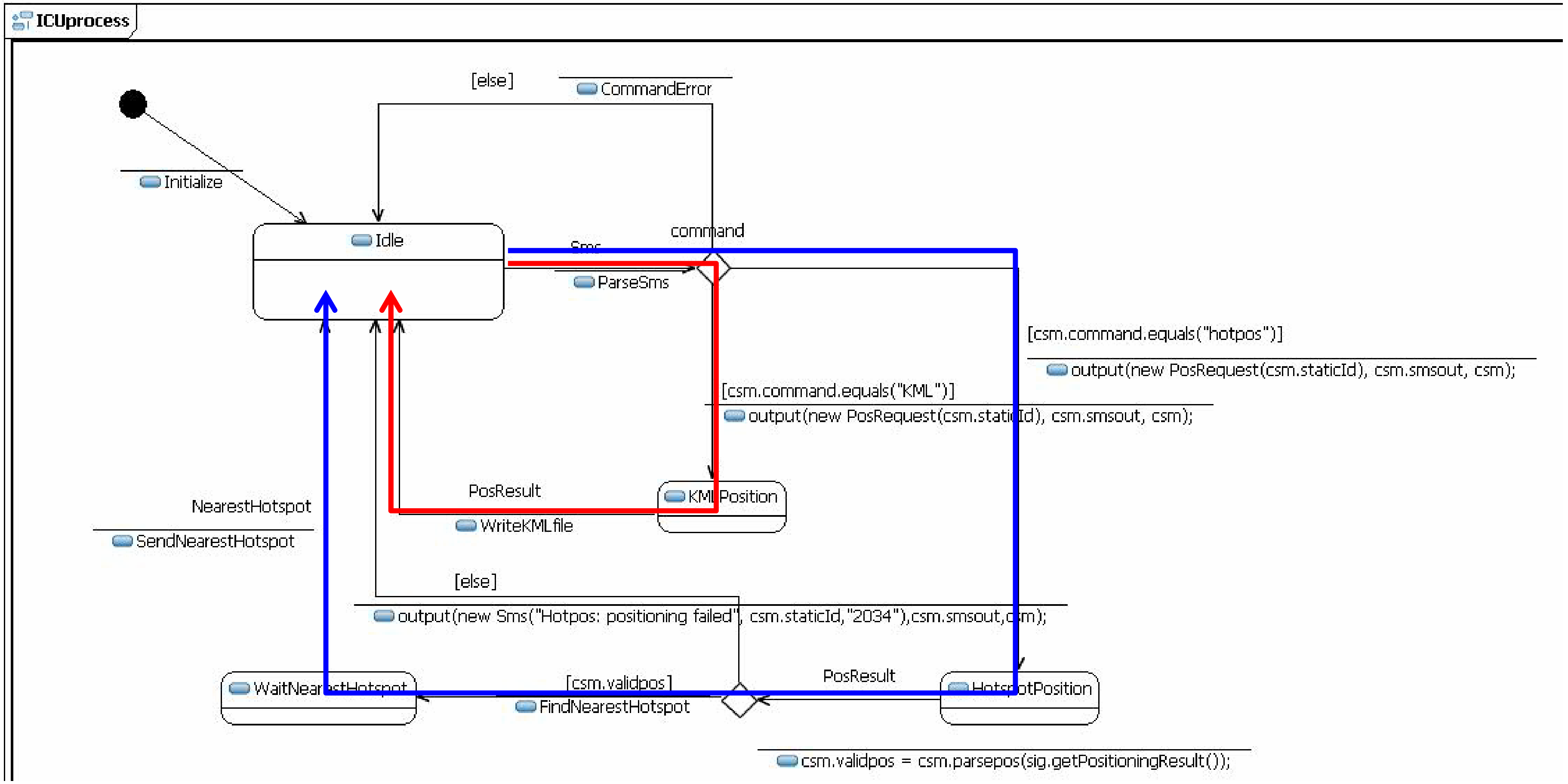
Services as Submachine States
Service instantiations as concurrent parts

Version 071102

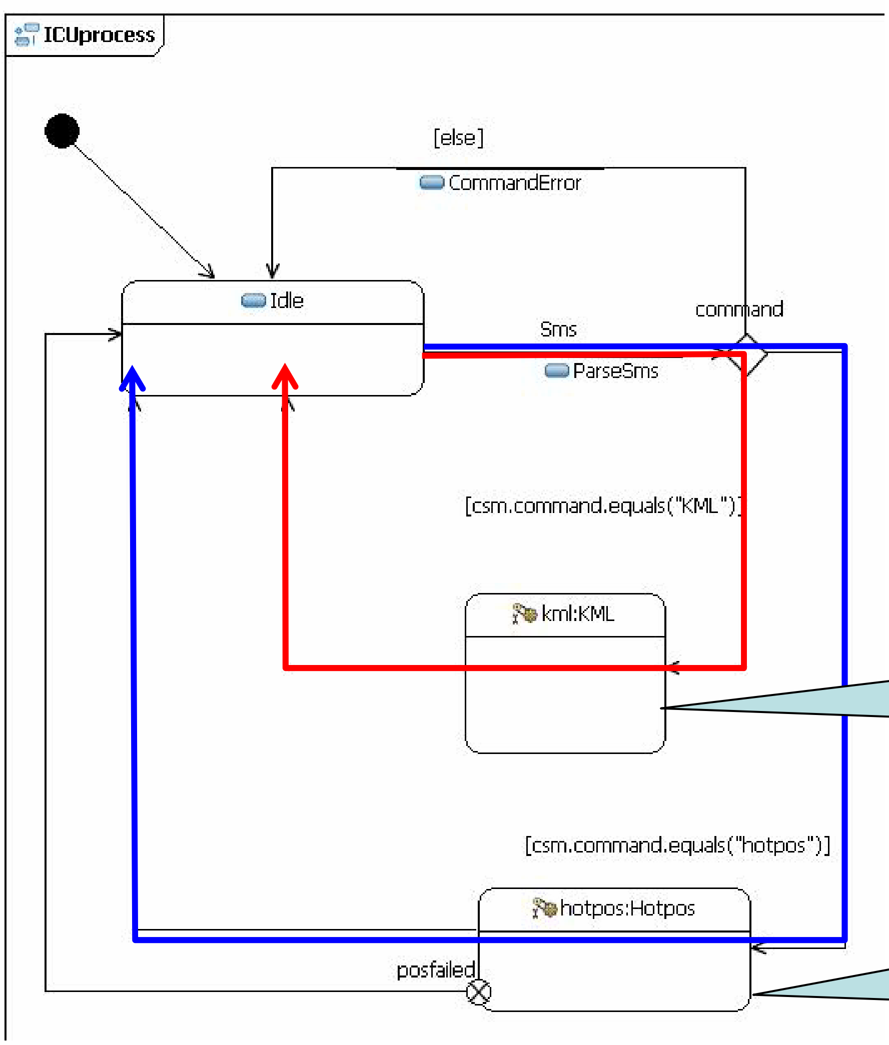
ICUprocess serving 2 services


ICUprocess
[else]
CommandError
Initialize
Idle
Sms
command
ParseSms
[csm.command.equals("hotpos")]
output(new PosRequest(csm.staticId), csm.smsout, csm);
[csm.command.equals("KML")]
output(new PosRequest(csm.staticId), csm.smsout, csm);
PosResult
KMLPosition
WriteKMLfile
NearestHotspot
SendNearestHotspot
[else]
output(new Sms("Hotpos: positioning failed", csm.staticId,"2034"),csm.smsout,csm);
WaitNearestHotspot
[csm.validpos]
FindNearestHotspot
PosResult
HotspotPosition
csm.validpos = csm.parsepos(sig.getPositioningResult());

Separation of Concerns

- Isolate reusable functions
  - through operation/method: *parsepos* and *deccoords*
- Separate independent concurrent tasks
  - through parts in composite structures: *icuproc* and *dataproc*
- Separate different alternating services
  - through submachinestates of internal state machines
  - *KML* and *Hotpos*
  - We have introduced the following invariant:
    - One user (defined by one mobile telephone) can only be involved in one (top level) service at one instant

ICUprocess with 2 submachine states



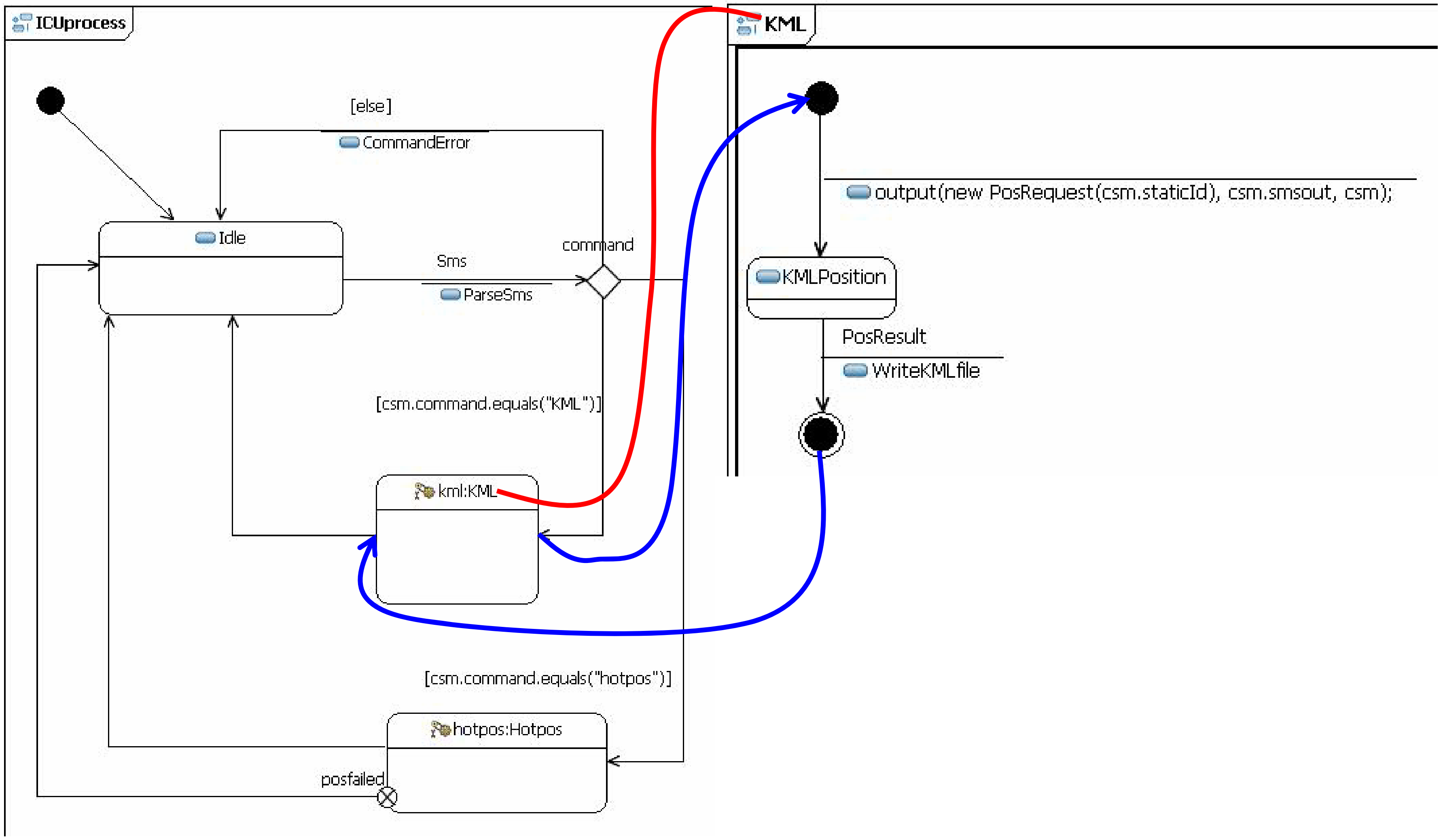
These state machines are not concurrent!

