Precise Action Semantics for UML

Stephen J. Mellor
Ian Wilkie

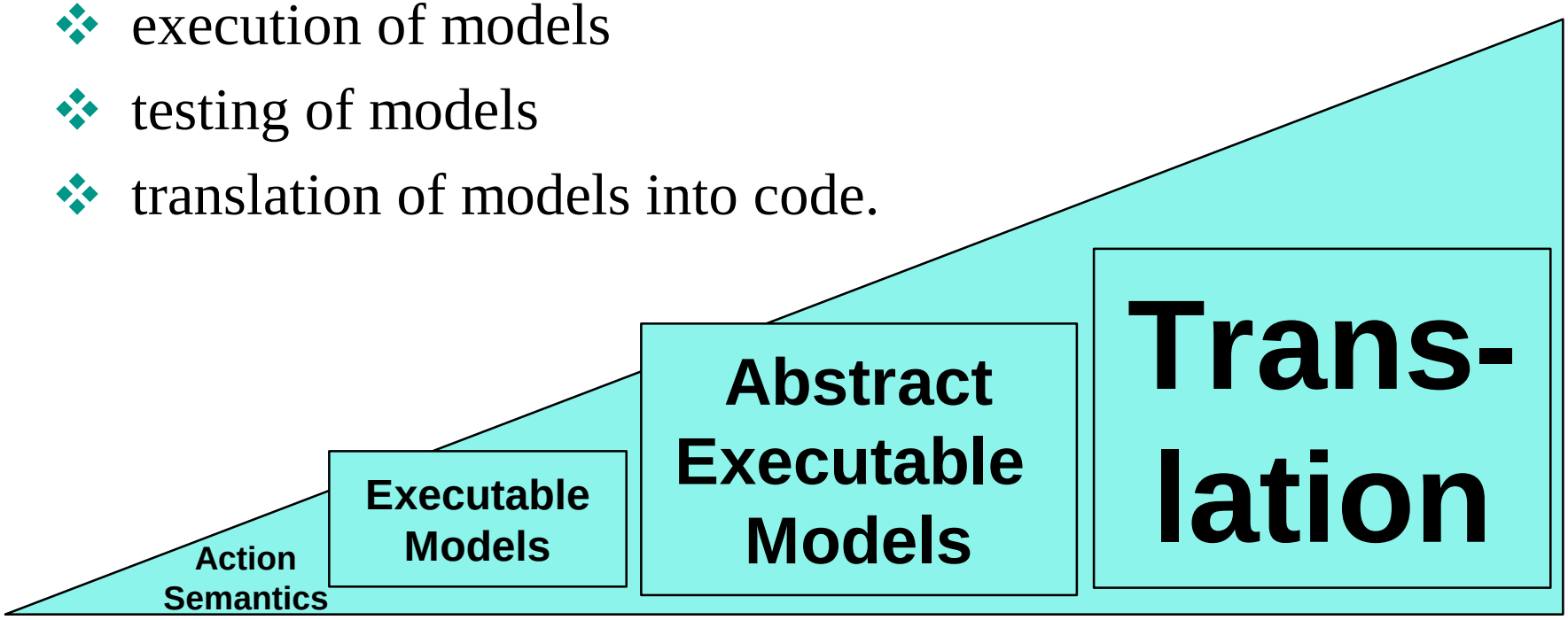




Benefits of Action Semantics

Action semantics and executable models enable:

- complete specification
- precise specification
- early verification with users
- execution of models
- testing of models
- translation of models into code.



