UML and Concurrency

Perdita Stevens

Division of informatics

University of Edinburgh

http://www.dcs.ed.ac.uk/home/pxs

Plan

- Intro: what is covered by “concurrency”?
- concurrency in activity diagrams: UML 1.x
- issues and formal semantics
- forthcoming: activity diagrams in UML2.0
- some (still) open issues

Pre-intro: what is UML?

Unified Model[l]ing Language

OMG standard since 1997; first major revision in progress [more later]

Multi-view diagrammatic language for describing the design of [OO software] systems

Compromise

Widely adopted, partially supported by many tools

What is concurrency?

In UML world, often conflated with

- real-time issues
- non-determinism, however caused

In TCS, often conflated with non-determinism, independence, temporal specification or process algebra.

Naive view: concerns two things happening at once.

Concurrency in UML

There are concurrency considerations of some kind in just about every UML diagram type, and in OCL.

The paper in the proceedings gives a survey and pointers.

Here we focus on activity diagrams, which are probably the most interesting diagram type from a concurrency point of view.

Activity diagrams

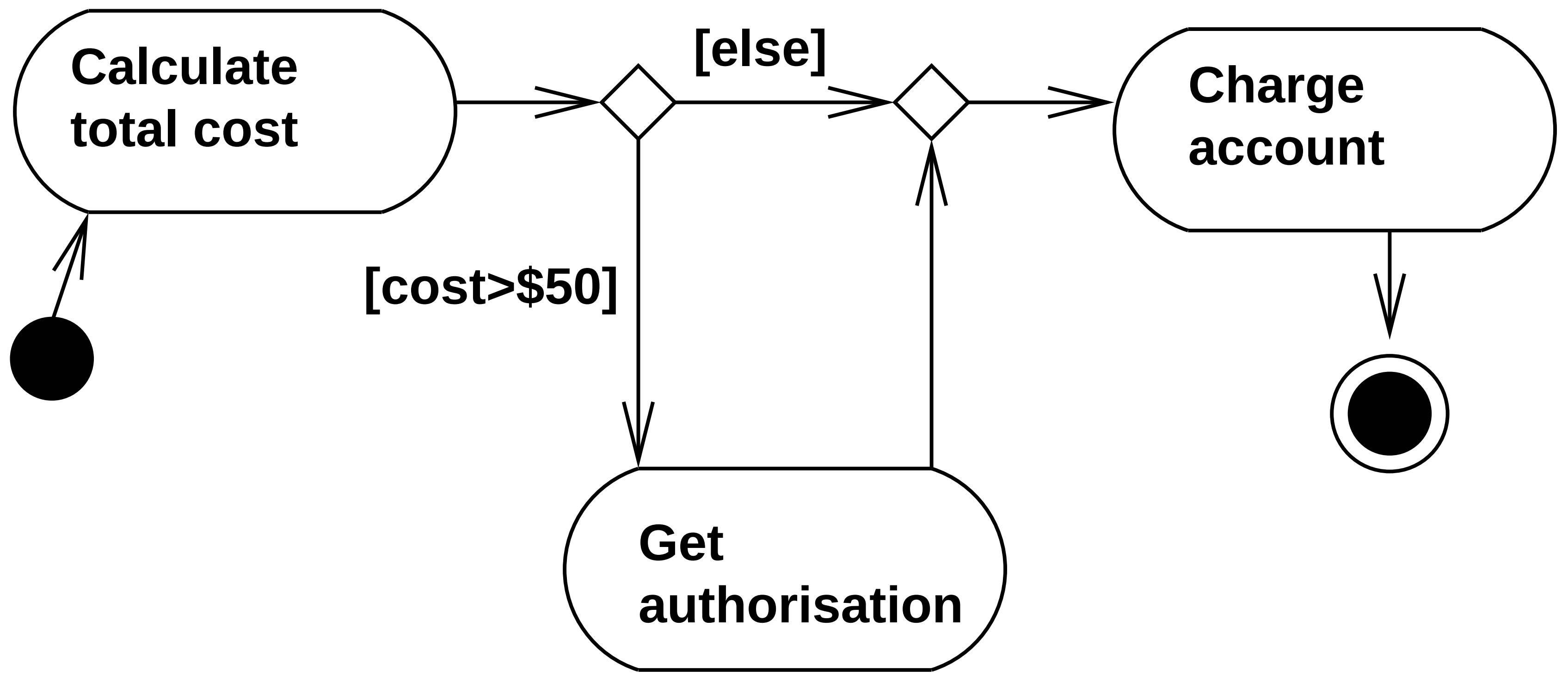
Focus on dependencies between the activities that go to make up some significant chunk of system behaviour, e.g.

