

# Towards a Formal Verification of Process Model Properties

## SIMPLEPDL & TOCL Case Study

Benoît Combemale<sup>1</sup>, Xavier Crégut<sup>1</sup>, Pierre-Loïc Garoche<sup>1</sup>,  
Xavier Thirioux<sup>1</sup> & François Vernadat<sup>2</sup>

<sup>1</sup>IRIT CNRS  
2, rue Charles Camichel - BP 7122  
F-31071 Toulouse Cedex 7

<sup>2</sup>LAAS CNRS  
7, avenue du colonel Roche  
F-31077 Toulouse Cedex 4

june 16, 2007

# Plan

- 1 Context
- 2 Case Study : Process Model Validation
  - SIMPLEPDL, a process modelling language
  - Properties on Process Models
- 3 An Approach to Validation through Petri Nets and LTL
  - Characterising Properties & States
  - Extending the Metamodel
  - Expressing Temporal Properties with TOCL
  - Denotational Semantics as Petri Nets
  - Model Validation and Feedback
- 4 Conclusion

# Context

## Some pieces of history

- 1997 : the UML, v 1.0
- 2000 : the MDA proposal by the OMG
- Nowadays, some major trends :
  - metamodeling
  - Domain Specific Languages
  - model transformation

# Context

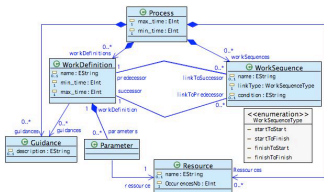
## Motivation

- to define a generic framework able to describe various execution semantics
  - axiomatic semantics (as `pre/post/inv` in the OCL)
  - operational semantics (as in Kermeta)
  - denotational semantics (model transformation)
- to extend existing frameworks, such as EMF or GME
- in order to simulate or validate models within

## Our approach : a property-driven methodology

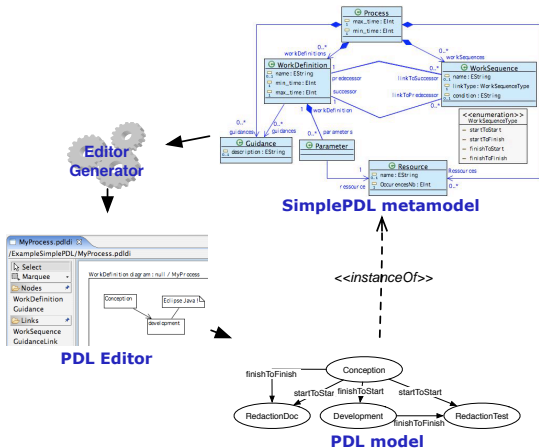
- 1 expertise in the specific domain
- 2 expertise in some validation technology

# Our Approach (through case study)

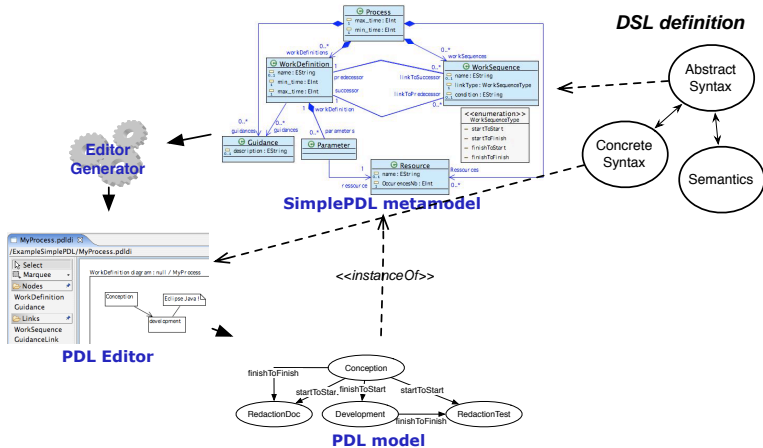


SimplePDL metamodel

# Our Approach (through case study)



# Our Approach (through case study)





# Plan

- 1 Context
- 2 Case Study : Process Model Validation
  - SIMPLEPDL, a process modelling language
  - Properties on Process Models
- 3 An Approach to Validation through Petri Nets and LTL
  - Characterising Properties & States
  - Extending the Metamodel
  - Expressing Temporal Properties with TOCL
  - Denotational Semantics as Petri Nets
  - Model Validation and Feedback
- 4 Conclusion