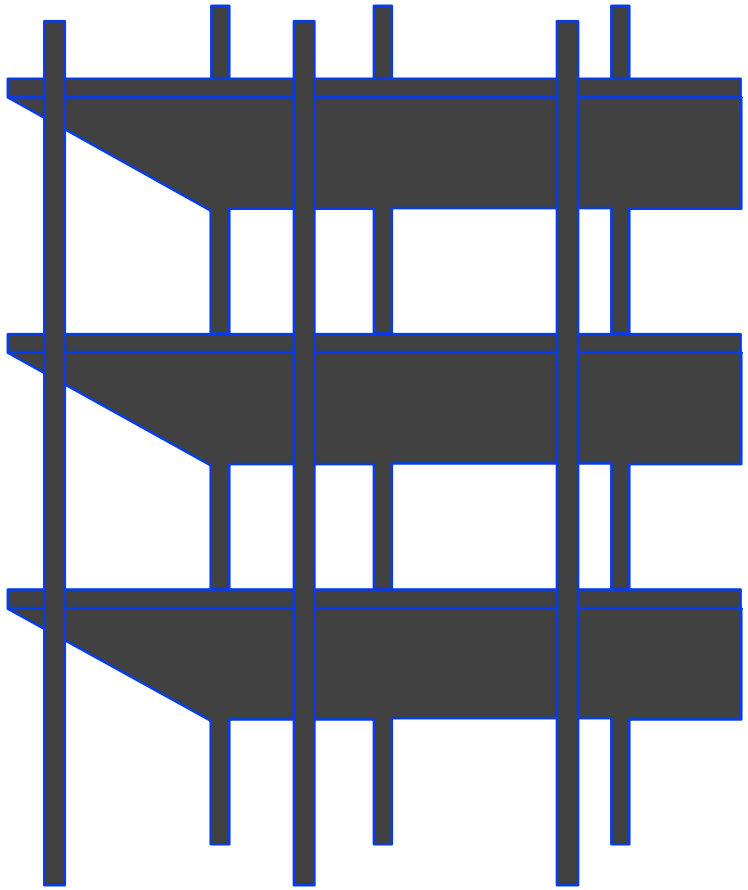
Three Levels

KENNEDY CARTER

PROJECT TECHNOLOGY, INC.

To have complete models, we need to specify semantics at three levels:

- State Chart and Operations
- Actions as a whole
- Individual processes



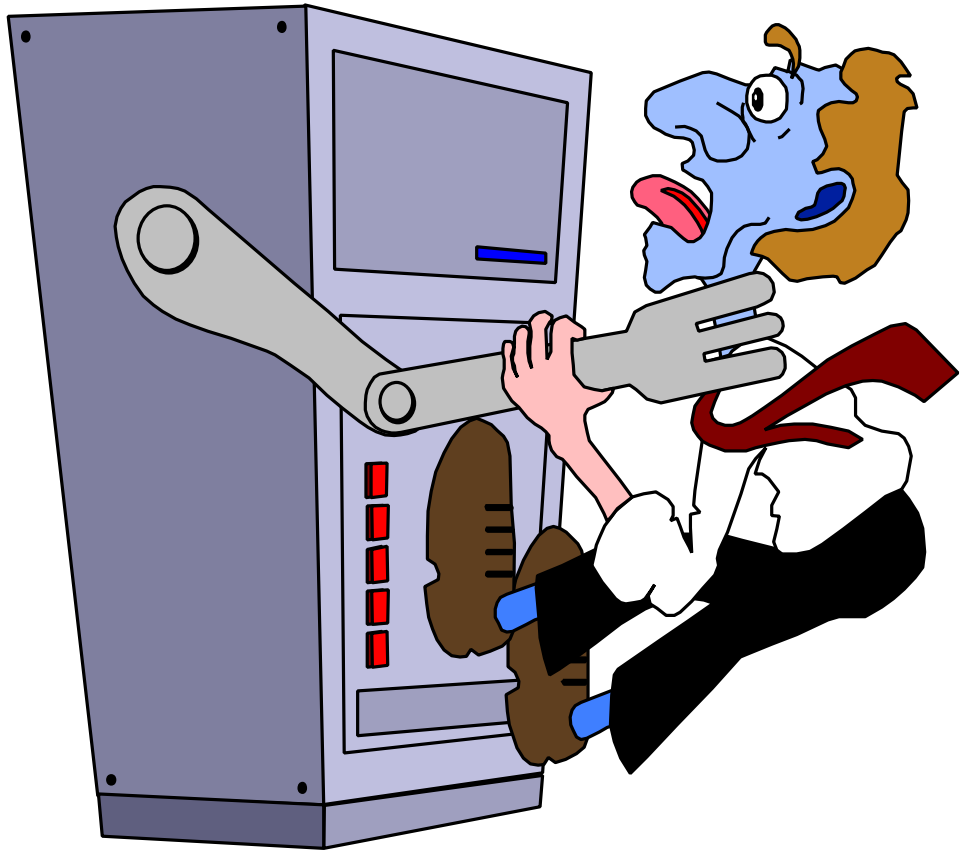
State Charts and Operations

What are interactions between the:

- asynchronous behaviour on a state chart,
- synchronous behaviour on a state chart

and

- synchronous behaviour of operations?



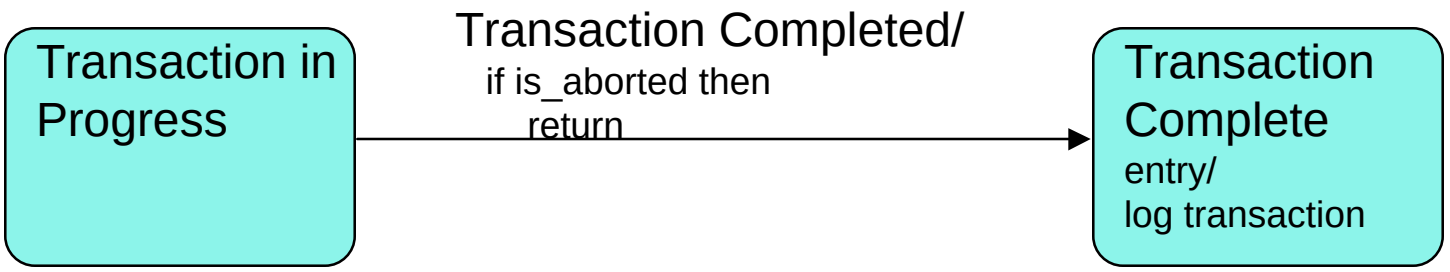
The UML Execution Engine

UML already has an implicit “execution engine” that governs the behaviour of models

- e.g. event processing defined in terms of queues

This engine is known to be incomplete, and inconsistent

- e.g. what happens if you execute a “return” in a transition action stimulated by “call” (synchronous) event ?



The UML Execution Engine

UML aims to cover all Object Oriented methods and paradigms....

.... strong resistance to restricting the execution to only one style

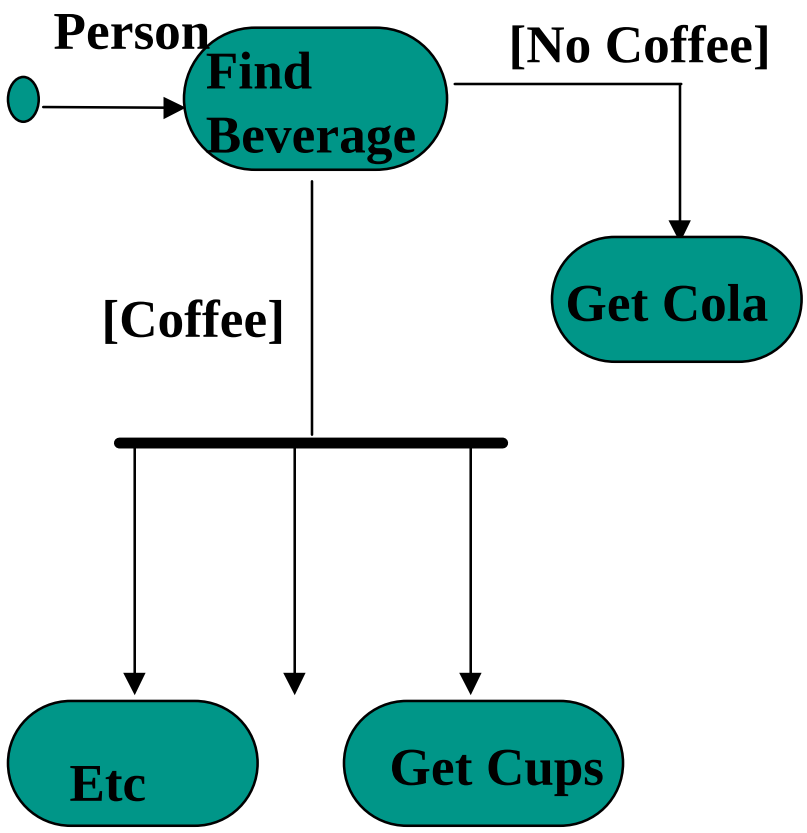
- "Execution Profiles" will address this issue
- Precise Action Semantics must interface with these ideas

