UML & Formal Methods

Panel – Does UML need its own formal notation ?

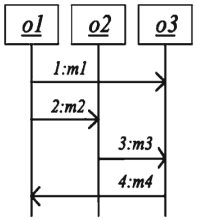
Frédéric Mallet

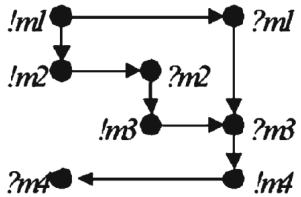
Aoste Project

UML & FM - Kita Kyushu -
September 27th, 2008

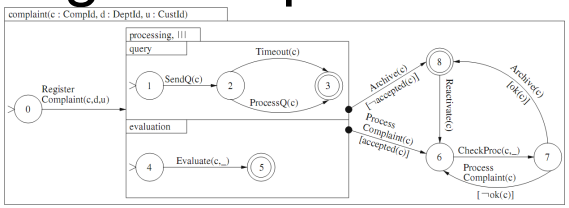
Session on Formal Behavior

- Ambiguity and structural properties of basic sequence diagrams
  - Interactions + trace

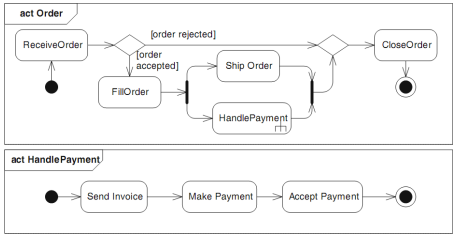


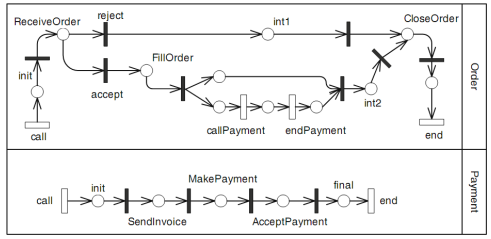


- Extending statecharts with process algebra operators
  - Untimed StateMachines + CSP

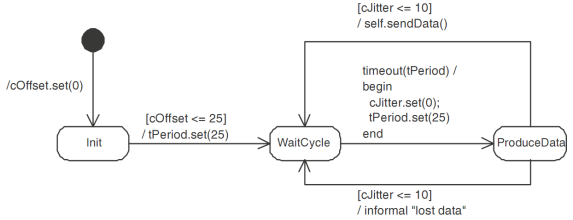


- UML Behavioral consistency checking using instantiable Petri Nets
  - Activities + PN





- Timing analysis and validation with UML
  - StateMachines + Timed automata



Two kinds of approaches

Supplemental (R. France et al., 98)

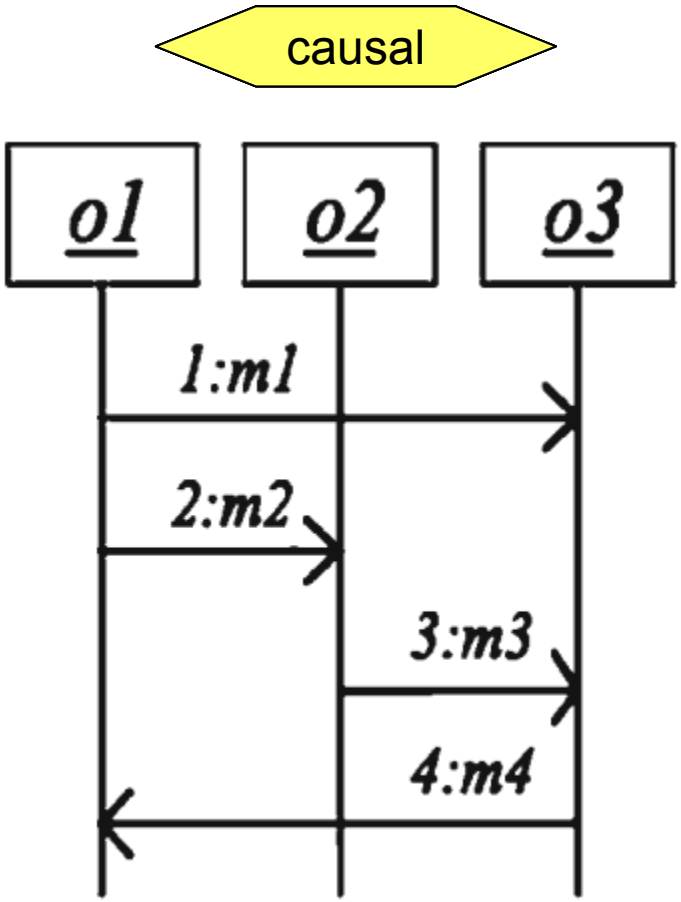
- Formally defining UML semantics
  - Use of a formal language to express what is expressed in English within the UML superstructure
  - Various formal languages : CSP, PN, 1st order logic, Timed Automata, …

Integrated

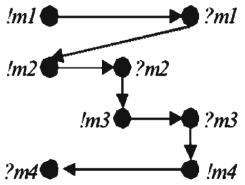
- Extending UML with semantics
  - UML is used as a tool editor
  - Semantics outside UML and applied with annotations (profile)