- carrying out of a use case [task that the system helps to carry out]
- explaining an an object's reaction to a message

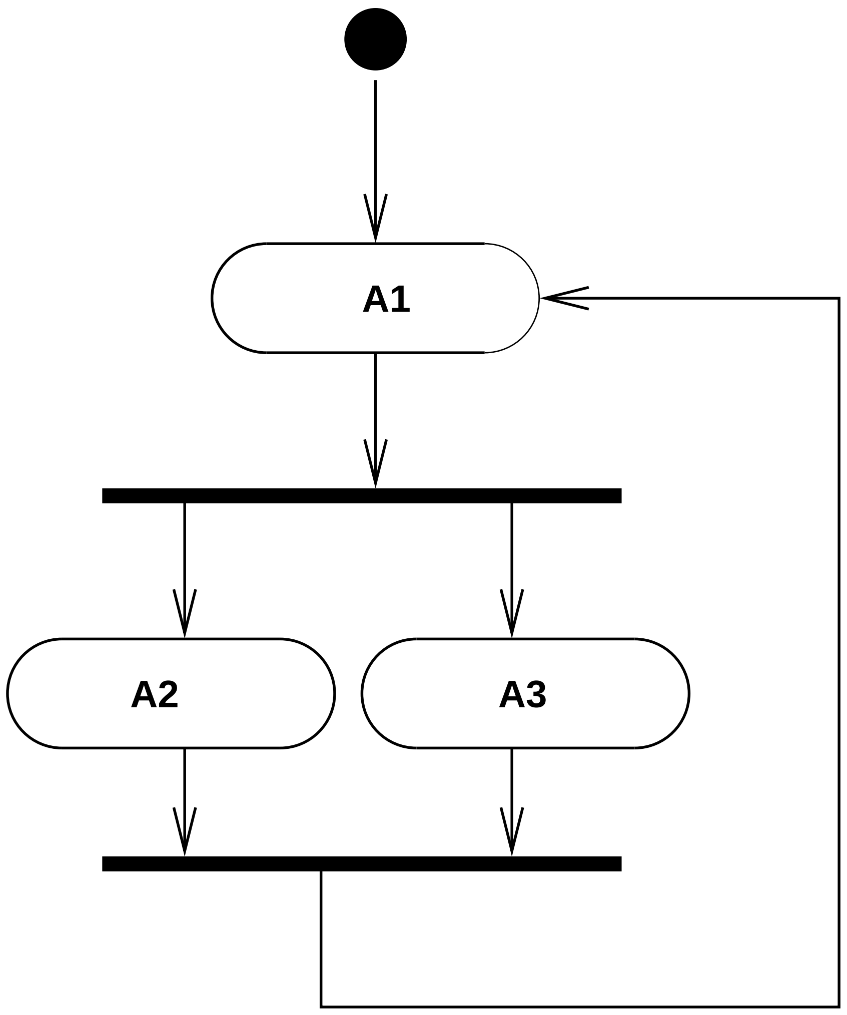
Also useful for showing the dependencies between use cases: e.g. *workflow* e.g. of an organisation.

By focusing on dependencies, activity diagrams help avoid making premature decisions about ordering of events.

Basic “flow chart” activity diagram



Concurrency in activity diagrams



Semantics of activity diagrams

It was natural to think of UML1.x activity diagrams as being a kind of Petri net, and many people including authors and tool developers did so.

It was just about possible to defend this from the UML definition...

... but this was not the intention of at least some of its authors. Instead activity diagrams were intended to have concurrency no more expressive than state diagrams.