Each submachine state refers to a state machine

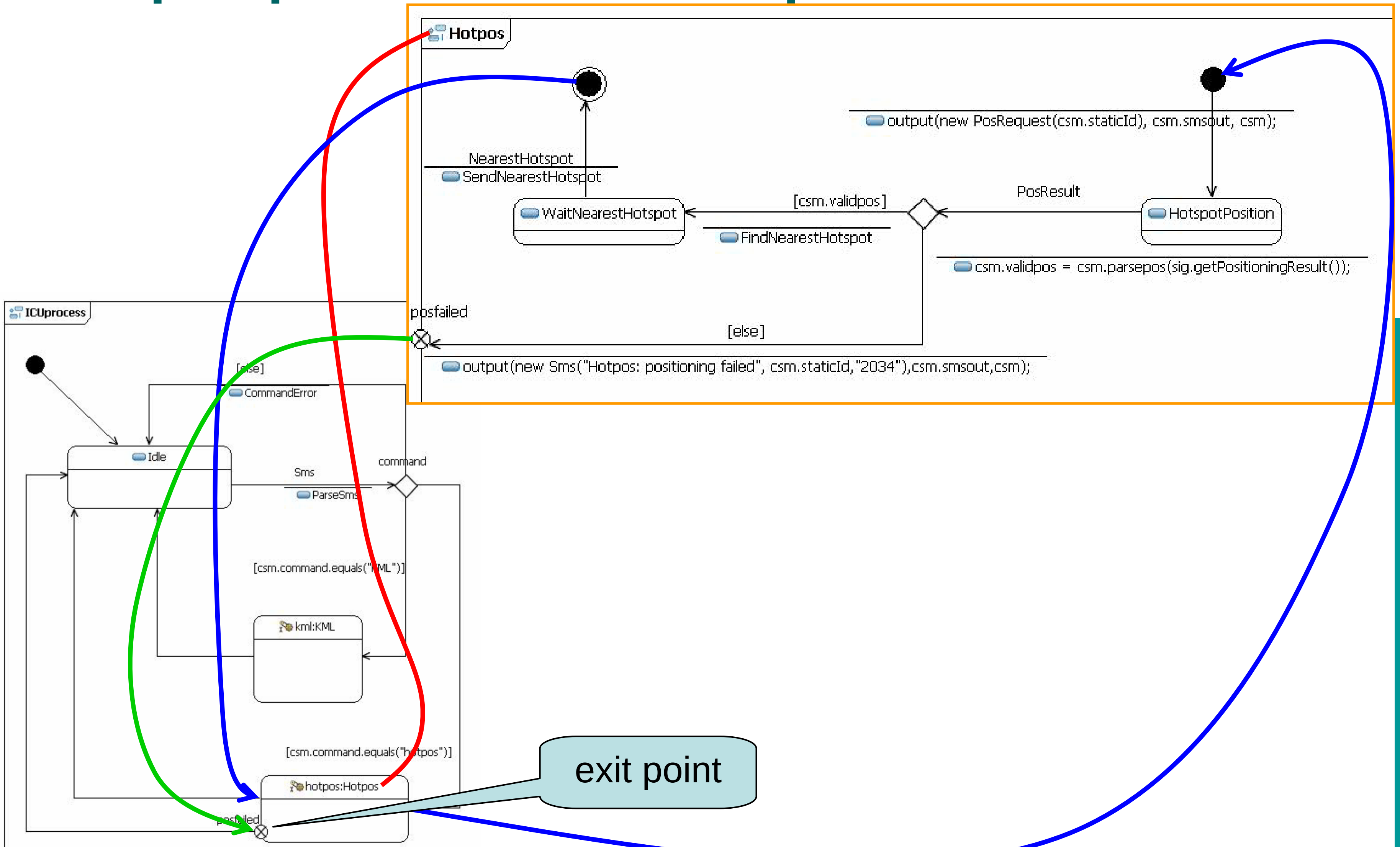
Submachine states

- Submachine states are states
- Submachine states have a state machine definition
  - but at the level of the submachine state, they are perceived only as states
- Submachine states are compiled into JavaFrame composite states
  - which must not be confused with composite structures!!!
  - UML also has something called ”composite states” but they are not as powerful as submachine states. The JavaFrame compiler does not recognize UML composite states.

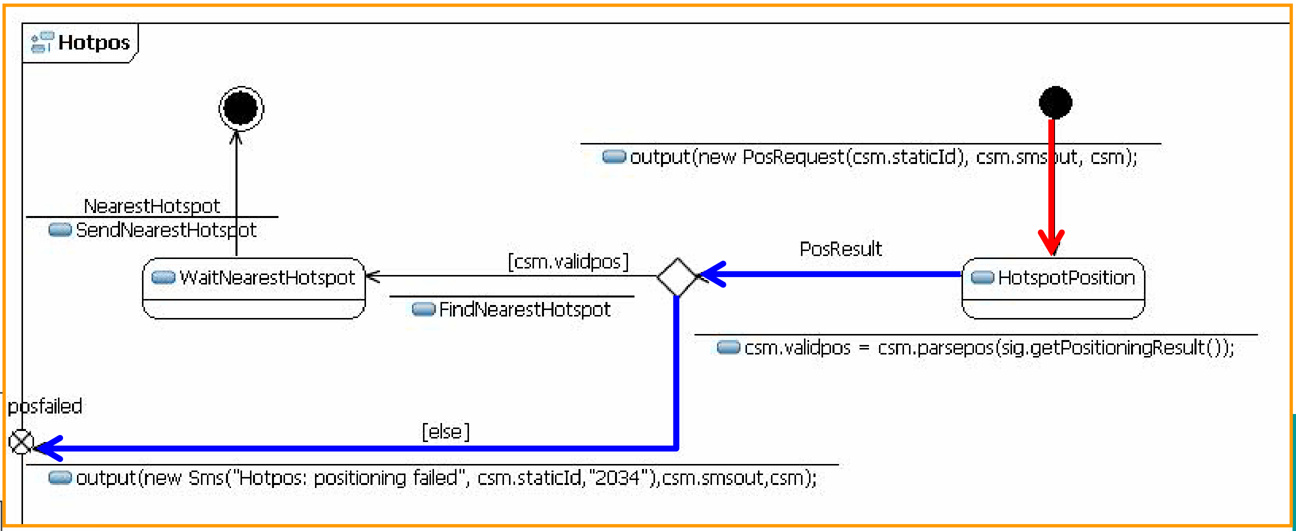
KML process inside ICUprocess

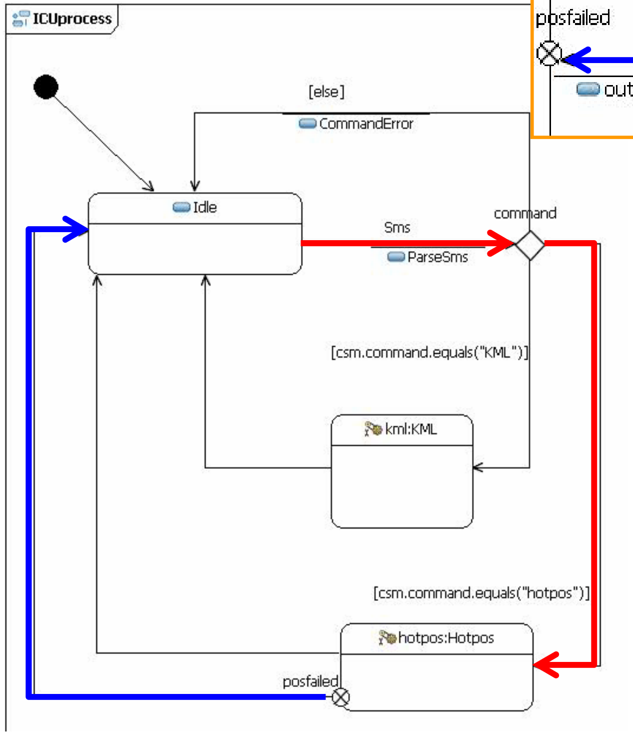


Hotpos process inside ICUprocess

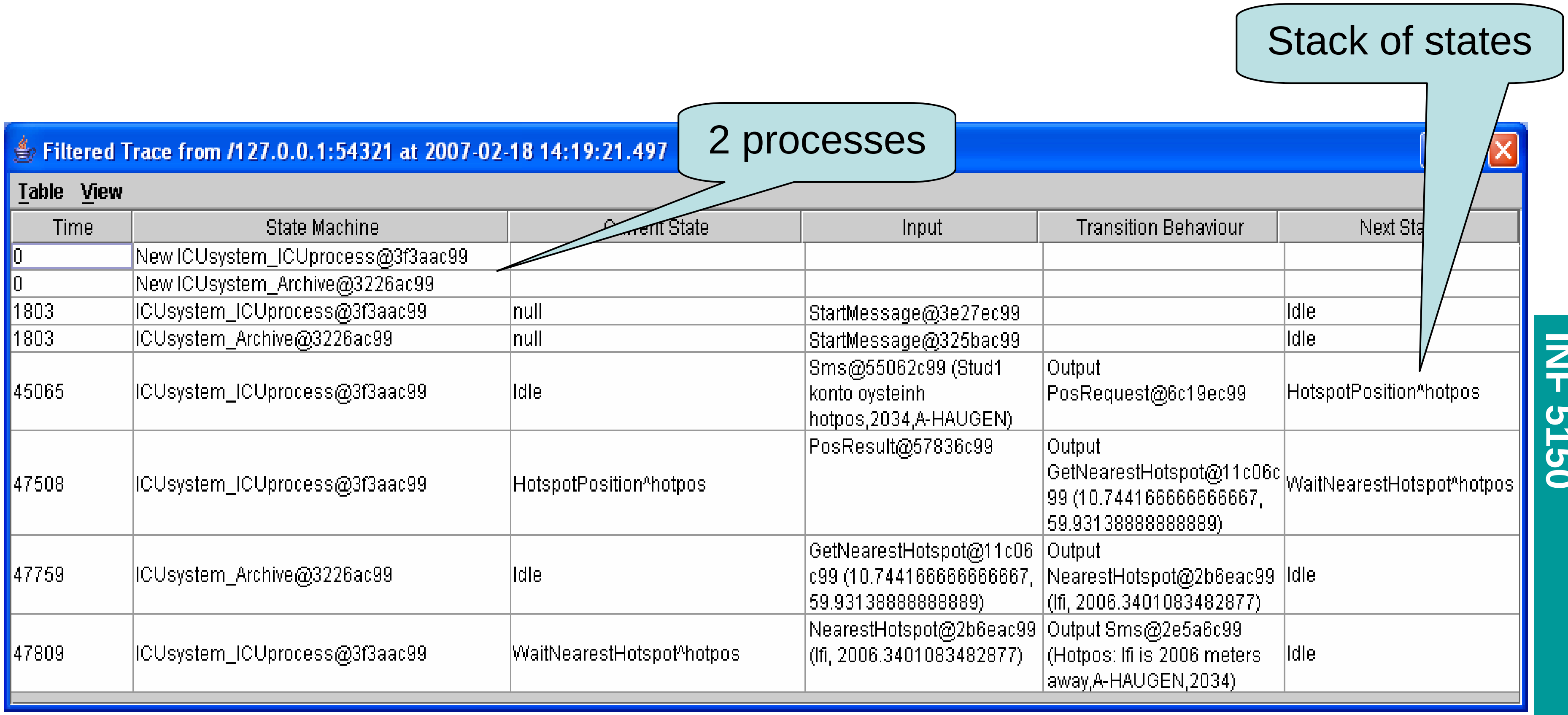

Hotpos
NearestHotspot
SendNearestHotspot
output(new PosRequest(csm.staticId), csm.smsout, csm);
WaitNearestHotspot
[csm.validpos]
FindNearestHotspot
PosResult
HotspotPosition
csm.validpos = csm.parsepos(sig.getPositioningResult());
posfailed
[else]
output(new Sms("Hotpos: positioning failed", csm.staticId,"2034"),csm.smsout,csm);
ICUprocess
[else]
CommandError
Idle
Sms
ParseSms
command
[csm.command.equals("KML")]
kml:KML
[csm.command.equals("hotpos")]
hotpos:Hotpos
posfailed
exit point

