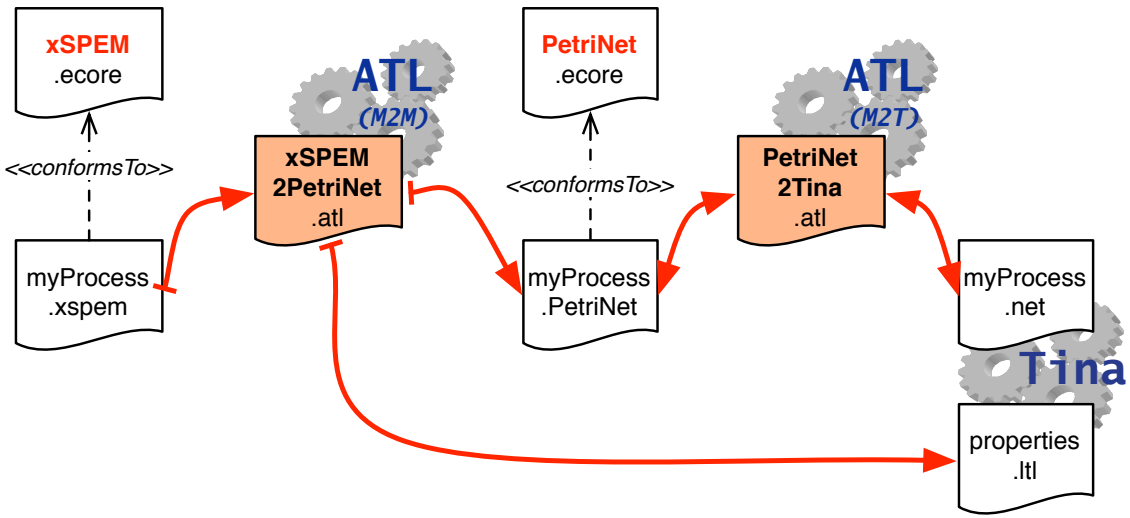
Case Study: (Process) Model Verification



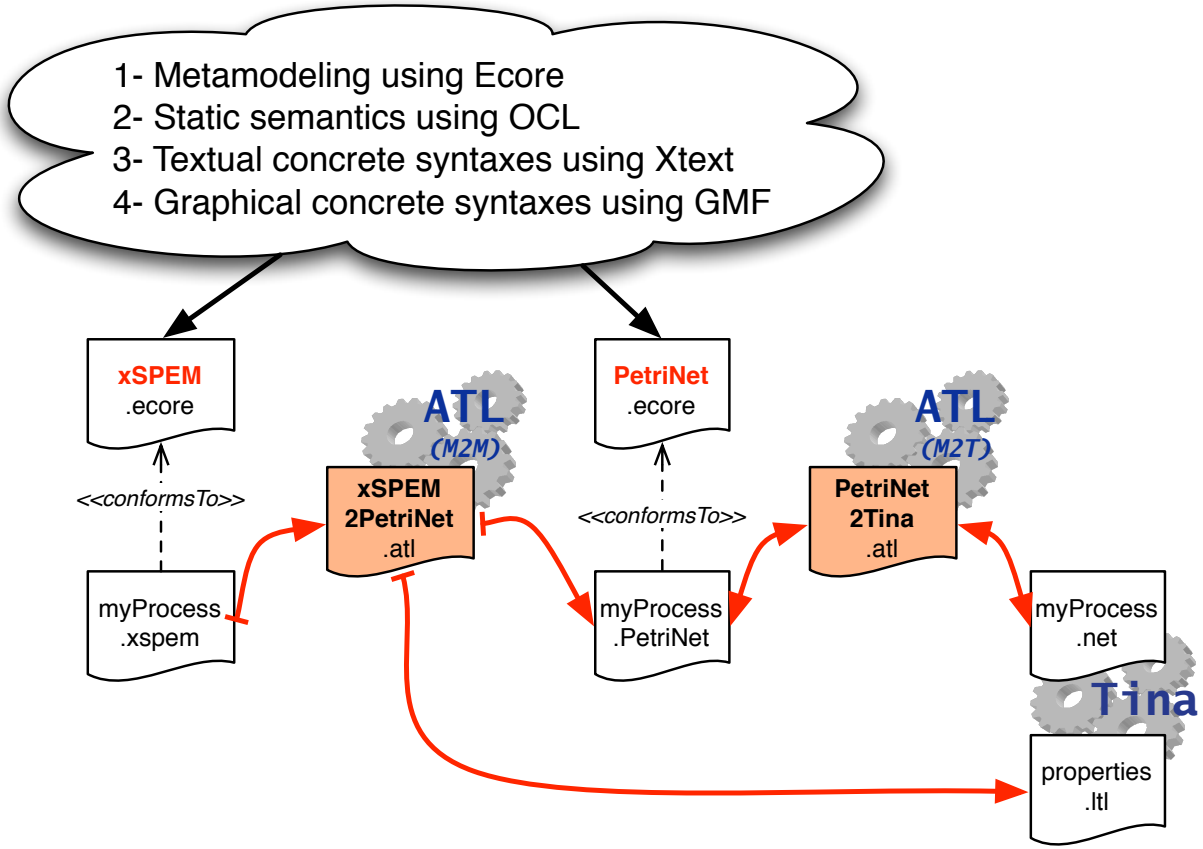
Will the process be finished by respecting the constraints (*e.g.*, schedule)?