Actions

The action-as-a-whole is concerned with:

- flow of information
- flow of control
- concurrency
- declaration of processes

The lowest level is concerned with the details of the processes themselves



Existing “Action” features in UML 1.3

UML 1.3 already has some “actions” defined:

- CreateAction
- CallAction
- AssignmentAction
- ReturnAction
- SendAction
- TerminateAction
- UninterpretedAction
- DestroyAction

but, no “control structures” ... at least as part of Actions

Problems with existing Actions in UML



PROJECT TECHNOLOGY, INC.

Incomplete

- No control structures
- No explicit semantics for local variables
- No parallel behaviour
- Cannot describe methods

Potentially inconsistent

- ReturnAction (UML 1.3 vs Rumbaugh)

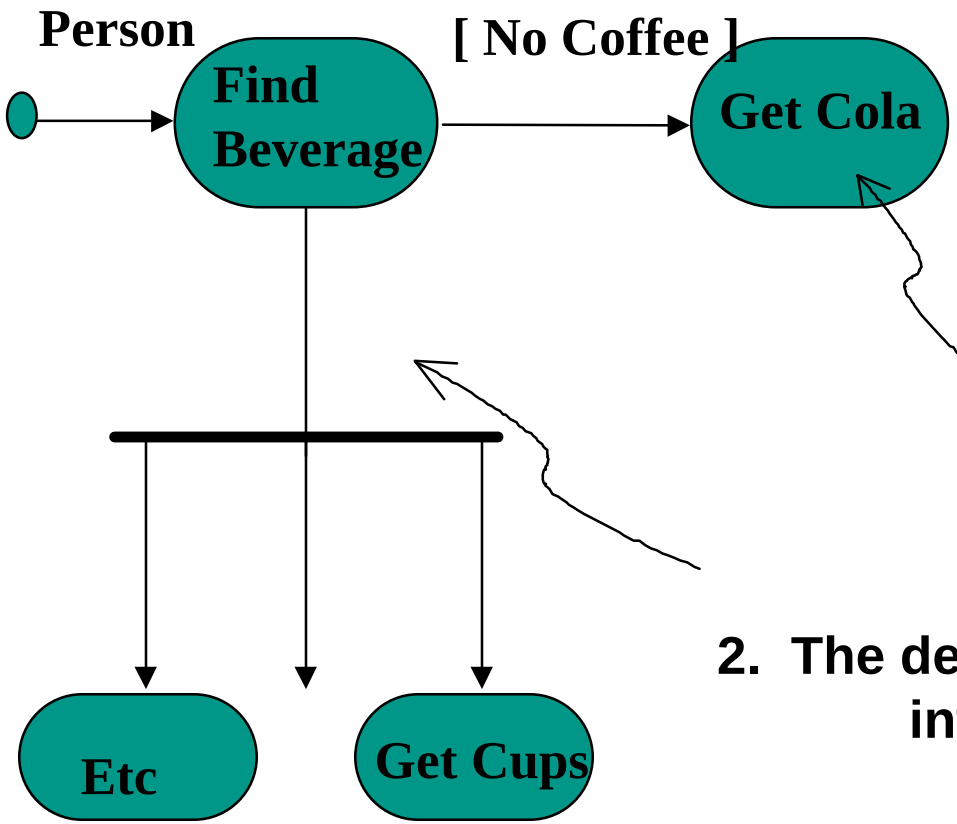
Subtleties are glossed over

- When exactly does the “frozen” property come into effect ?