# Plan

## 1 Context

## 2 Case Study : Process Model Validation

- SIMPLEPDL, a process modelling language
- Properties on Process Models

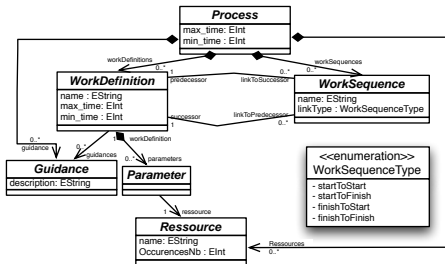
## 3 An Approach to Validation through Petri Nets and LTL

- Characterising Properties & States
- Extending the Metamodel
- Expressing Temporal Properties with TOCL
- Denotational Semantics as Petri Nets
- Model Validation and Feedback

## 4 Conclusion

# SIMPLEPDL, a Process Modelling Language

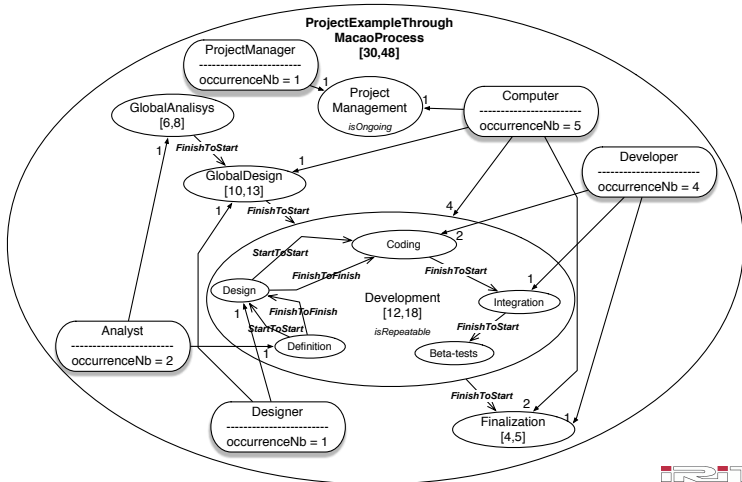
- stems from :
  - SPEM (OMG)
  - UMA (EPF)
- could easily be extended :
  - roles
  - products
  - ...



SIMPLEPDL metamodel

# Modelling a complex process

## MACAO process



# What do we want to check ?

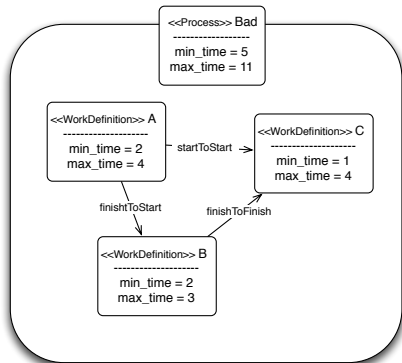
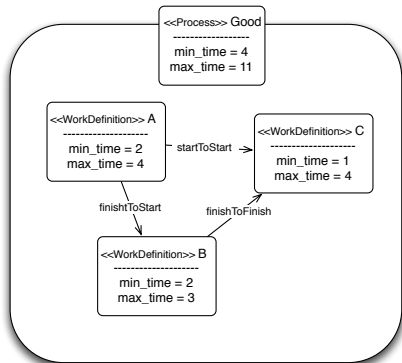
- resource constraints
  - computers
  - manpower
- timing constraints
  - minimum achievement time
  - maximum achievement time
- causality constraints
  - startToStart
  - startToFinish
  - finishToStart
  - finishToFinish
- ...
  - for some execution
  - or for all executions