Semantics of activity diagrams

Currently, activity diagrams are seen as a kind of state diagram...

n-fork = enter state with n concurrent substates

n-join = leave state with n concurrent substates

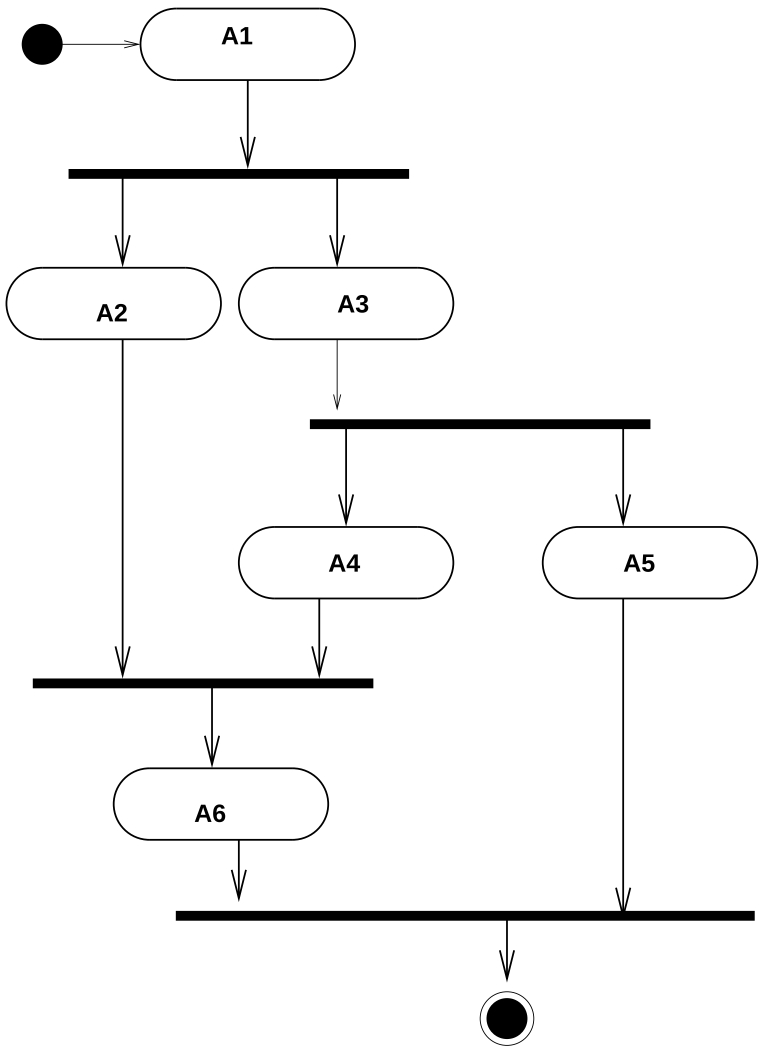
This severely limits the expressiveness of activity diagrams and in practice the limitation is often ignored...

The limitation is often ignored...

...including by the UML1.4 authors.

p3-158, OMG doc formal/01-09-67

“Cross synchronisation”