Actions spread throughout the UML specification

- “deferred event” requires a “defer” action

The Three Levels



1. The definition of when this executes.

3. The definition of the activities themselves

2. The definition of how the activities interact (the “lines”).

Possible Solution Components

What is available to solve the problem?

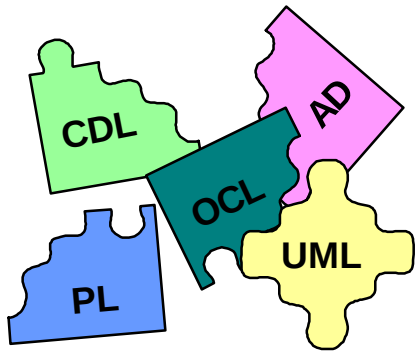
UML graphics?

OCL for data access?

Activity Diagrams?

CDL?

Programming Languages?





Why not just use OCL ?

OCL is a *constraint language*, intended for the specification of static assertions about the model

OCL was not conceived for, and lacks the constructs for, specifying execution

Even if OCL is used to specify processing by implication (by using pre and post conditions), this does not address the central problem of the semantics

..... OCL may provide the basis for a language and a syntax

The OMG Timetable

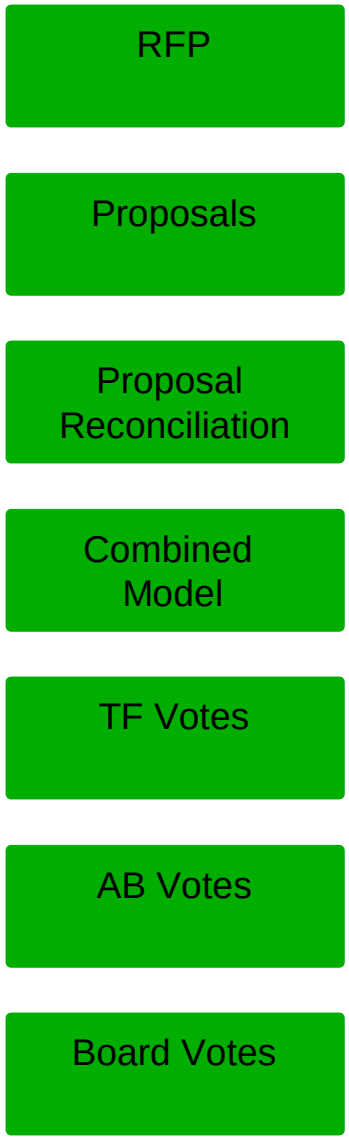
The Object Management Group is the standards body for the UML

Version 1.3 of the UML finalised in June 1999

The OMG issued a request for proposals on Precise Action Semantics in November 1998