# What do we want to check ?

- resource constraints
  - computers
  - manpower
- timing constraints
  - minimum achievement time
  - maximum achievement time
- causality constraints
  - startToStart
  - startToFinish
  - finishToStart
  - finishToFinish
- ...
- for some execution
- or for all executions

# A sample property

- checking the global *min\_time* constraint



# Plan

## 1 Context

## 2 Case Study : Process Model Validation

- SIMPLEPDL, a process modelling language
- Properties on Process Models

## 3 An Approach to Validation through Petri Nets and LTL

- Characterising Properties & States
- Extending the Metamodel
- Expressing Temporal Properties with TOCL
- Denotational Semantics as Petri Nets
- Model Validation and Feedback

## 4 Conclusion

## Some SimplePDL-expert properties

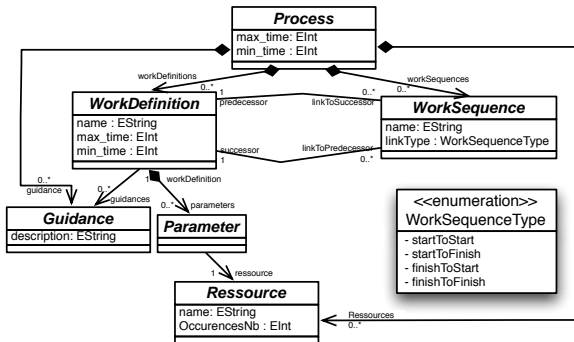
### For all executions

- every WD must start and then finish
- once a WD is finished, it remains so
- resource and causality constraints must hold

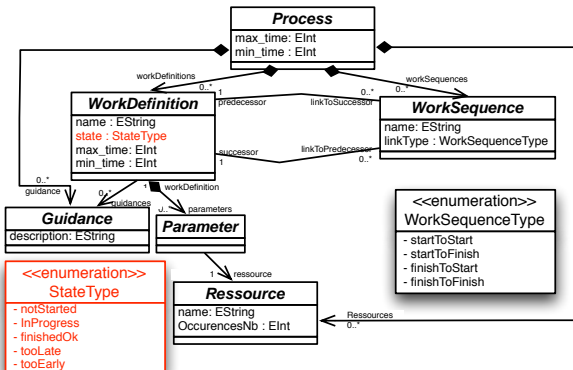
### For some execution

- every WD must take between *min* and *max* time units to complete
- the overall process is able to finish

