

Different notations for describing use case scenarios

Gregor v.Bochmann

School of Information Technology and Engineering (SITE)

University of Ottawa, Canada

This work was supported by a Strategic Grant of the Natural Sciences and
Engineering Research Council of Canada

Presented at the [ECMDA 2005](#) Workshop
A Formal Semantics for UML
November 2005, Nürnberg, Germany



Abstract

UML Use Case Diagrams are a first step towards the definition of system requirements, however, they do not provide enough information for many purposes. In this presentation, I will report on research on requirements engineering performed in Ottawa in relation with the use of UML Activity Diagrams and Use Case Maps (UCM) for defining requirements in terms of what we call "scenarios". This may include any of the following aspects: (a) external actors and system components involved, (b) activities performed, (c) rules concerning the order of activities, (d) object flow, including constraints on object attributes and parameters. We have defined a "Core Scenario Model" (CSM) to capture the common semantics of Activity Diagrams and Use Case Maps. The semantics of this CSM is defined in terms of a translation into (Colored) Petri nets. Translation tools from UCM to CSM and from CSM to Petri nets have been implemented. The former also includes information about performance requirements.



Outline

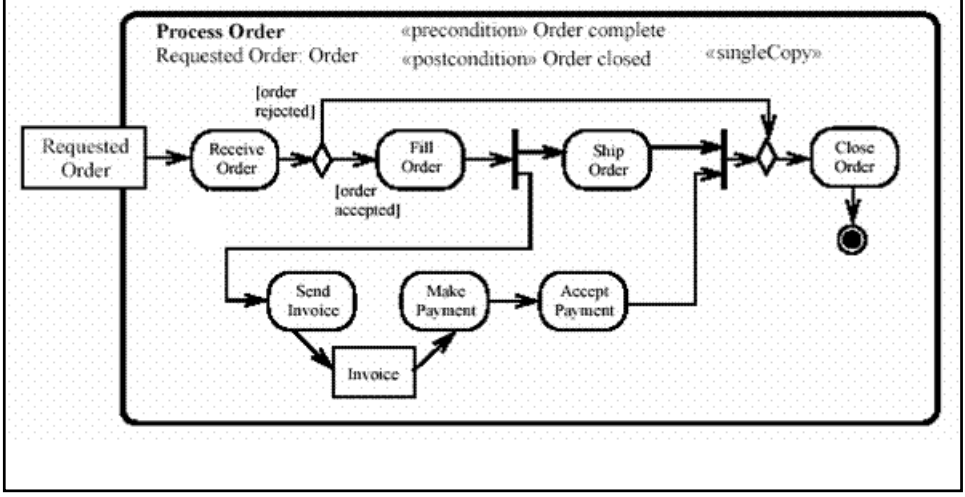
- Activity Diagrams - Use Case Maps: an example
- Aspects of requirements specifications
- The Core Scenario Model
- Petri net semantics
- Limitations
- Related issues
 - BPEL
 - Performance
 - Protocol derivation
 - Transactions



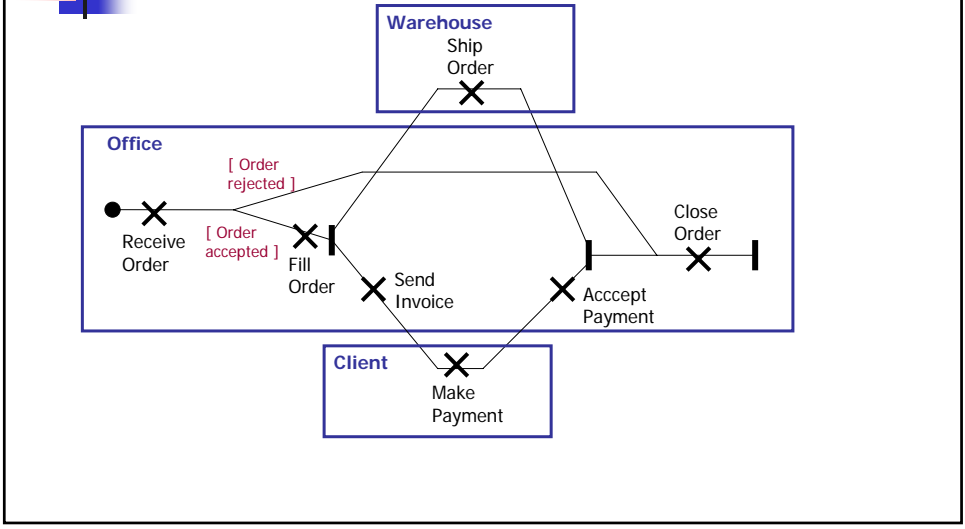
Collaborations

- NSERC Strategic Grant on “Requirements-Driven Development of Distributed Systems” with Daniel Amyot, Luigi Logrippo, Amy Felty, Stephane Somé and Bob Probert
- Collaboration with Murray Woodside’s group at Carleton University (performance aspects in UML)