Two assembled transitions


Hotpos
output(new PosRequest(csm.staticId), csm.smsout, csm);
NearestHotspot
SendNearestHotspot
WaitNearestHotspot
[csm.validpos]
FindNearestHotspot
PosResult
HotspotPosition
csm.validpos = csm.parsepos(sig.getPositioningResult());
posfailed
[else]
output(new Sms("Hotpos: positioning failed", csm.staticId,"2034"),csm.smsout,csm);

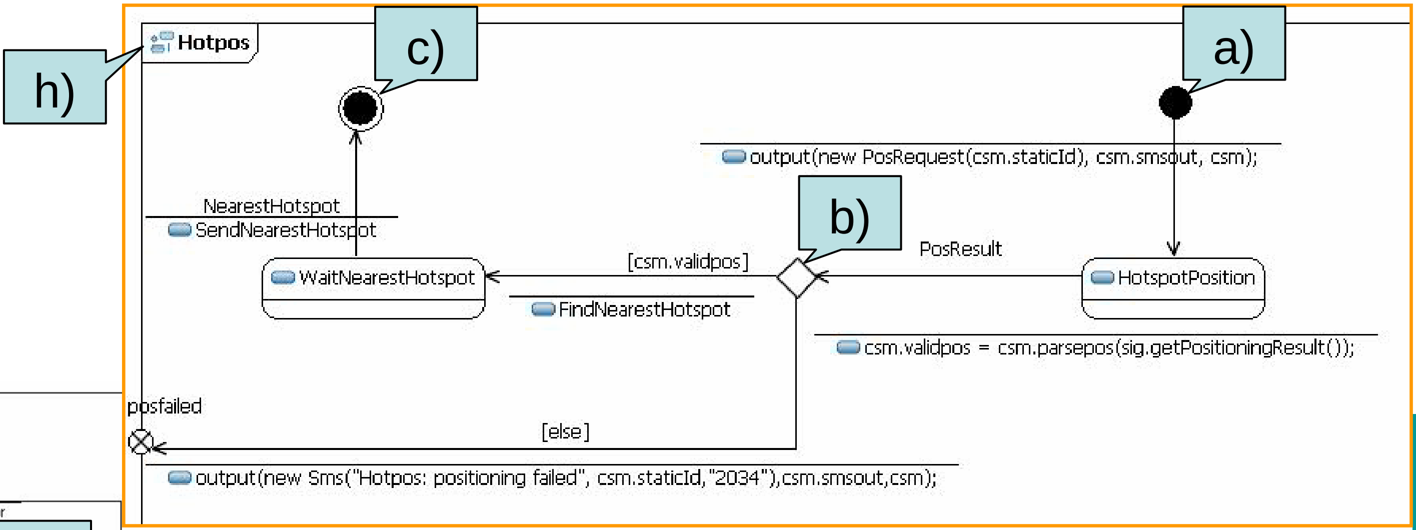


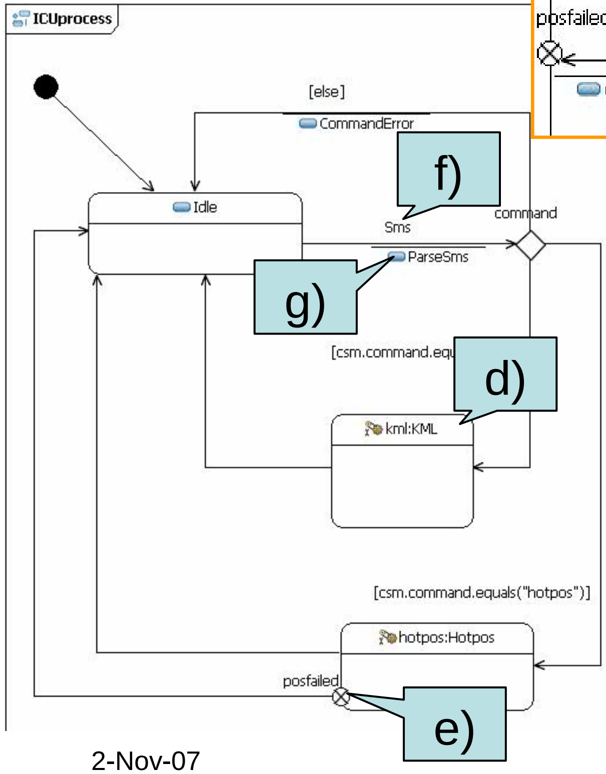
Execution as seen from JFTrace


Filtered Trace from /127.0.0.1:54321 at 2007-02-18 14:19:21.497
Table View
2 processes
Stack of states

Time	State Machine	C[illegible]ent State	Input	Transition Behaviour	Next Sta[illegible]
0	New ICUsystem_ICUprocess@3f3aac99				
0	New ICUsystem_Archive@3226ac99				
1803	ICUsystem_ICUprocess@3f3aac99	null	StartMessage@3e27ec99		Idle
1803	ICUsystem_Archive@3226ac99	null	StartMessage@325bac99		Idle
45065	ICUsystem_ICUprocess@3f3aac99	Idle	Sms@55062c99 (Stud1 konto oysteinh hotpos,2034,A-HAUGEN)	Output PosRequest@6c19ec99	HotspotPosition^hotpos
47508	ICUsystem_ICUprocess@3f3aac99	HotspotPosition^hotpos	PosResult@57836c99	Output GetNearestHotspot@11c06c99 (10.744166666666667, 59.931388888888889)	WaitNearestHotspot^hotpos
47759	ICUsystem_Archive@3226ac99	Idle	GetNearestHotspot@11c06c99 (10.744166666666667, 59.931388888888889)	Output NearestHotspot@2b6eac99 (Ifi, 2006.3401083482877)	Idle
47809	ICUsystem_ICUprocess@3f3aac99	WaitNearestHotspot^hotpos	NearestHotspot@2b6eac99 (Ifi, 2006.3401083482877)	Output Sms@2e5a6c99 (Hotpos: Ifi is 2006 meters away,A-HAUGEN,2034)	Idle

Write down the names of these elements


h)
Hotpos
c)
a)
output(new PosRequest(csm.staticId), csm.smsout, csm);
NearestHotspot
SendNearestHotspot
b)
PosResult
[csm.validpos]
WaitNearestHotspot
HotspotPosition
FindNearestHotspot
csm.validpos = csm.parsepos(sig.getPositioningResult());
posfailed
[else]
output(new Sms("Hotpos: positioning failed", csm.staticId,"2034"),csm.smsout,csm);



a) Initial state (pseudo state)

b) Choice (pseudo state)

c) Final state

d) Submachine state

e) Exit-point

f) Trigger

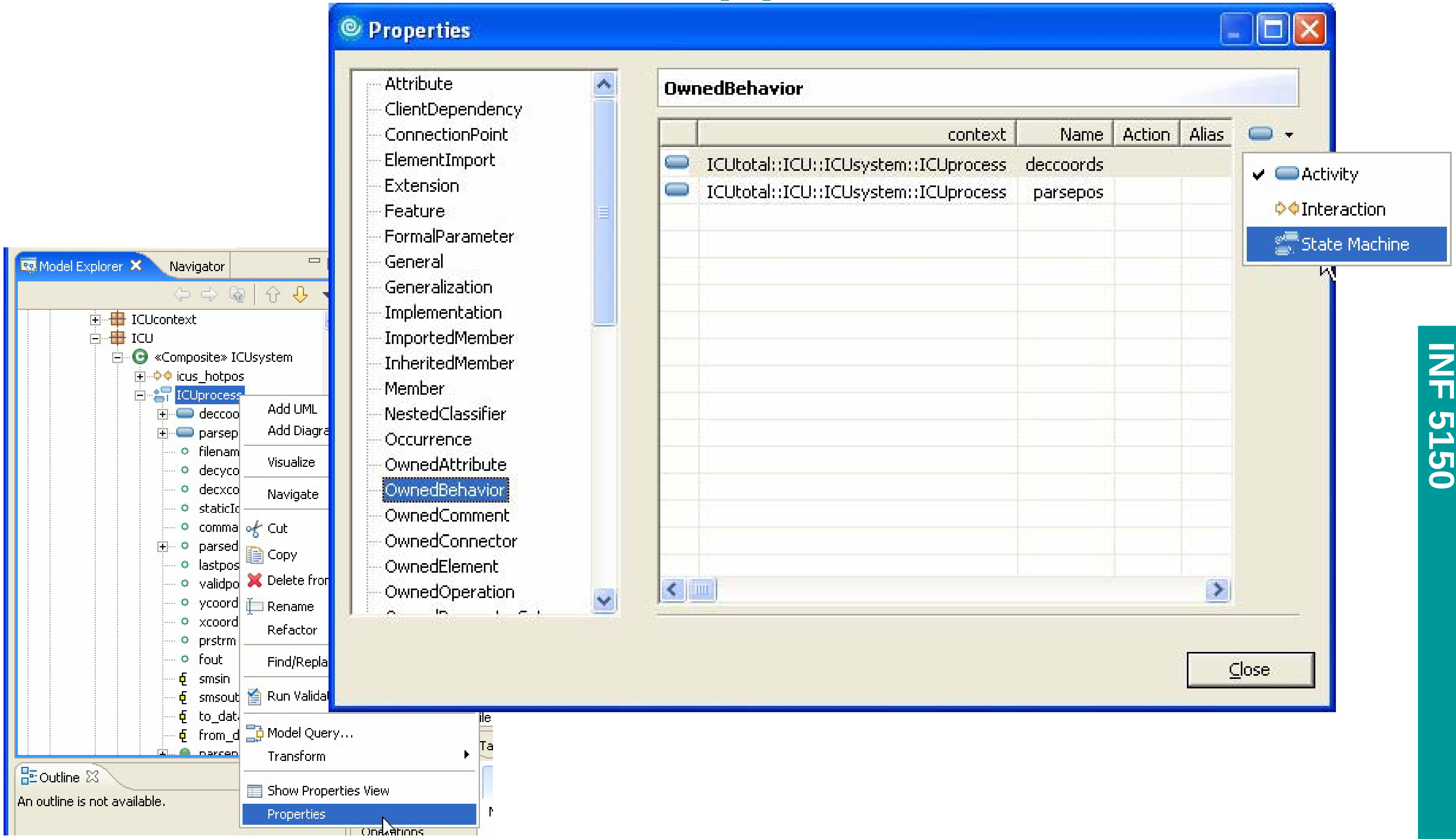
g) Effect

h) State Machine

How to change ICU3 into ICU4 with RSM

The technicalities of changing the old model into a new one using submachine states