# Extending the metamodel



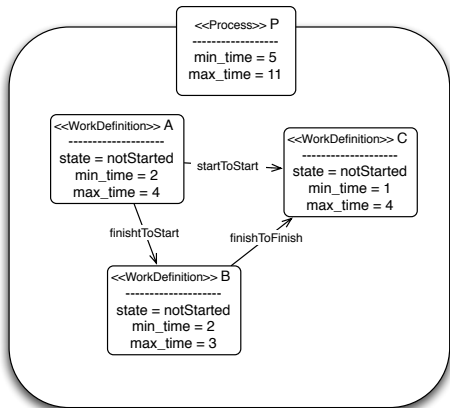
# Extending the metamodel



# A sample run

## Illustrating operational semantics

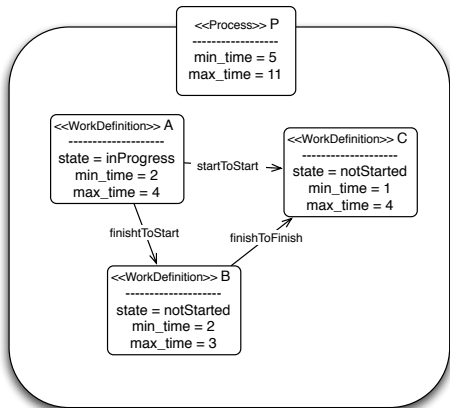
- $t = 0$  : *WDs* are notStarted



# A sample run

## Illustrating operational semantics

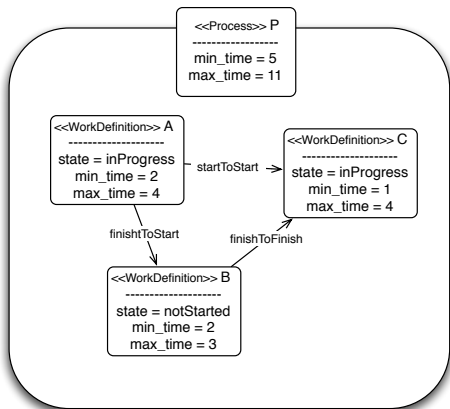
- $t = 0$  : *WDs* are notStarted
- $t = 1$  : *A* starts



# A sample run

## Illustrating operational semantics

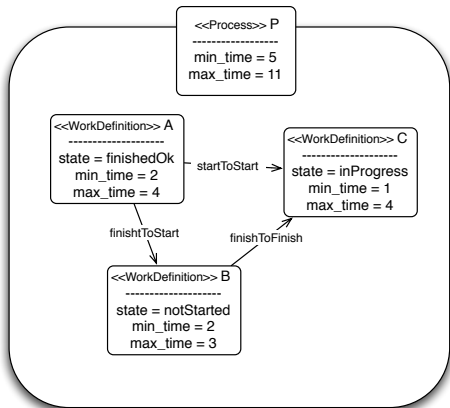
- $t = 0$  : *WDs* are notStarted
- $t = 1$  : *A* starts
- $t = 3$  : *B* starts



# A sample run

## Illustrating operational semantics

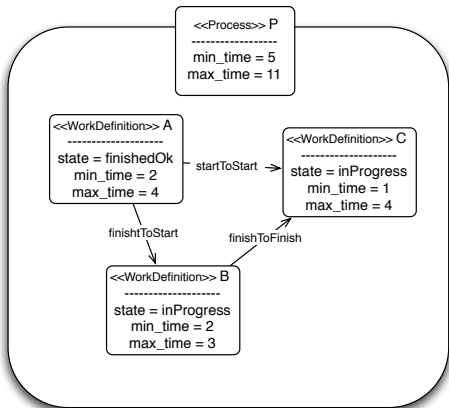
- $t = 0$  : *WDs* are notStarted
- $t = 1$  : *A* starts
- $t = 3$  : *B* starts
- $t = 4$  : *A* completes



# A sample run

## Illustrating operational semantics

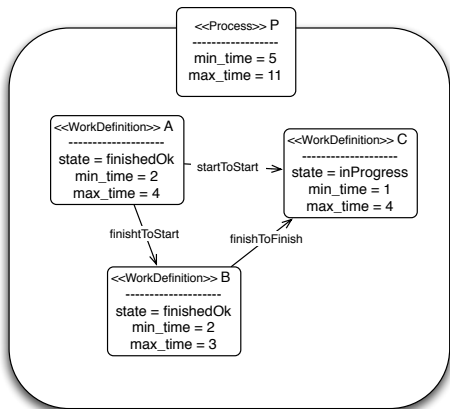
- $t = 0$  : *WDs* are notStarted
- $t = 1$  : *A* starts
- $t = 3$  : *B* starts
- $t = 4$  : *A* completes
- $t = 5$  : *C* starts