Example: Activity Diagram



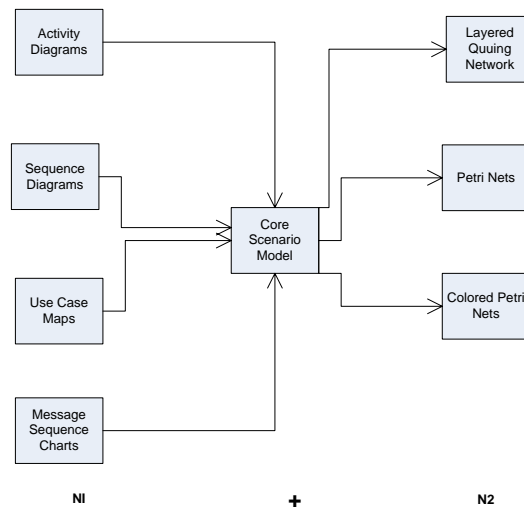
Example: Use Case Map

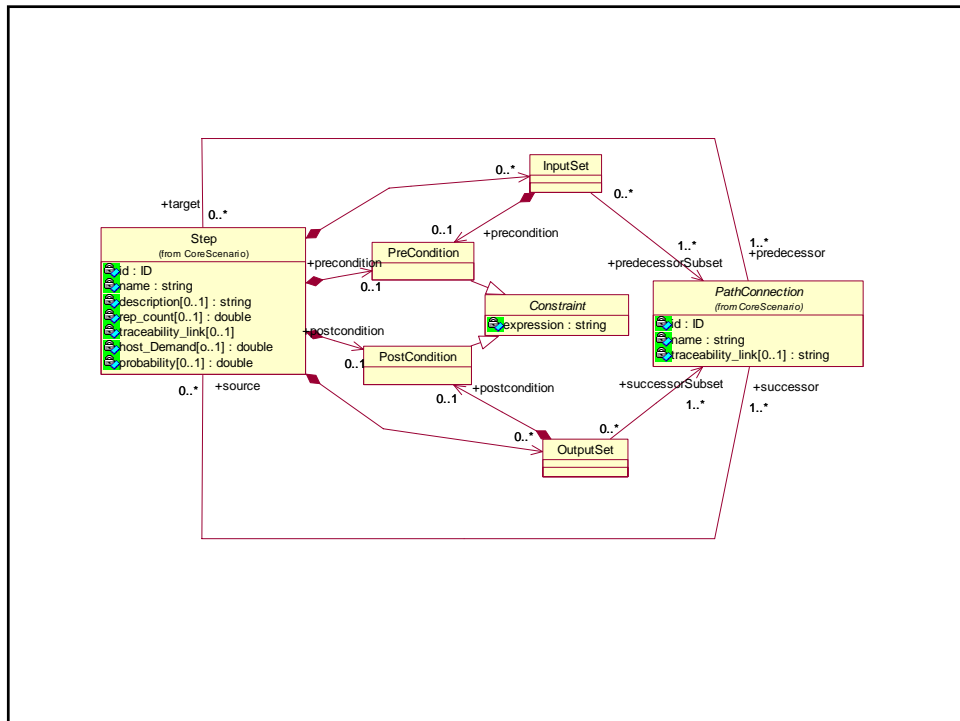
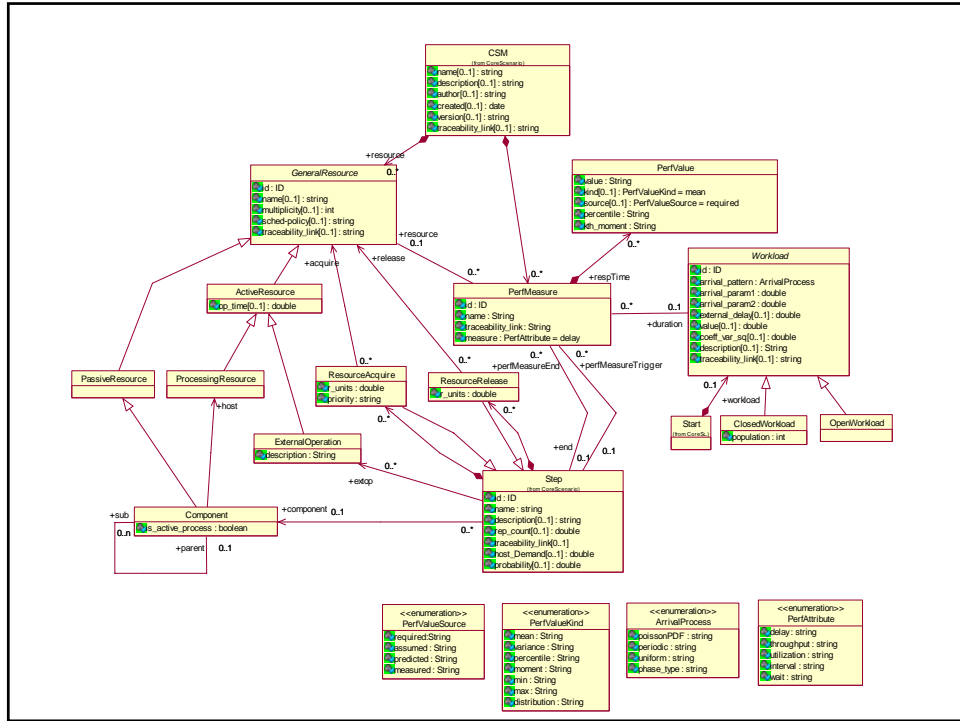