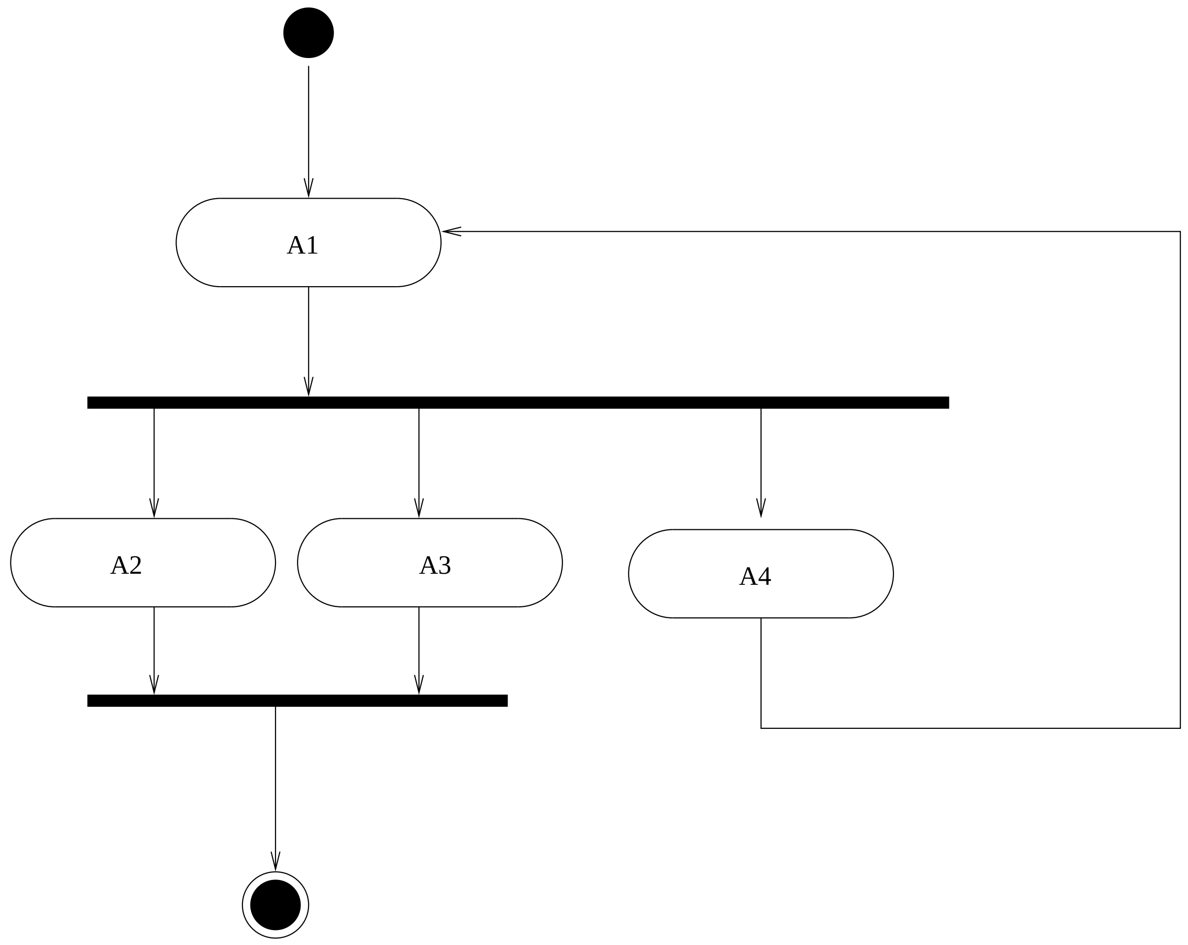
So we want more expressiveness

But how much, and of exactly what kind?

Easiest: one-safe Petri nets, static structure (no “instances” of activities).

But sometimes we can make more or less convincing arguments for wanting more than this. E.g.

Is this sensible?



Formal motivated work on activity diagrams

Work includes

Börger Cavarra Riccobene: An ASM semantics for UML activity diagrams

Bolton Davies: Activity graphs and processes

Eshuis Wieringa: A real-time execution semantics for UML activity diagrams

Petriu Wong: Using activity diagrams for representing concurrent behaviour

UML2.0

Current state: “final” proposals were submitted January 2003.

Voting and adoption process about to begin.

Largest group of proposers: U2P

We’ll discuss activity diagrams as revised in their proposal.

U2P

Submitters: Alcatel, Computer Associates, Enea Business Software, Ericsson, Fujitsu, Hewlett-Packard, I-Logix, International Business Machines, IONA, Kabira Technologies, Motorola, Oracle, Rational Software, SOFTEAM, Telelogic, Unisys, and WebGain

Supporters: Advanced Concepts Center LLC, Ceira Technologies, Commissariat á L'Energie Atomique, Compuware, DaimlerChrysler, Embarcadero Technologies, France Telecom, Fraunhofer FOKUS, Gentleware, Intelli-corp, Jaczone, Kennedy Carter, Klasse Objecten, KLOCwork, Lockheed Martin, Mercury Computer, MSC.Software, Northeastern University, Popkin Software, Proforma, Sims Associates, Sun Microsys-tems, Syntropy Ltd., Thales Group, University of Kaiserslautern, VERIMAG, and 88solutions

U2P Superstructure proposal

Incremental development of UML1.x.

Additional notation (e.g. to support component based architecture)

Language definition similar to UML1.x but (if anything) less precise:

- constraints expressed in English, not OCL.
- no explicit “mapping” section to decode concrete syntax

Activity diagrams in U2P UML2.0

p237 - 330 !