Extensions: resources? duration? etc.

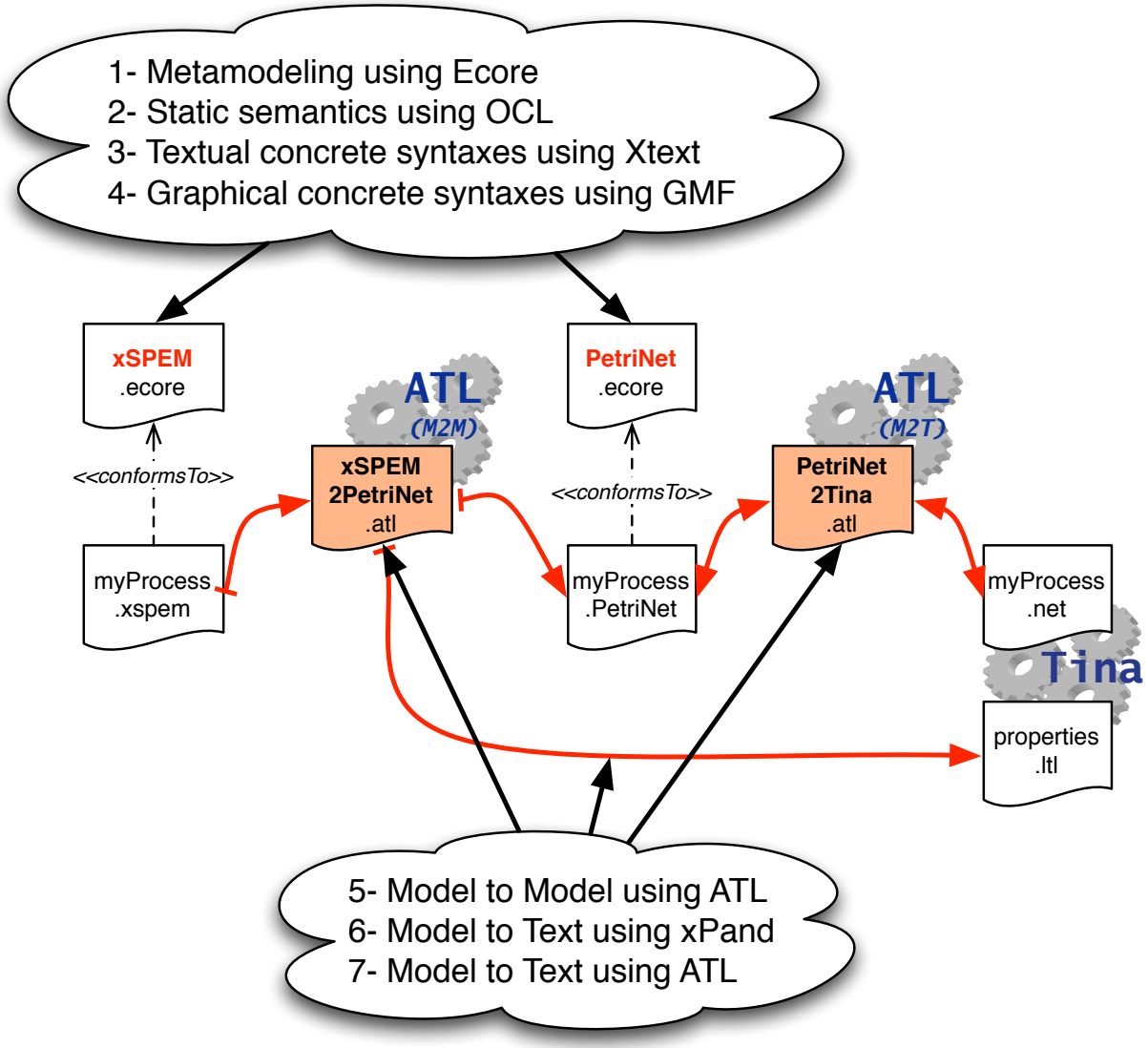
Case Study: (Process) Model Verification