# A sample run

## Illustrating operational semantics

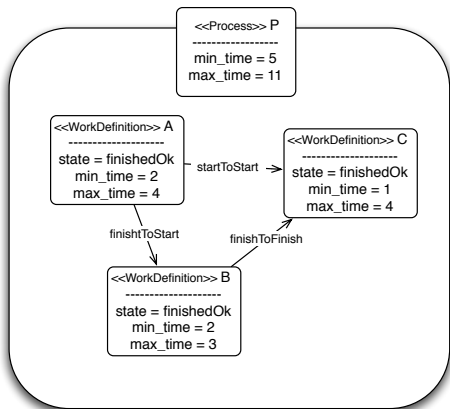
- $t = 0$  : *WDs* are notStarted
- $t = 1$  : *A* starts
- $t = 3$  : *B* starts
- $t = 4$  : *A* completes
- $t = 5$  : *C* starts
- $t = 7$  : *B* completes



# A sample run

## Illustrating operational semantics

- $t = 0$  : *WDs* are notStarted
- $t = 1$  : *A* starts
- $t = 3$  : *B* starts
- $t = 4$  : *A* completes
- $t = 5$  : *C* starts
- $t = 7$  : *B* completes
- $t = 8$  : *C* completes



# The Temporal Object Constraint Language

## TOCL (Gogolla & al., 2002) embeds

- the Object Constraint Language for spatial relations
- the Linear Temporal Logic for time relations

## TOCL is used

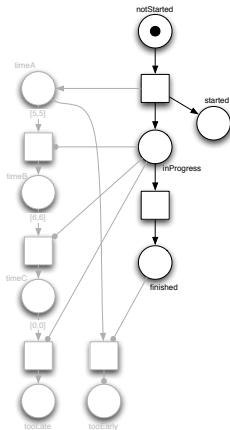
- to express fine behavioral spec (*next, existsNext, always, sometime, ...*)
- about some execution or all executions

## Some properties of WD alone

- $\forall w, (w.state = \text{notStarted} \wedge \text{sometime } w.state = \text{inProgress})$
- $\forall w, \text{always } (w.state = \text{inProgress} \Rightarrow \text{sometime } w.state \in \{\text{finishedOk}, \text{tooEarly}, \text{tooLate}\})$
- $\forall w, \text{always } (w.state = \text{finishedOk} \Rightarrow \text{always } w.state = \text{finishedOk})$
- $\neg \exists w, \text{always } w.state \neq \text{finishedOk}$

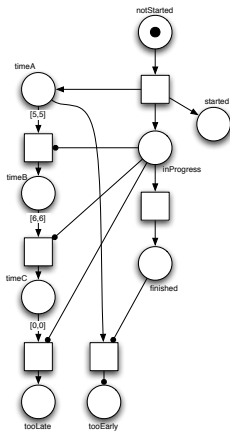
# Expressing WorkDefinition Semantics through Petri Nets

A first net to encode simple states :



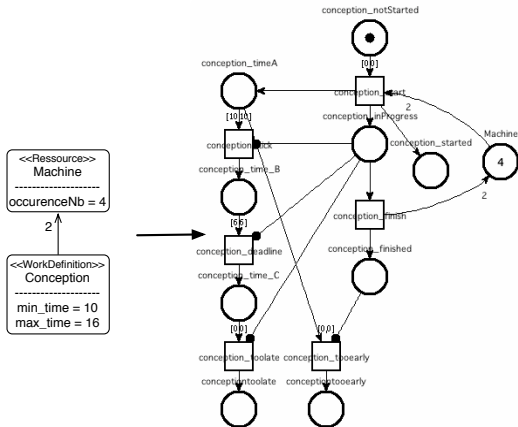
# Expressing WorkDefinition Semantics through Petri Nets

Another net to encode time constraints :