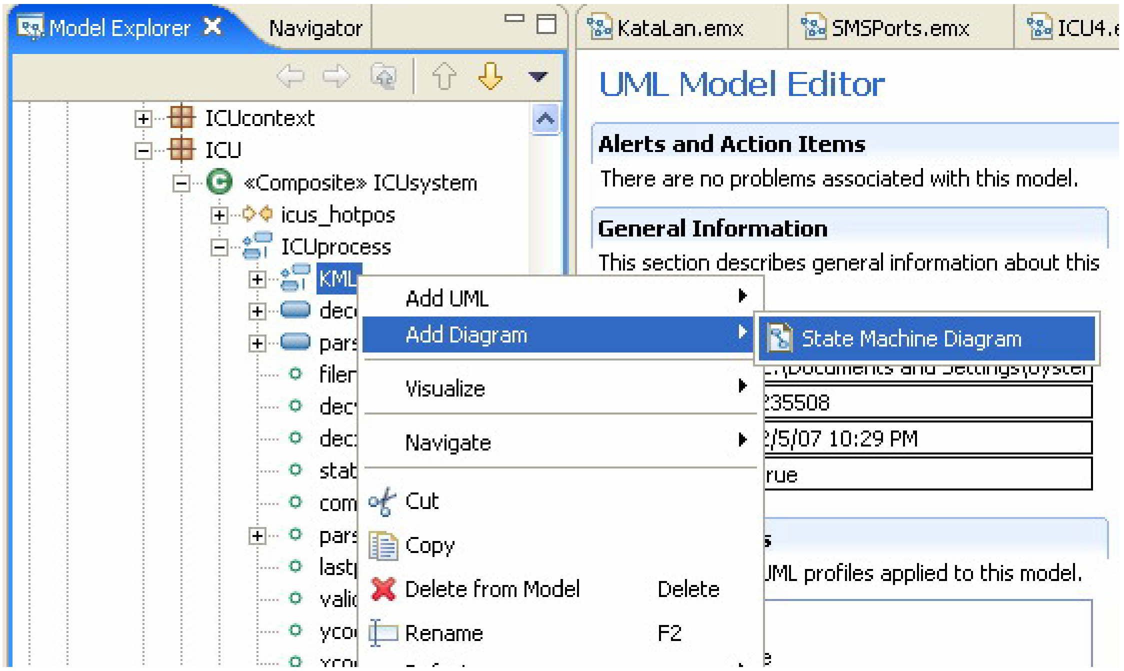
Add *local* state machine(s)



Properties

	context	Name	Action	Alias
	ICUtotal::ICU::ICUsystem::ICUprocess	deccoords		
	ICUtotal::ICU::ICUsystem::ICUprocess	parsepos		

And then adding a diagram to the state machine



UML Model Editor

Alerts and Action Items

There are no problems associated with this model.

General Information

This section describes general information about this

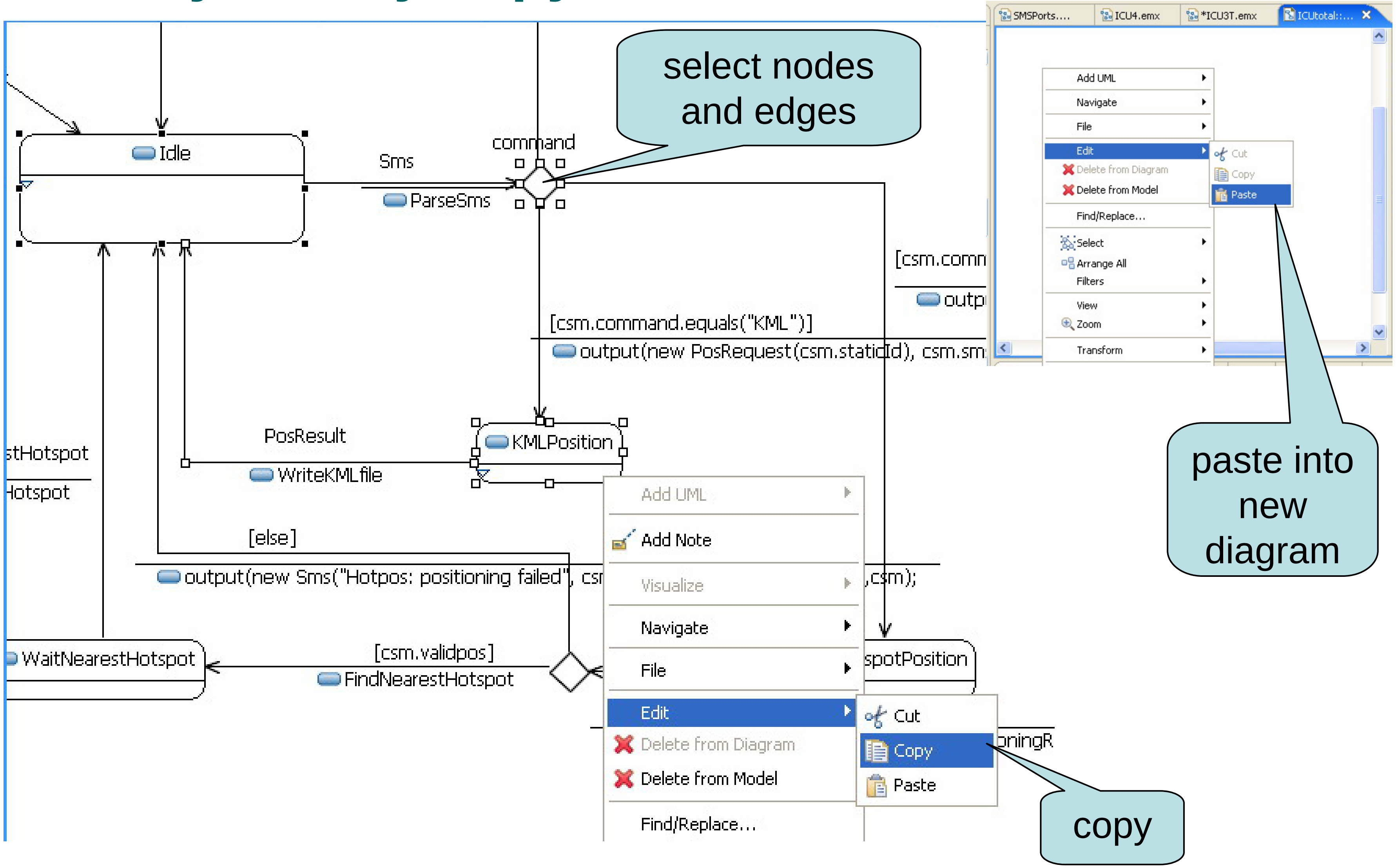
235508

2/5/07 10:29 PM

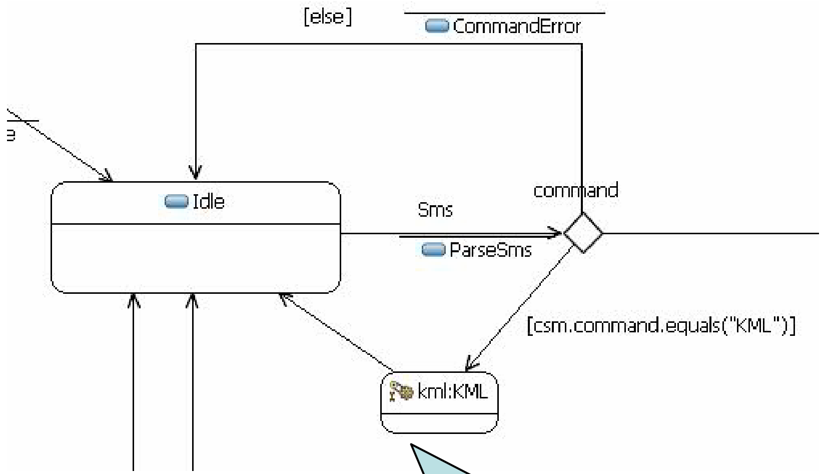
rue

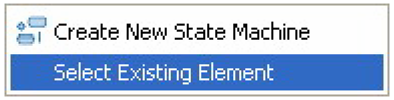
UML profiles applied to this model.

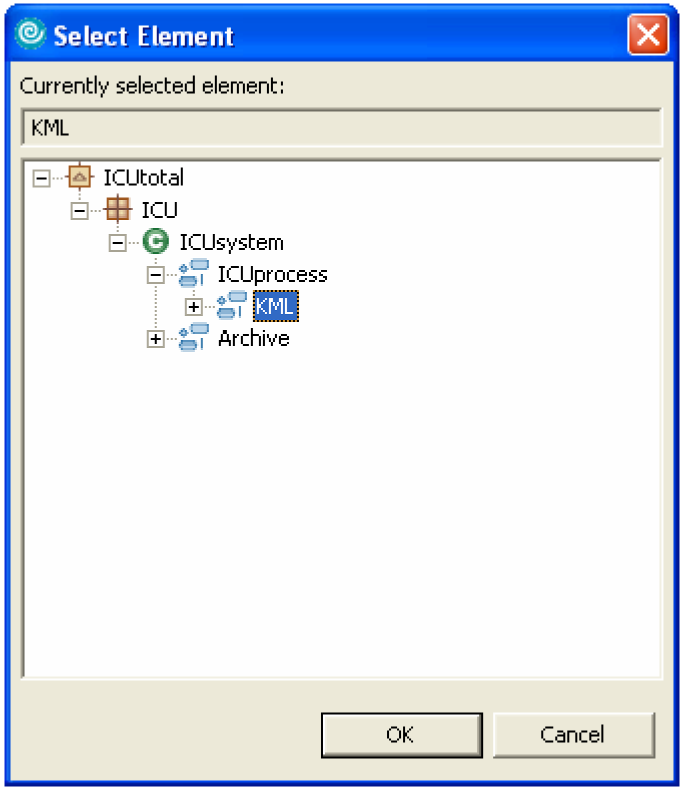
Then you may copy in state machine elements

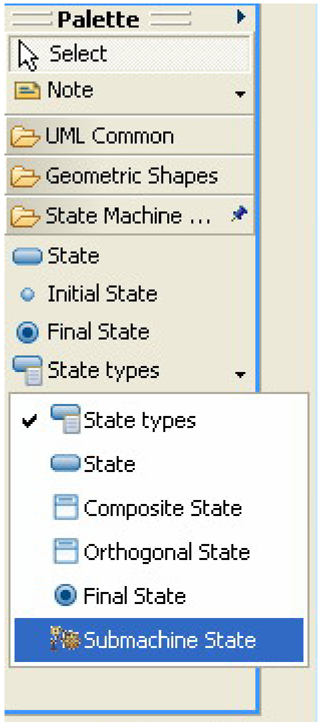


Then clean up (here ICUprocess)