Submission to be delivered in February 2000

OMG will vote in November 2000



Deliverables from the Process

Extended (and modified) UML meta-model, with class descriptions

Semiformal execution engine specification

Well defined semantic variation points to support Execution Profiles

At least one example syntax for a language based on Precise AS

.... this work is not standardising a language

The consortia

There are two consortia proposing to submit

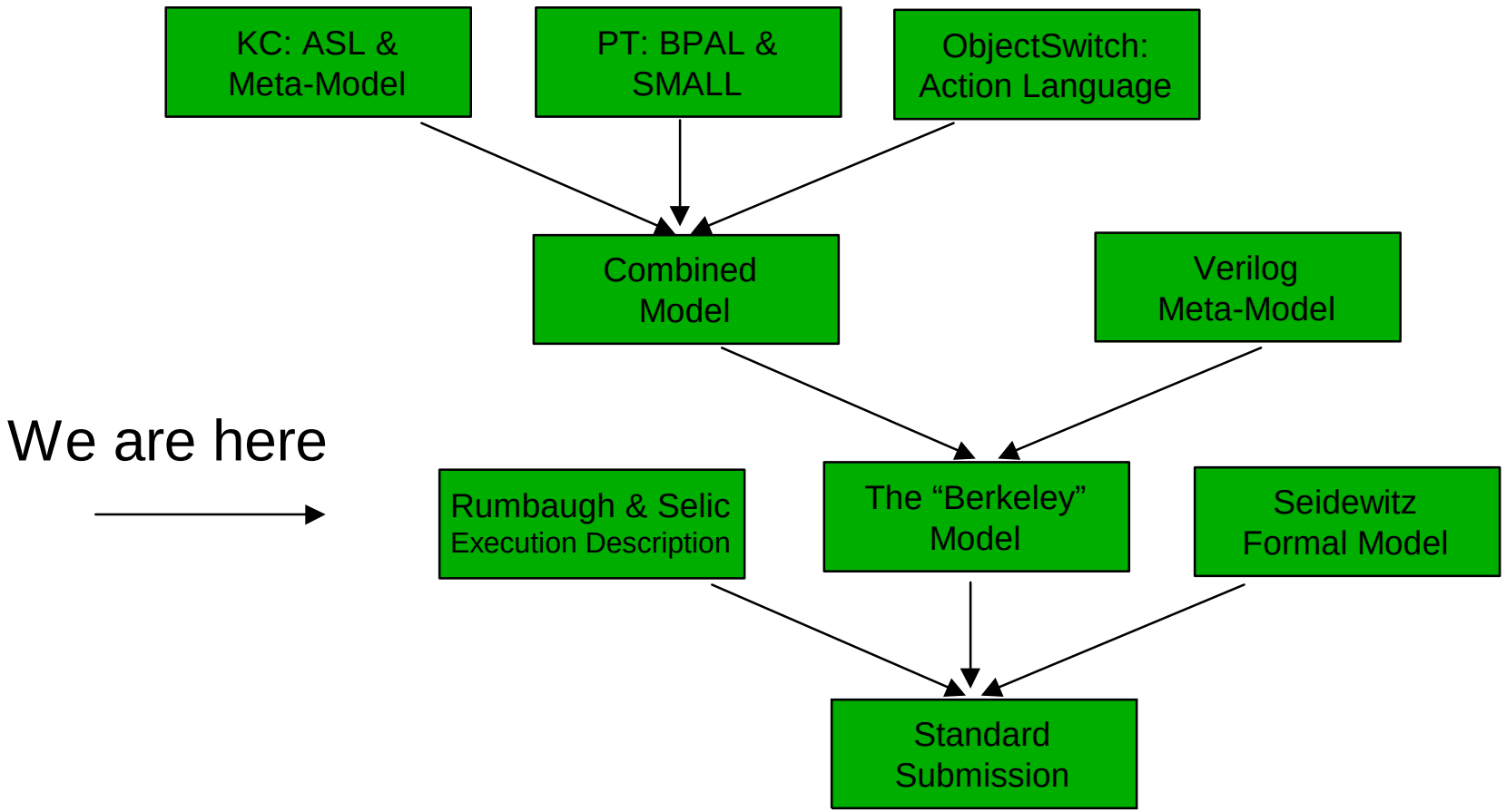
Larger consortium consists of:

- PT/KC/ObjectSwitch/T-Vec/Verilog/Rational/ObjecTime
- AlcaTel/Simware/Telelogic/Software Productivity Consortium/I-Logix/SHL/DHR/Booz-Allan & Hamilton/ROX Software
- http://uml.simware.com

Smaller consortium based around:

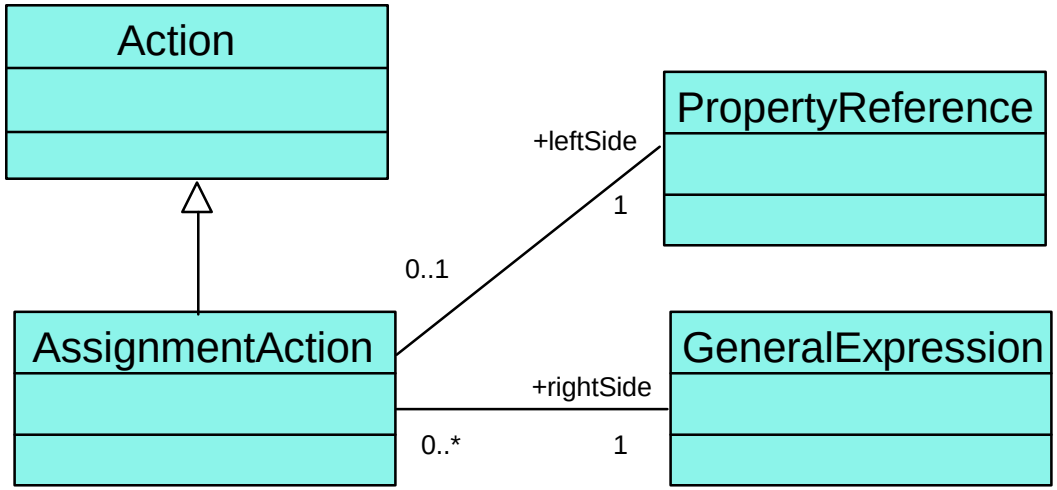
- DSTC, Brisbane
- http://www.dstc.edu.au

History of the work so far ....