Case Study: (Process) Model Verification



Case Study: (Process) Model Verification



Schedule of the Course

- One lecture (≈2h) and one practical work session (≈2h) for each topic

Metamodeling	1. Metamodeling using Eclipse/EMF
	2. Static semantics using OCL
	3. Textual concrete syntaxes using Xtext
	4. Graphical concrete syntaxes using GMF
Model transformations	5. Model to Model transformation using ATL
	6. Model to Text transformation using xPand
	7. Model to Text transformation using ATL

- a project is done by students based on the implementation of several extensions to the main case study

Project	Implementing extensions to the case study

Outline

1 Context and Motivations

2 Case Study and Content of the Course
- Case Study: Process Model Verification
- Schedule of the Course

3 Knowledge Achieved by Students
- Metamodeling
- Model Transformation

4 Discussions

5 Conclusion & Perspectives

Metamodeling I

1. **Abstract syntax** (metamodel) to define concepts and their relationships
   - Tools demonstrated on xSPEM: EMF/Ecore to define metamodels (using the Eclipse Graphical Editor), and the EMF Tree Editor to build conforming models
   - Open exercise: define the Petri net metamodel:
     - Petri net is easy to understand/formalize (graphical examples)
     - Several possible natural metamodels
     - All constraints are not captured by the abstract syntax $\Longrightarrow$ balance to be found between abstract syntax and static semantics
2. **Static semantics** to express properties not captured by the abstract syntax
   - Used tools: OCL with the Topcased checker
   - Main Students' Weaknesses/Problems
     - Invariant are misplaced
     - Overuse of the allInstances() operator $\Longrightarrow$ poor performances, and error-prone

Metamodeling II

3. **Concrete syntaxes** to easily handle models conforming to the abstract syntax
   - Textual concrete syntaxes: Xtext
     - Related to grammars and compiler construction
     - Specific metamodel according to the chosen syntax $\Longrightarrow$ M2M transformations required
   - Graphical concrete syntaxes: GMF
     - Good example of MDE!
     - But tools are not mature enough! (Nasty bugs!)