Create submachinestate
Attach transitions
Remove copied material

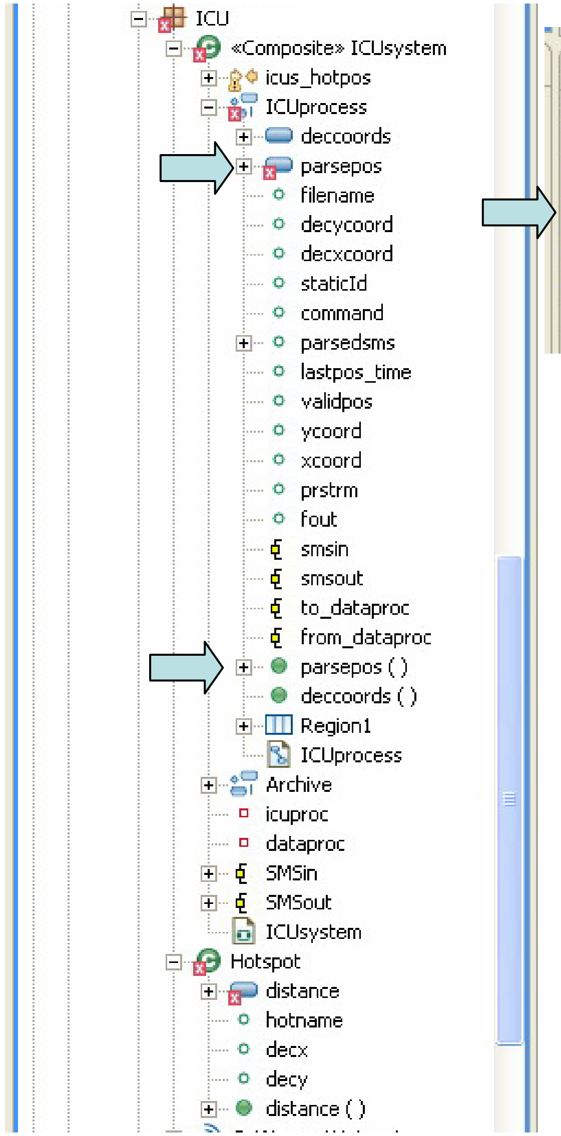
A few points on model validation

about errors and warnings

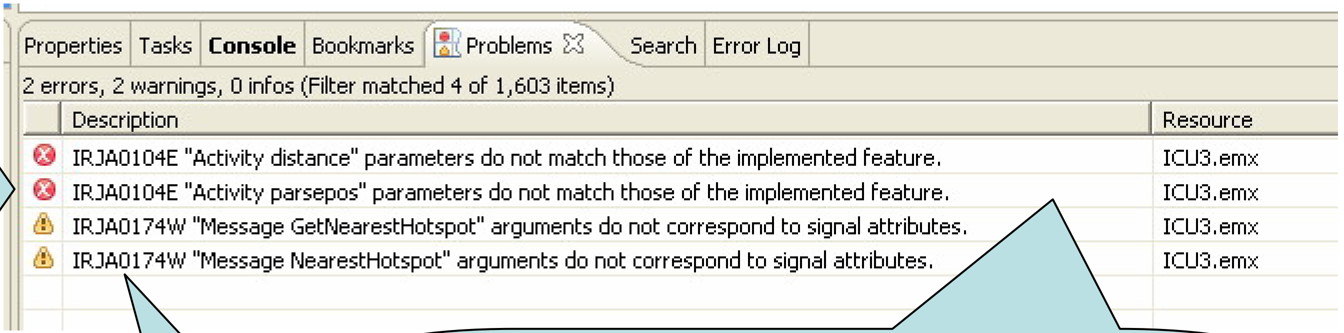
The UML models should be validated

- A validated UML model means that
  - The model is syntactically correct
  - The model satisfies a number of static requirements
- The RSM validation does not include dynamic validation
  - which would have discovered properties of the running model
- Our JavaFrame profile works on a subset of UML
  - which means that some requirements are not significant
  - they should still preferably be correct even though they would not matter for the execution
    - but later versions may take them into account

Problems with ICU3



ICU
«Composite» ICUsystem
icus_hotpos
ICUprocess
deccoords
parsepos
filename
decycoord
decxcoord
staticId
command
parsedsms
lastpos_time
validpos
ycoord
xcoord
prstrm
fout
smsin
smsout
to_dataproc
from_dataproc
parsepos ( )
deccoords ( )
Region1
ICUprocess
Archive
icuproc
dataproc
SMSin
SMSout
ICUsystem
Hotspot
distance
hotname
decx
decy
distance ( )



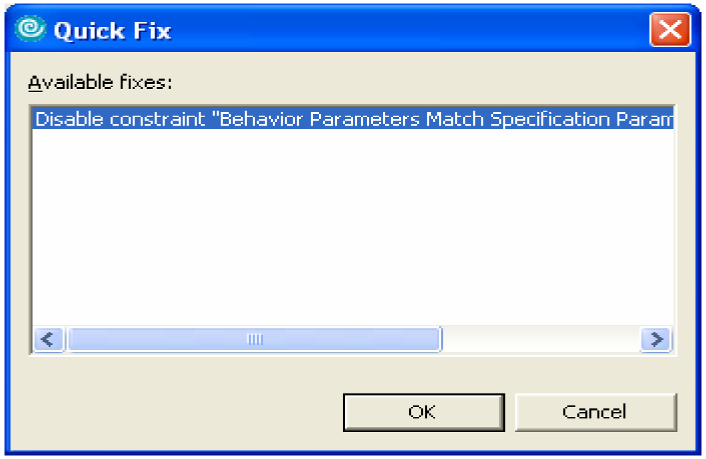
2 errors, 2 warnings, 0 infos (Filter matched 4 of 1,603 items)

Description	Resource
IRJA0104E "Activity distance" parameters do not match those of the implemented feature.	ICU3.emx
IRJA0104E "Activity parsepos" parameters do not match those of the implemented feature.	ICU3.emx
IRJA0174W "Message GetNearestHotspot" arguments do not correspond to signal attributes.	ICU3.emx
IRJA0174W "Message NearestHotspot" arguments do not correspond to signal attributes.	ICU3.emx

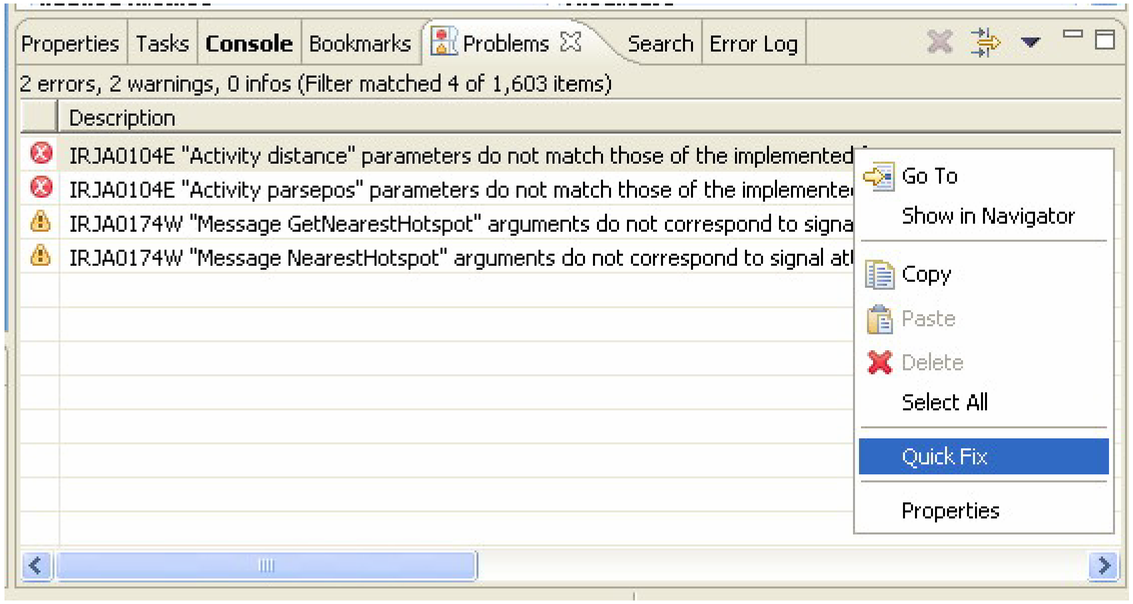
Mismatch between parameters, but the implementation still works

Messages in SeDi are not coded exactly as UML2 specifies regarding parameters. This is on purpose.

Fixing these problems by ignoring them



Disable constraint "Behavior Parameters Match Specification Param



2 errors, 2 warnings, 0 infos (Filter matched 4 of 1,603 items)

Description
IRJA0104E "Activity distance" parameters do not match those of the implemented
IRJA0104E "Activity parsepos" parameters do not match those of the implemente
IRJA0174W "Message GetNearestHotspot" arguments do not correspond to signa
IRJA0174W "Message NearestHotspot" arguments do not correspond to signal at

Sessions – Multiple concurrent users

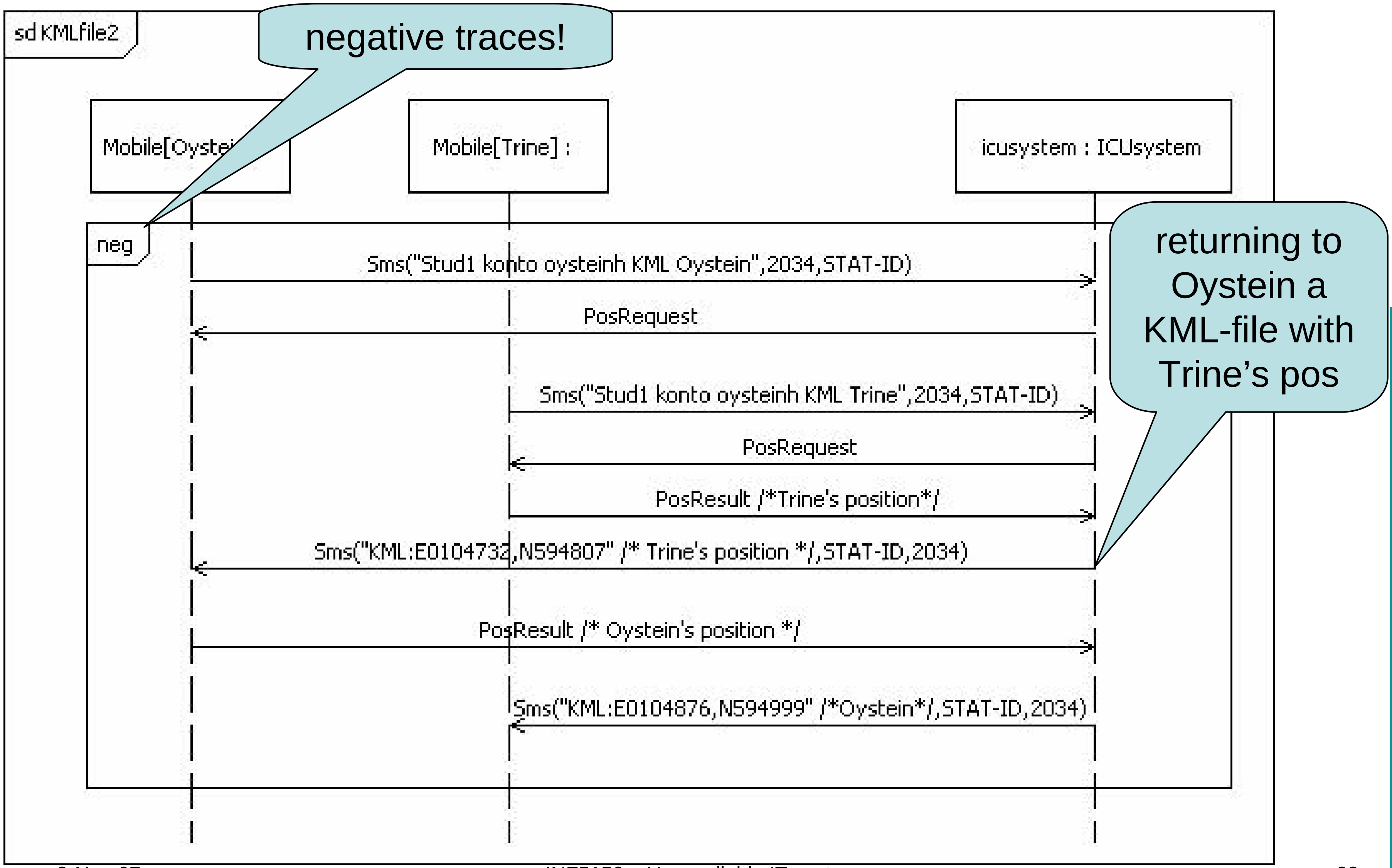
Motivation

- Assume having several users using ICU concurrently
  - The system could try and handle one user at the time
  - The system could try and handle everybody at the same time, but keep their data apart
- Some things take real time outside the ICU system
  - Users thinking
  - Positioning
  - SMS forwarding
- Potentially
  - Handling all users ”at the same time” may gain overall throughput