Concepts for requirements

- Each **Use Case** is a scenario
 - **Actions** done by **actors** in some given **order**
- Action: Activity / Responsibility
- Actor: Swim Lane / Component
- Order: sequence, alternative, concurrency, arbitrary control flows (similar to Petri nets)
- **Abstraction**: refinement of activity / Plug-In
- **Data Flow**: Object flow / *not in UCMs*. Question: what type of data is exchanged (an extension of control flow)
 - Input assertions for input data flow
 - Output assertions for output data flow
 - conditions for alternatives (*also in UCMs*)

The Core Scenario Model







Translation of CSM into Petri nets

- The basic concepts are very similar
 - See following table

Shabaz Maqbool built a tool that translates the XML representation of CSM into the XML representation used by Colored Petri nets (CPN Tool)

Concept	Activity Diagram	Petri Nets
Scenario representation	Activity	A Colored Petri Net
Component	Swimlane / Partition	modeled as a Place (see below)
Function and action performed	Action	Transition
Scenario start and stop	Initial Node & Final Node (Flow Final)	A place without any incoming or outgoing edge, respectively.
Alternative scenario	Sub activity	Subpage (in CPN Tools)
Concurrent flows	Fork Node	modeled as a Transition
Alternative flow	Decision Node	modeled as a Place
Sequence flow	ActivityEdge	Arc
Merge of alternatives	Merge Node	modeled as a Place
Merge of concurrent flows	Join Node	modeled as a Transition
Objects (data)	Object Node	modeled as a Place



Some special considerations

- In Petri nets, transitions must alternate with places in a scenario
 - Introduce dummy place between Action, Fork or Join nodes
 - Introduce dummy transition between Decision, Merge, Object, Start and Stop nodes
- Components (Swim Lanes) can be modeled by a place with input and output arcs to all actions within that component (see my earlier work on DMR's method and comparison with UML, 2000, "Activity nets")
- Alternate sets of input or output pins for a given activity can be modeled by several (alternate) transitions



Limitations

- Certain concepts defined for Activity Diagrams are difficult to model with Petri nets, e.g.
 - The semantics of interrupting the processing of an activity, which is used in UML
 - for the activity final node,
 - for interruptible activity region, and
 - for exception handling



Related issues

- BPEL (the Business Process Execution Language used in the context of Web Services) contains also very similar concepts
 - Different graphical notations have been proposed. Why not use Activity Diagrams ?
 - Translation from CSM to PBEL ? – *BPEL as implementation language*



Related issues (ii)

- CSM can also represent semantics of Message Sequence Charts and UML Sequence Diagrams
- Performance notations for UML: work by group at Carleton University
 - Performance extensions to CSM in line with performance notations being elaborated for UML
 - Performance analysis and simulation tools based on extended CSM



Related issues (iii)

- Obtaining distributed system designs automatically
 - Our work on protocol derivation from service specifications (based on service specifications in the form of process algebras or Petri nets) can be used to derive the protocol for a distributed system based on given requirements in the form of a CSM.
 - We are presently working on integrating the concept of transactions into the scenario notation.



Conclusions

- Different notations for requirement scenarios have similar concepts
 - Depending on the context, one wants to high-light certain aspects (e.g. actions and control flow, components, or object flow); Activity Diagrams and Use Case Maps are complementary
- Concepts of sequence diagrams also very similar
- A common Core Scenario Model is useful for defining semantics for these notations and for building tools
- . . . I am interested in more exchanges with people working on these topics