Connection with state diagrams abandonned

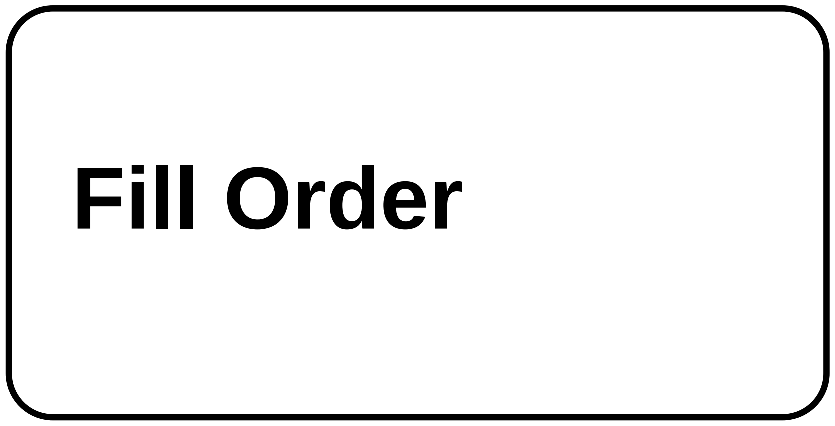
New semantics based on Petri nets with queuing

Concurrent structure separated from hierarchical structure

Object flow via input/output pins

Basics: actions

“Primitive” indecomposable (but not atomic) behaviour chunk



i.e. finally, now that these have nothing to do with states, they are notated the same way as states :-)

Basics: activities

“Complex” behavior, coordinating actions and possibly other activities.

p262 of U2P proposal.

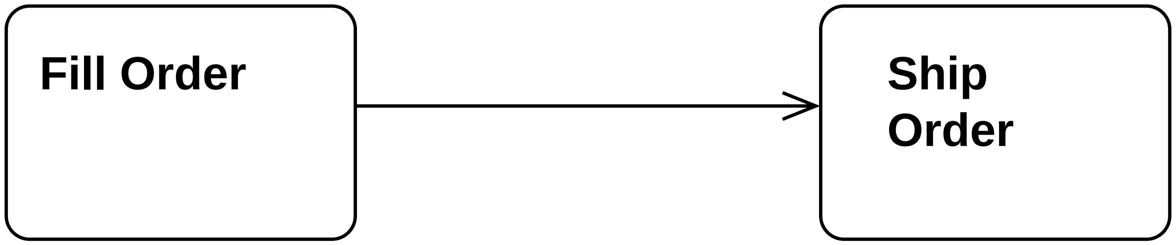
Token flow 1

Tokens may be objects, data or loci of control

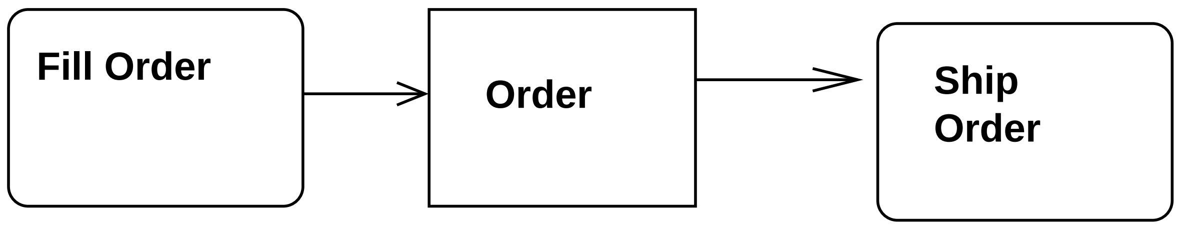
Objects and data usually treated identically.