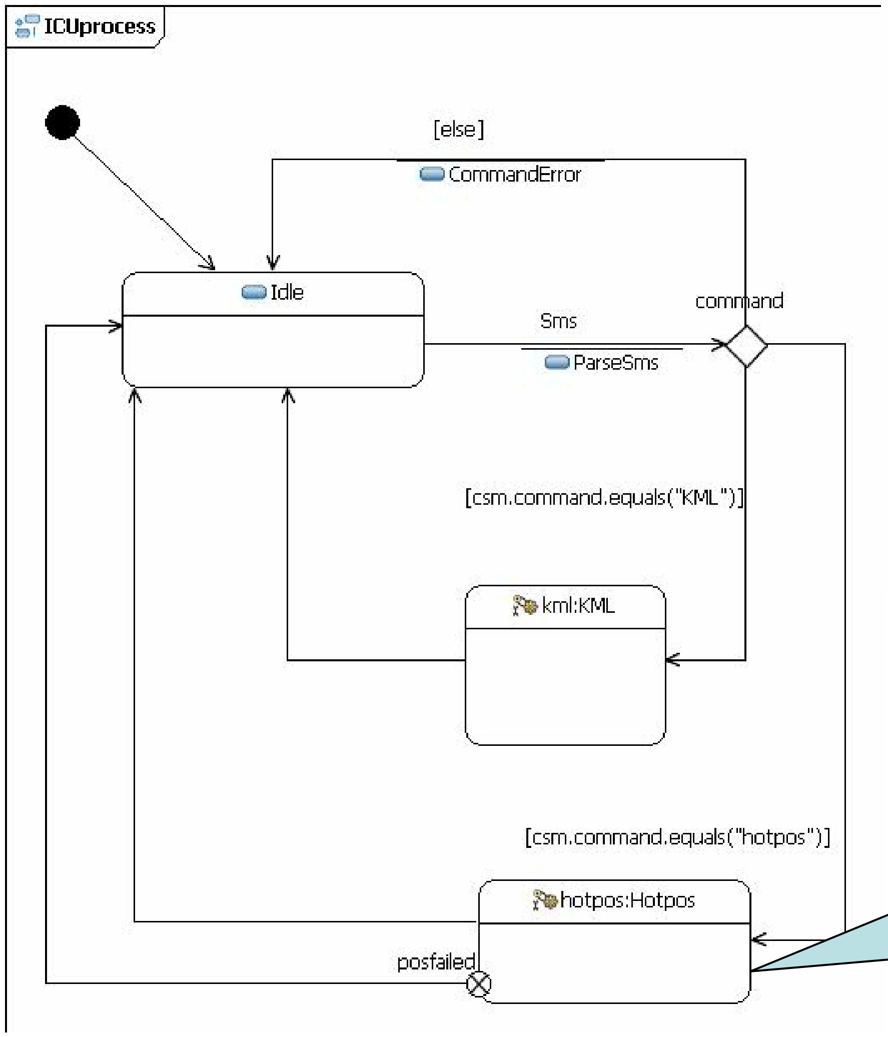
Risks

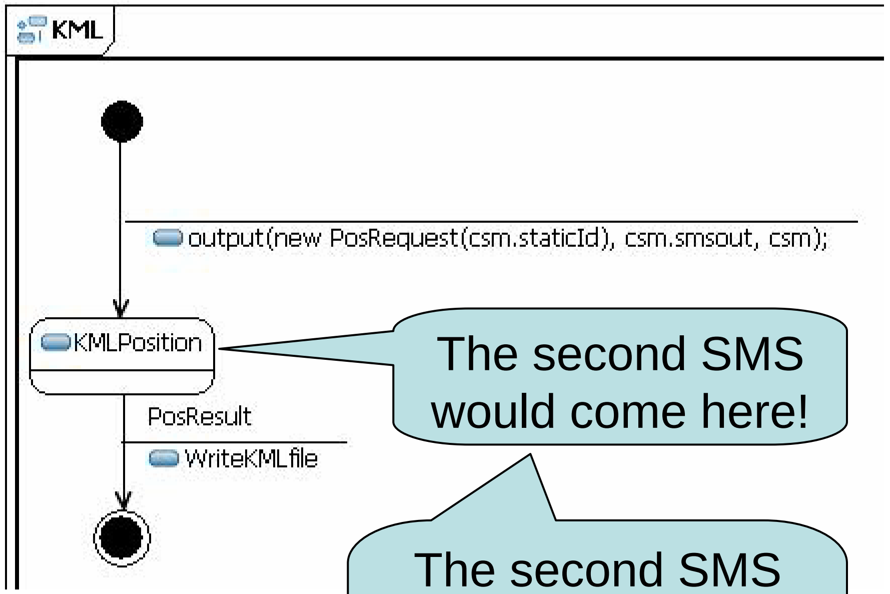
- The ICU system confuses which user has which position
- The ICU system returns SMS'es to the wrong user
- Coordinates are garbled
  - x-coordinate from one user and y-coordinate from another

This should not happen ....


sd KMLfile2
negative traces!
Mobile[Oystei
Mobile[Trine] :
icusystem : ICUsystem
neg
Sms("Stud1 konto oysteinh KML Oystein",2034,STAT-ID)
PosRequest
Sms("Stud1 konto oysteinh KML Trine",2034,STAT-ID)
PosRequest
PosResult /*Trine's position*/
Sms("KML:E0104732,N594807" /* Trine's position */,STAT-ID,2034)
PosResult /* Oystein's position */
Sms("KML:E0104876,N594999" /*Oystein*/,STAT-ID,2034)
returning to Oystein a KML-file with Trine's pos

What would our current design do?



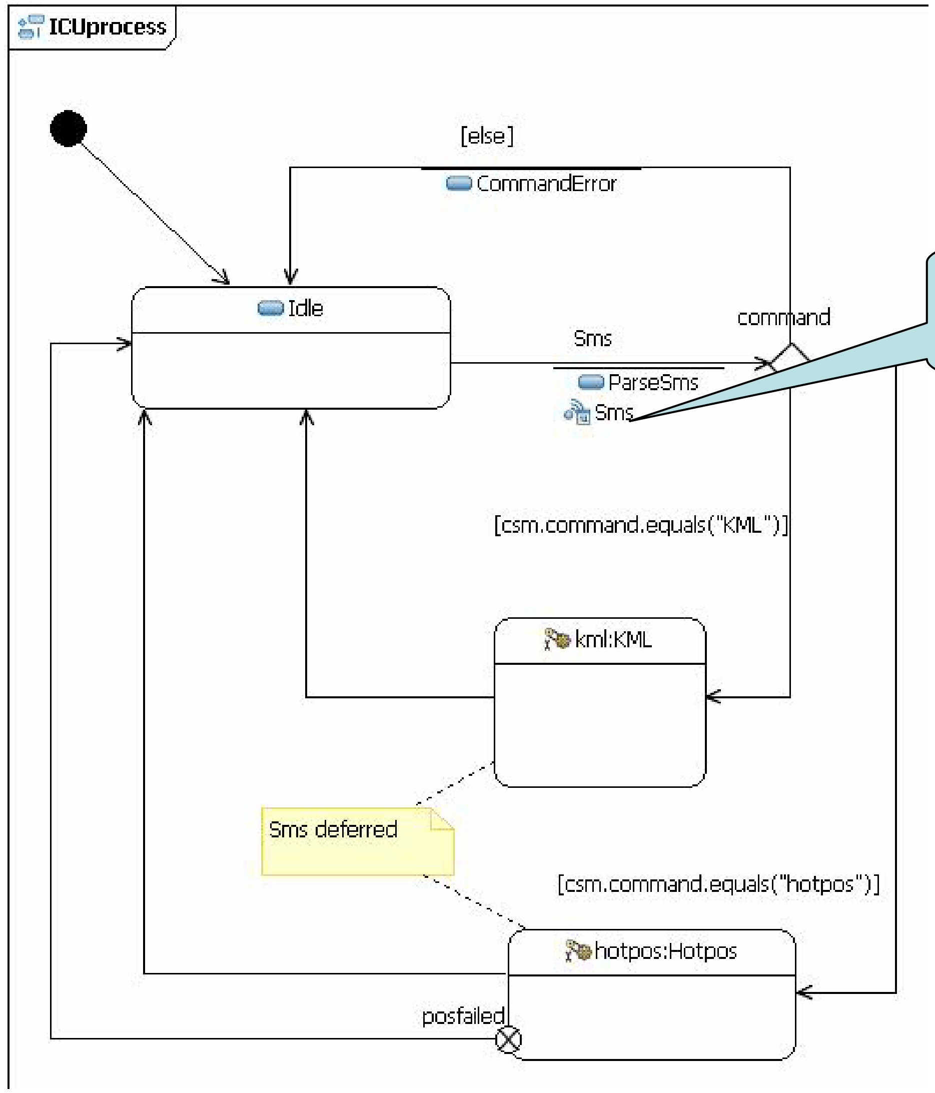


The second SMS would come here!

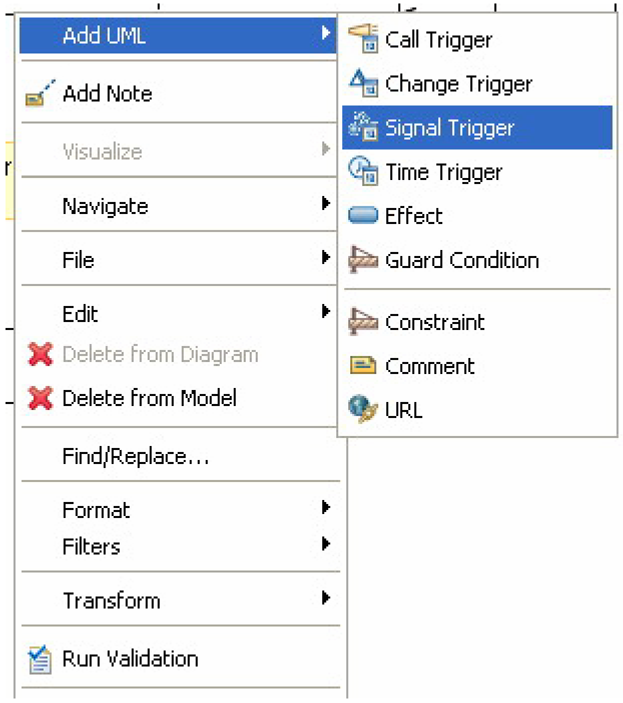
The second SMS would simply be discarded as a ”default transition”

Introducing *defer* for *Sms* on every state but Idle, will cause services to be performed in sequence