- In both cases,
  - Usually addressing (part of) Interactions, Activities, StateMachines
  - Transformations towards proprietary tools to perform analysis and/or verifications: simulation, model-checking, …

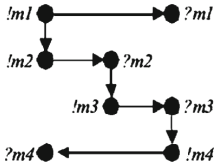
First paper



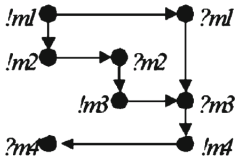
linear



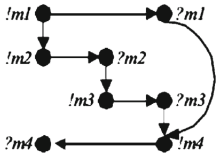
emission



UML2



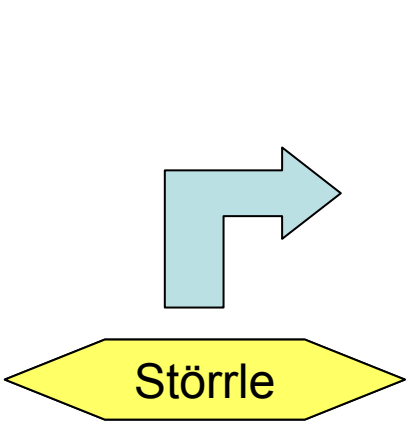
causal

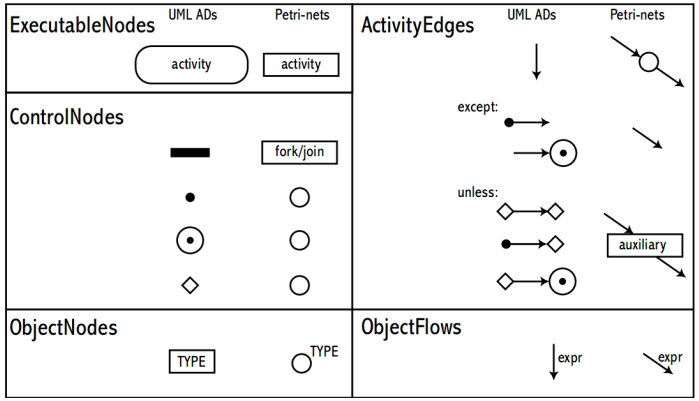


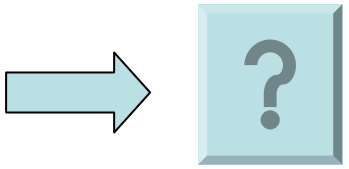
- Compare their constraining power : level of concurrency
- Do we want to choose between all these ?
  - Use UML as a framework for combining all of these semantics
  - Apply directors (like in Ptolemy) to choose the suitable semantics