     $\Longrightarrow$ We plan to shift to OBEO Designer

$\Longrightarrow$ **Definition of DSML** (Domain Specific Modeling Language)

$\Longrightarrow$ **From MDE to SLE...** (Software Language Engineering)

Model Transformation I

1. **Model-to-Model transformations** (ATL)
   - Translation from Process to PetriNet representation (more suitable for verification activities)

   $\Rightarrow$ Motivations for ATL: declarative language rather than metaprogramming one
   - the case study illustrates: one to one rules, and one to many rules

   $\Longrightarrow$ illustrates *resolveTemp* construct (*resolveIn* in QVT)
   - advanced features: rule inheritance, called rules, imperative constructs

Model Transformation II

2. **Model-to-Text transformations** (xPand and ATL)
   - Generation of LTL properties the PetriNet model must verify
   - Generation of the Tina file representing the PetriNet model

   ⇒ Usage of ATL queries: metaprogramming language
   - helpers are new methods added on existing metamodels' elements

   ⇒ Usage of Xpand: template language
   - syntax is verbose and quite difficult (use of « and »)
   - not well suited for Tina file generation (very few keywords, very light syntax)
   - better for LTL formulae generation (constant parts)

Outline

1 Context and Motivations

2 Case Study and Content of the Course
- Case Study: Process Model Verification
- Schedule of the Course

3 Knowledge Achieved by Students
- Metamodeling
- Model Transformation

4 Discussions

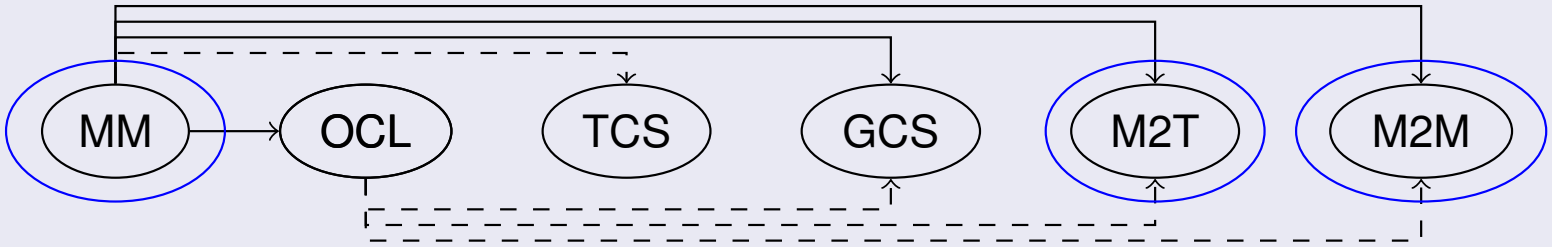
5 Conclusion & Perspectives

Discussions I

The formal verification case study is well accepted

- Even by students with no formal background (Petri nets are quite intuitive)
- An introduction to formal verification
- A coherent use of different MDE concepts and tools

The course and the case study are very modular



- Several curriculums:
  - N7: overview of MDE,
  - INSA: adds case study extensions as an assignment,
  - UTM: introduction to model transformations

Discussions II

Technical overload

- A great number of various technologies
- Tools are not mature enough (GMF, ATL)
- Always looking at new possibilities (Obeo Designer, Acceleo)

Studying verification does not imply the verification of the study

- Students clearly lack a culture of test and verification of their work
  - They often perform only one test to validate their work!
- The subject of the case study does not compensate this lack

Students' difficulties: depends on their curriculum

- Either comfortable with tools or abstraction
- Both abstraction abilities and tools friendliness are required

Outline

1 Context and Motivations

2 Case Study and Content of the Course
- Case Study: Process Model Verification
- Schedule of the Course

3 Knowledge Achieved by Students
- Metamodeling
- Model Transformation

4 Discussions

5 Conclusion & Perspectives

Conclusion

- Introduction of the MDE concepts and tools through the formal verification of domain specific models
- Attractive way that are getting more and more interest in the safety critical systems industry
- Bridging the *schizophrenic paradoxical* gap between *domain specific* and *verification specific* languages
- MDE as technological backbone for SLE...

Perspectives

- ... MDE is quite close to SLE (abstract syntax, concrete syntax, transformation of abstract syntax, etc.)
- Rely on MDE technologies in our compiler course instead of classical attribute grammar technologies

Thank you
for your attention...

Questions?