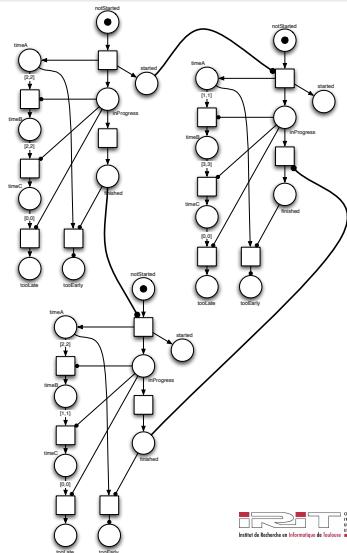
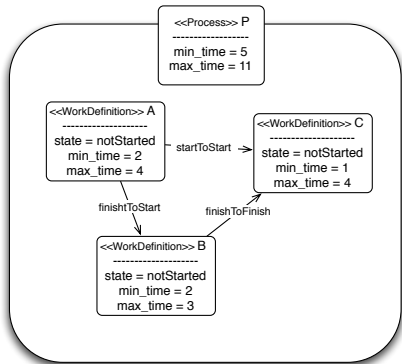
# Expressing WorkDefinition Semantics through Petri Nets

Then, we add resource constraints :



# Expressing WorkDefinition Semantics through Petri Nets

Finally, we add causality constraints :

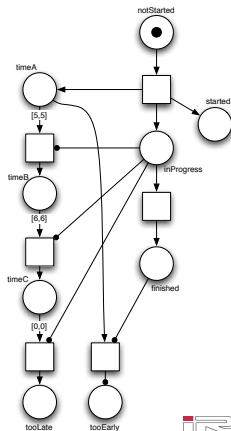


# A sample run

## Translation into Petri nets

A WD with  $min\_time = 5$  and  $max\_time = 11$  time units

- $t = 0$  : WD is notStarted

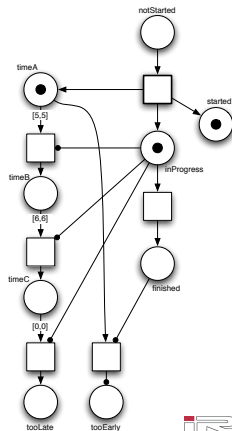


# A sample run

## Translation into Petri nets

A WD with  $min\_time = 5$  and  $max\_time = 11$  time units

- $t = 0$  : WD is notStarted
- $t = 1$  : WD starts



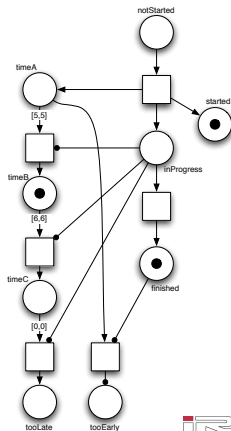


# A sample run

## Translation into Petri nets

A WD with  $min\_time = 5$  and  $max\_time = 11$  time units

- $t = 0$  : WD is notStarted
- $t = 1$  : WD starts
- $t = 6$  : WD is now on time
- $t = 7$  : WD completes on time



## Some features of our translation

### Nice properties

- functional pattern-matching ATL program
- structural (a WD is a net & a WD.state is a marking)
- modular (a constraint is also a net)
- incremental (a constraint may be plugged in & out)
- traceable

### Target language comes equipped : <http://www.laas.fr/tina/>

- **nd** (*NetDraw*) : editor and simulator of temporal Petri nets
- **tina** : scanner of temporal Petri nets state spaces
- **selt** : model-checker for the temporal logic *SE-LTL* (State/Event *LTL*), with counter-example generation

# Plan

- 1 Context
- 2 Case Study : Process Model Validation
  - SIMPLEPDL, a process modelling language
  - Properties on Process Models
- 3 An Approach to Validation through Petri Nets and LTL
  - Characterising Properties & States
  - Extending the Metamodel
  - Expressing Temporal Properties with TOCL
  - Denotational Semantics as Petri Nets
  - Model Validation and Feedback
- 4 Conclusion

# Conclusion

## Conclusion & Future Works

### What was achieved

- definition of an execution semantics and spec. for SimplePDL
- well-behaved translation into Petri net tools
- automatic validation of process properties
- with a feedback (traceability)
- implementation and integration as a TOPCASED service

### What remains to be done

- generalisation to other source DSLs & target languages
- proof & methodology to establish equivalence between semantics (operational vs. denotational)
- automatic traceability

Thank you  
for your attention...