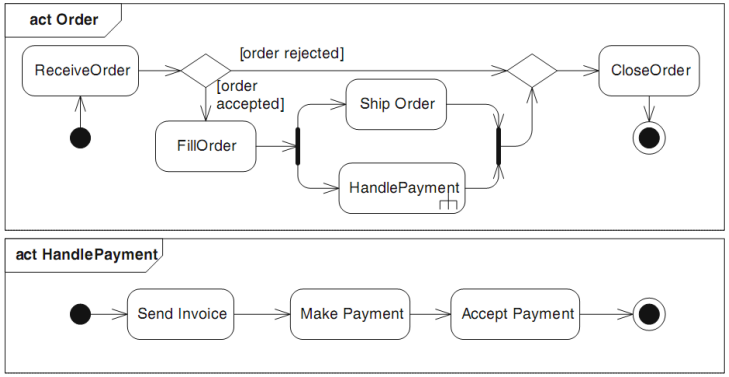
Störrle, ENTCS, 2005

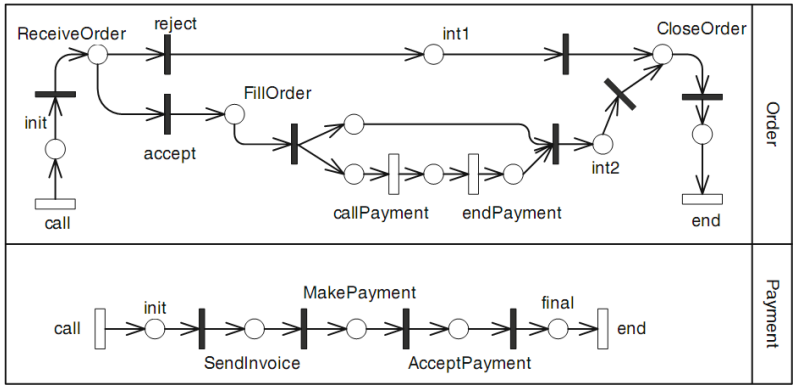
Third paper

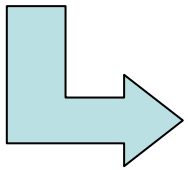


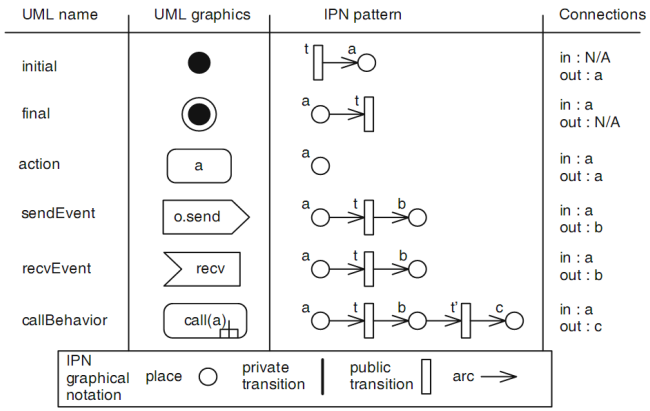


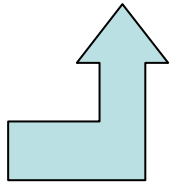




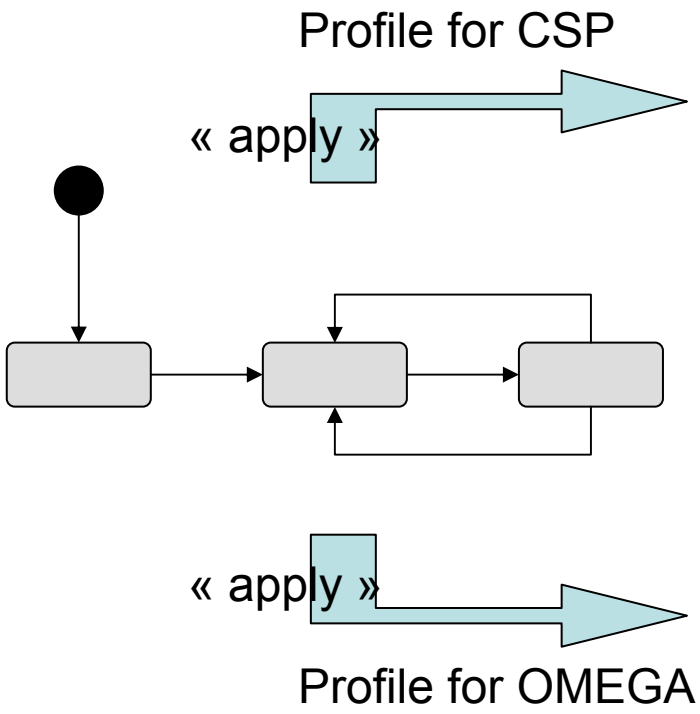


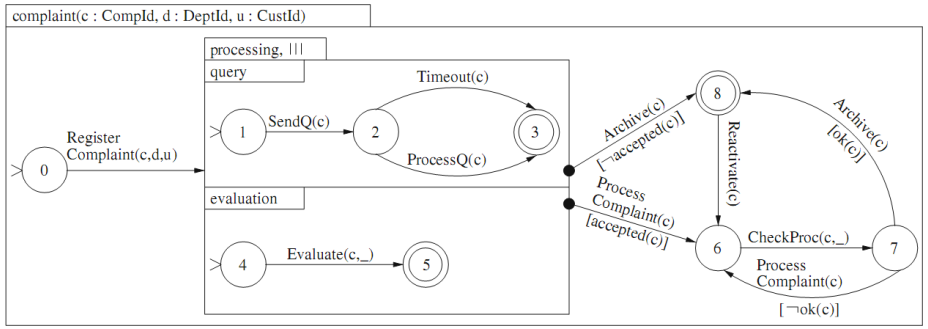


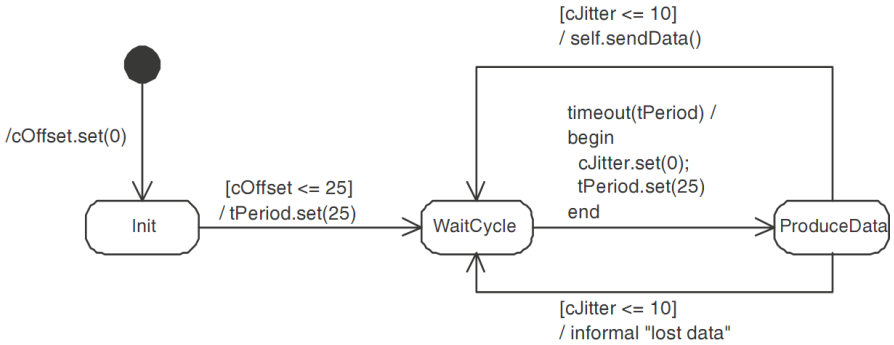




Second and Fourth Papers







- How to combine the two diagrams ?
  - Put them next to each others ?

Remarks

- Theoretically, UML is all about interoperability
- Practically, UML offers no mechanism to define how different models interoperate
- Should we just keep the semantics outside UML ?
  - And use transformation tools from UML to something formal ?
- If no, the *language* to express the interoperability must be:
  - Language independent
  - Technology neutral
  - A meta-MoCC
  - Is Executable UML language the answer ?