However, there is a distinction between *control flows* and *object flows*

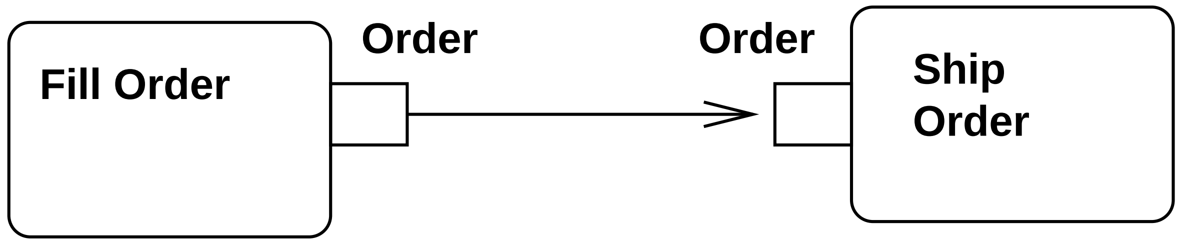
Control flow example



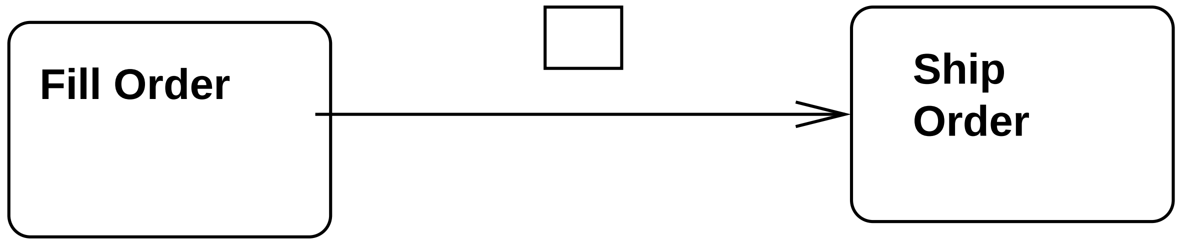
Object flow example 1



Object flow example 2



Object flow example 3



Token flow 2

Source node, target node and edge may all have rules about when token flow is permitted - the token flows only if all are satisfied.

An edge may have a weight – if *at least* that number of tokens are available at the source, *all* are offered to the target.

“Multiple tokens containing the same value may reside in the object node at the same time.” (p313)

Semantic issues 1

Interleaving vs true concurrency?

Need a consistent dynamic semantics for a model, reflected in activity diagrams and state diagrams, interaction diagrams,...

(True concurrent semantics natural for activity diagrams... but rest of UML much more naturally treated by interleaving. So...?)

Semantic issues 2

Dynamic structure: treatment of multiple invocations of the same activity

Activity::isSingleCopy defaults to false – no interaction between tokens from different invocations.

(Maybe this only needs checking? Is there still an issue here or not?)

Semantic issues 3

ActivityNode and ActivityEdge inherit from RedefinableElement.

This means they can be redefined in the context of a specialising classifier (e.g. should give relationship between activity diagram specifying A::foo and B::foo where B $<$ A)

A metamodel operation isConsistentWith for each should have been included... but curiously, is missing right now :-)

Oddities

If the mustIsolate flag is true for an activity node, then any access to an object by an action within the node must not conflict with access to the object by an action outside the node. A conflict is defined as an attempt to write to the object by one or both of the actions. If such a conflict potentially exists, then no such access by an action outside the node may be interleaved with the execution of any action inside the node. This speci-fication does not constrain the ways in which this rule may be enforced. If it is impossible to execute a program in accordance with these rules, then the moisdel is ill formed.

Esoterica

1. streaming input/output
2. exceptions (an Action may raise an exception, which may be handled by an ExceptionHandler node)
3. local pre/post conditions for Actions
4. activities may support operations e.g. start, stop, abort; queries, e.g. how long has this activity been operating for (Workflow...)
5. InterruptibleActivityRegions (with dedicated interrupting edges)
6. ActivityFinalNode, FlowFinalNode
7. ActivityPartitions (replacing swimlanes)

8. CentralBufferNode (allowing object queueing) and DataStore
9. ConditionalNodes (with test clauses allowed to have side effects!)
10. LoopNodes
11. ExpansionRegions with ExpansionKinds (parallel, iterative, stream) (replacing dynamicConcurrency etc. for operating on collections)
12. ObjectFlow::isMulticast/isMultireceive (fig 6-92)

Conclusion and challenge

We have discussed activity diagrams, in which the most interesting aspects of concurrency in UML appear, both as they are now and as they may be in UML2.0.

Your mission, should you choose to accept it

- Can you use ASMs to expose *and solve* problems?
- Can you use ASMs to provide *really useful* tool support?
- Can you use ASMs to make any other significant contribution?

If so...

Advertisements

PITPAT Eindhoven Jun 30 - Jul 4 (Arend Rensink & PS)

Submission date: 21 March

UML San Francisco Oct 20-24

Submission dates: 8 Apr (abstracts) 15 Apr (full/short papers)

FMOODS Paris Nov 3-7 (Uwe Nestmann & PS)

Submission dates: 2 Jun (abstracts) 9 Jun (papers)