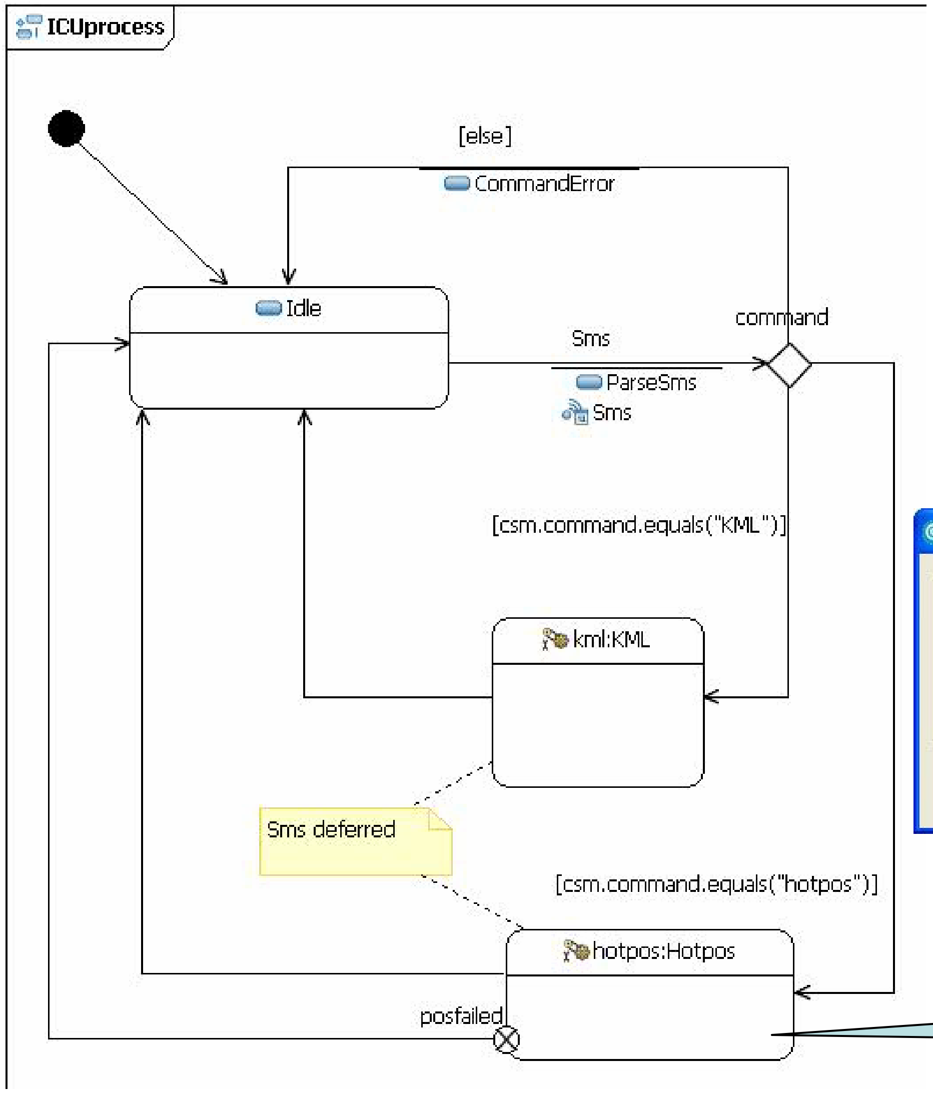
Defining Signal trigger Sms

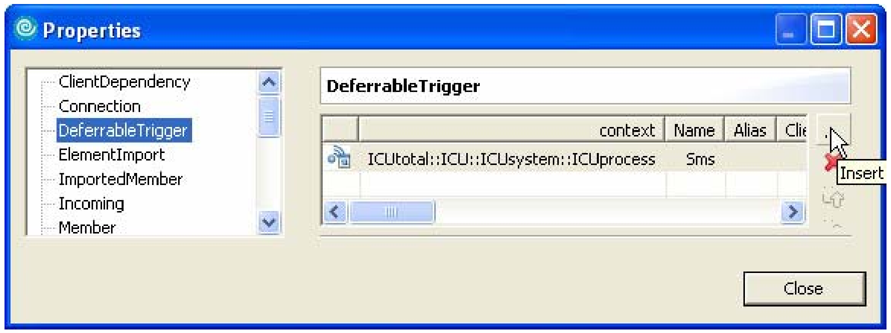


Making SMS a ”signal trigger” on a transition



Defer on the service submachine states



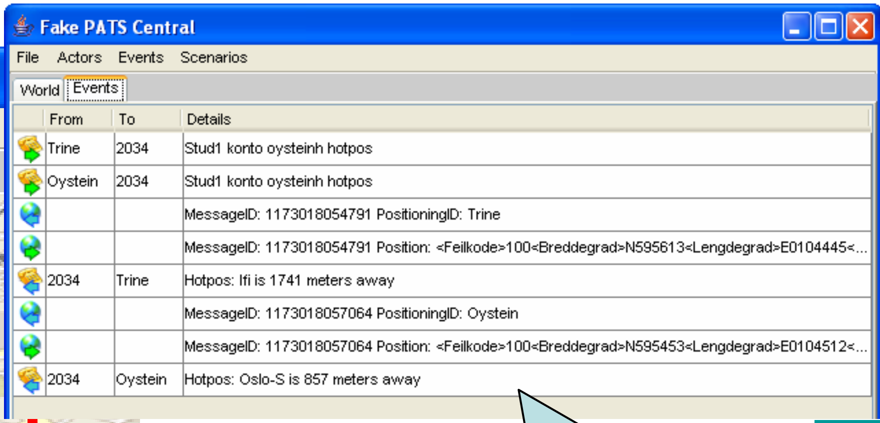


DeferrableTrigger

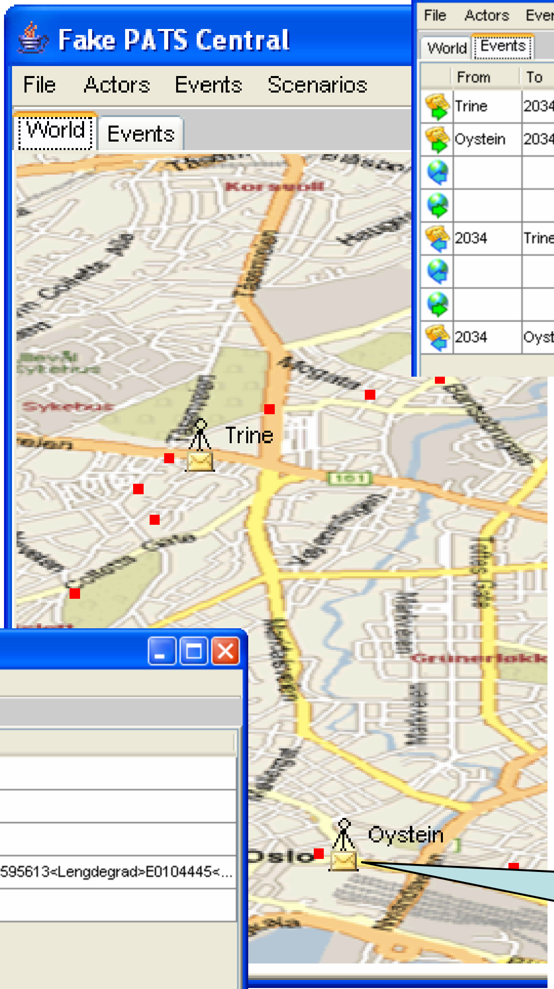
	context	Name	Alias	Clie
	ICUtotal::ICU::ICUsystem::ICUprocess	Sms		

Defining a "deferrable trigger"

Comparing ICU4 and ICU4-DEFER



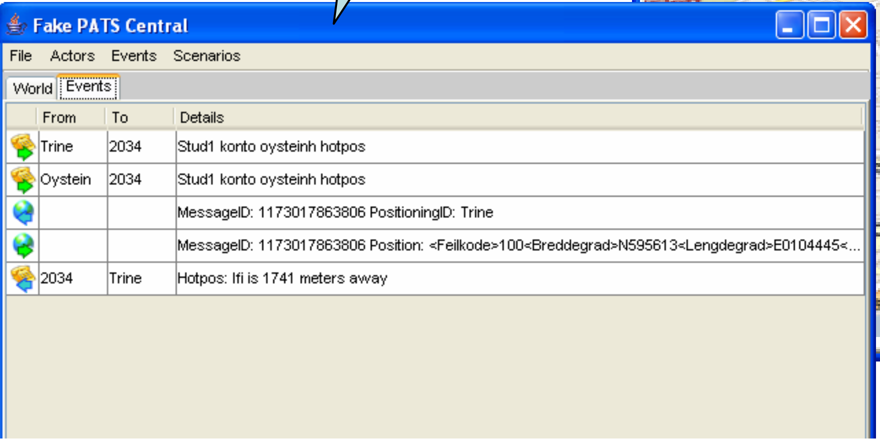
From	To	Details
Trine	2034	Stud1 konto oysteinh hotpos
Oystein	2034	Stud1 konto oysteinh hotpos
		MessageID: 1173018054791 PositioningID: Trine
		MessageID: 1173018054791 Position: <Feilkode>100<Breddegrad>N595613<Lengdegrad>E0104445<...
2034	Trine	Hotpos: Ifi is 1741 meters away
		MessageID: 1173018057064 PositioningID: Oystein
		MessageID: 1173018057064 Position: <Feilkode>100<Breddegrad>N595453<Lengdegrad>E0104512<...
2034	Oystein	Hotpos: Oslo-S is 857 meters away



ICU4 ignored the second service request

ICU4-DEFER sequences the requests

queued "Stud1 konto oysteinh hotpos"

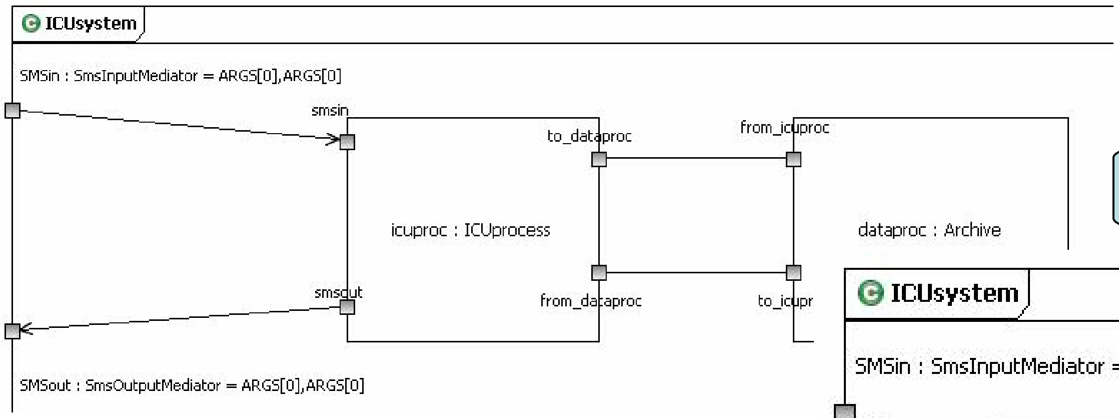


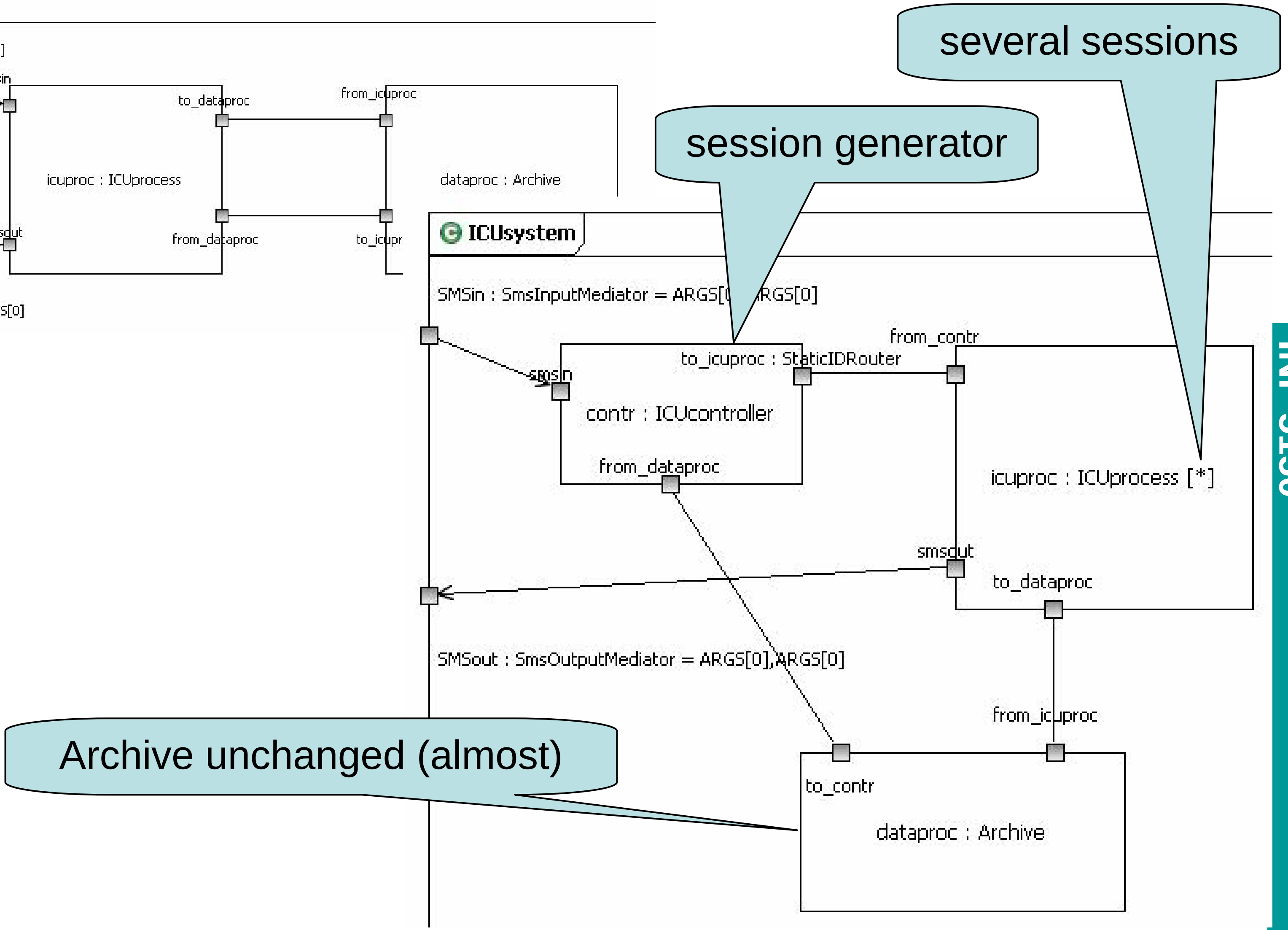
From	To	Details
Trine	2034	Stud1 konto oysteinh hotpos
Oystein	2034	Stud1 konto oysteinh hotpos
		MessageID: 1173017863806 PositioningID: Trine
		MessageID: 1173017863806 Position: <Feilkode>100<Breddegrad>N595613<Lengdegrad>E0104445<...
2034	Trine	Hotpos: Ifi is 1741 meters away