The “Berkeley” Model

This is a meta-model of the static structure of the Action part of a UML model

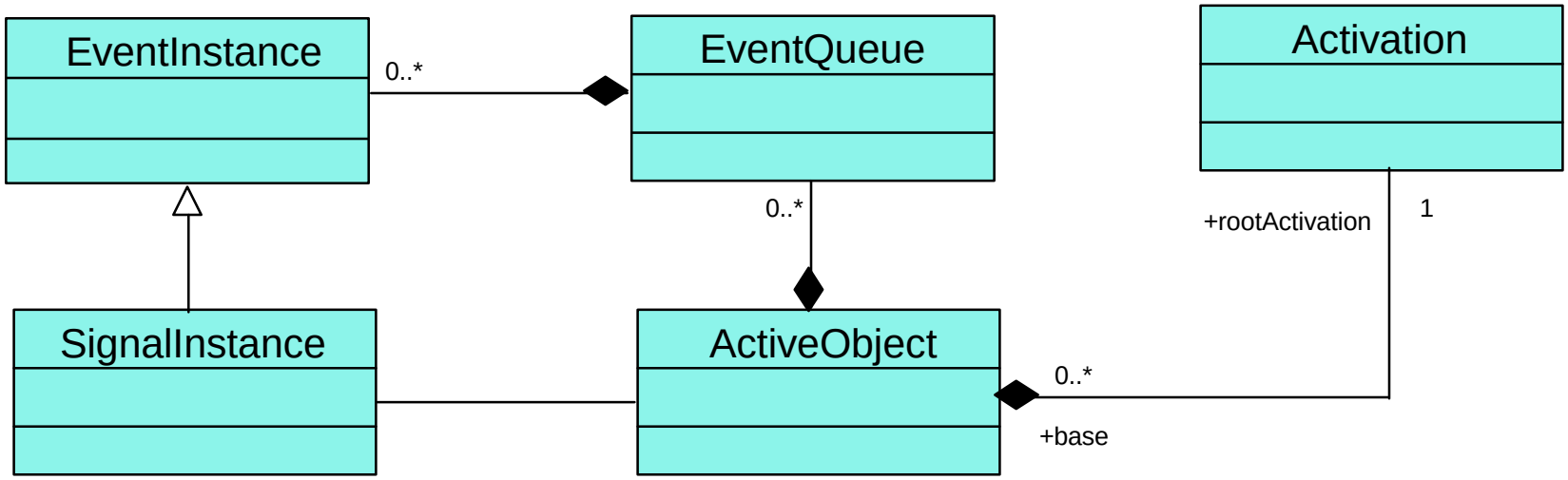


Effectively an extension to the current UML meta-model

Execution semantics are defined textually in the class descriptions

Rumbaugh and Selic Execution Model

Defines a framework upon which to describe threads of execution and their interaction



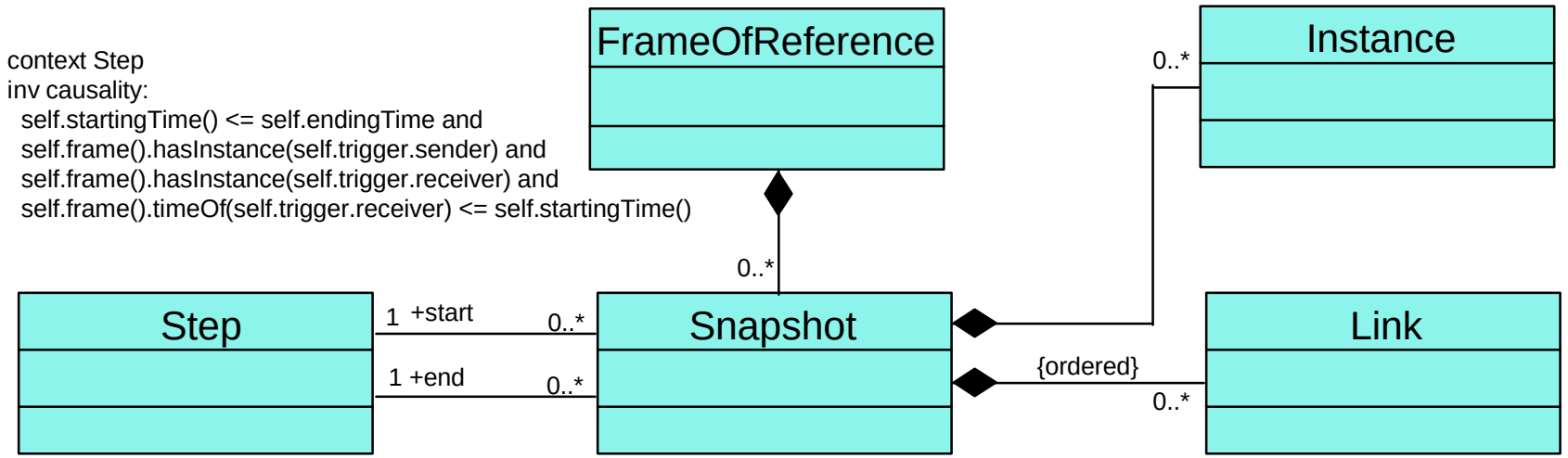
The emphasis is on a textual description of behaviour

The root activation attached to an active object may perform a receive action in which it waits for an input event to be present in on the queue

Seidewitz Formal Model

Defined through a meta-model of run-time UML

Run-time behaviour specified by OCL constraints on the instances in the run-time model



Given Shlaer-Mellor like constraints (limited assumptions about ordering and simultaneity) can prove Frame invariance of execution results

Outstanding Issues

KENNEDY CARTER

PROJECT TECHNOLOGY, INC.

There are a number of issues to deal with before submission can be made:

- The meta-model is still incomplete
- The two execution models must be reconciled with each other and integrated with the Berkeley meta-model
- Activity Diagrams must be reconciled with these models
- Exceptions have not yet been fully considered

Conclusions

The UML community appreciates the need for precise action semantics

A proposal is being developed with backing from the key players

Scheduled to become a standard by the end of 2000