The ”session” solution

- Each initiative by a user is represented by a state machine (a session)
  - with all the temporary data associated with that user
  - taking care of all the communication related to that user
- The session is generated when the user initiates a service
- The session is terminated when the service is finished

A new composite structure

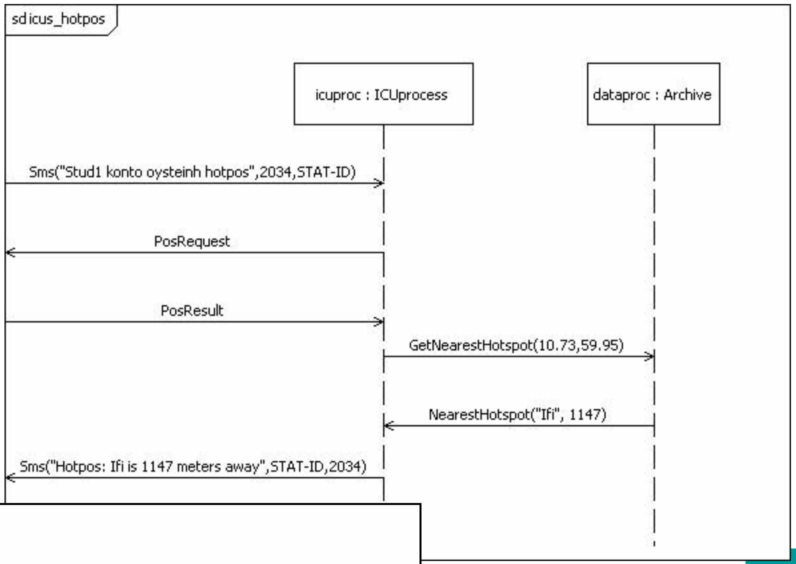


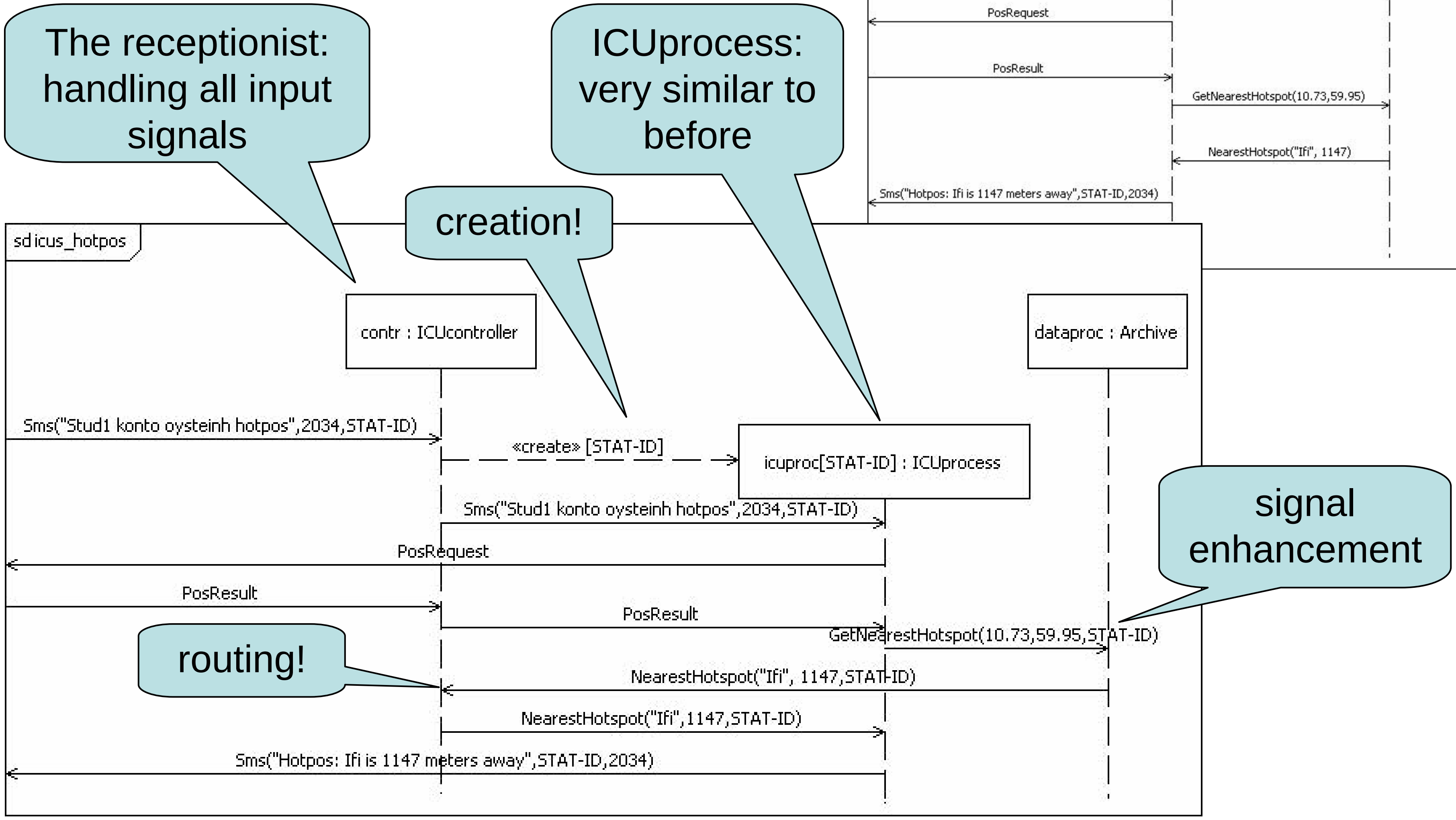


Buzzzzz Groups (5 minutes)

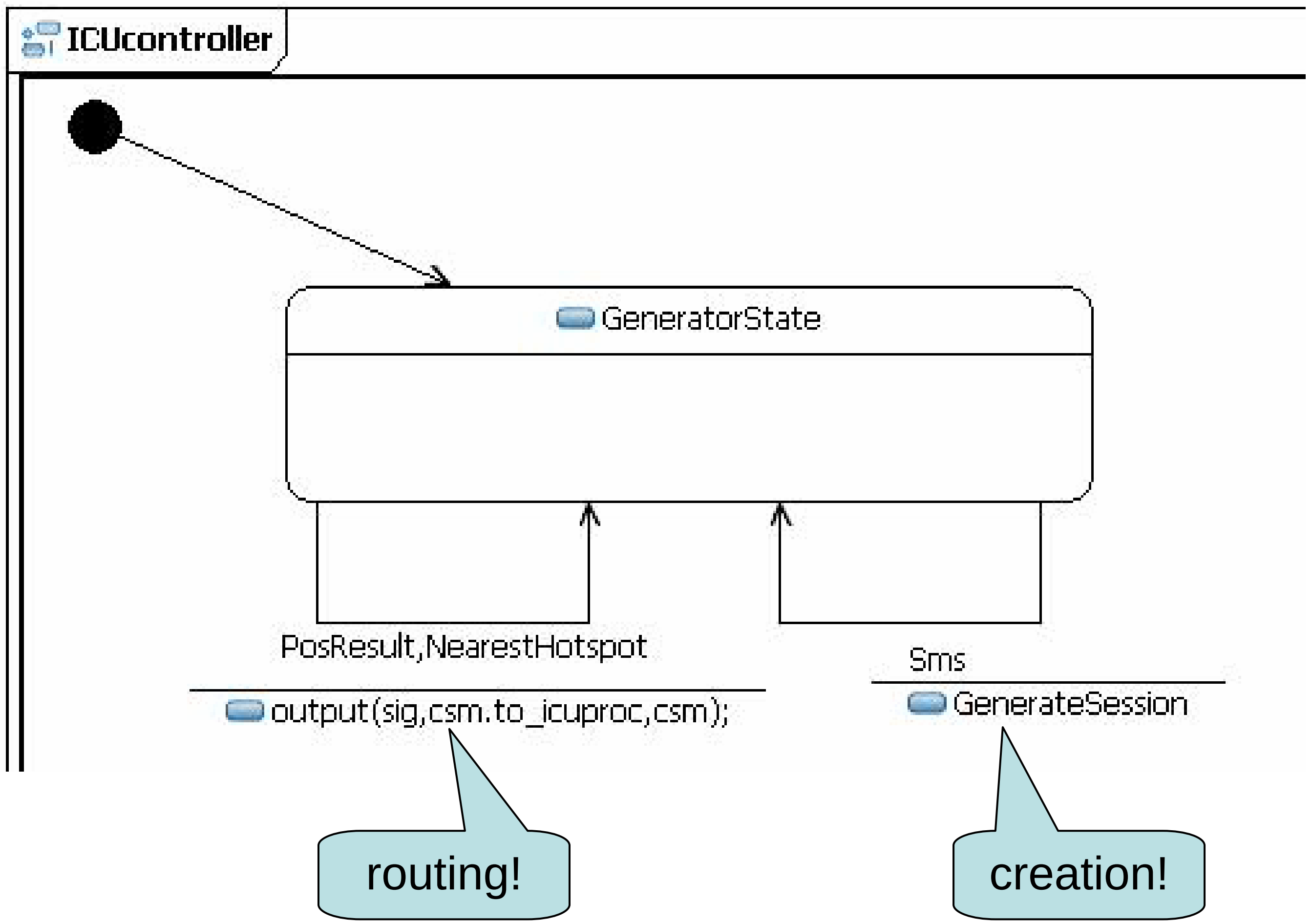
- Discuss what represents sessions in the ICU systems
- Discuss what could represent sessions in the ”Survival of the SMSest”
- Determine what should identify a session of the ICU system
- Determine what could identify a session in the Survival of the SMSest

Enhancing the behavior

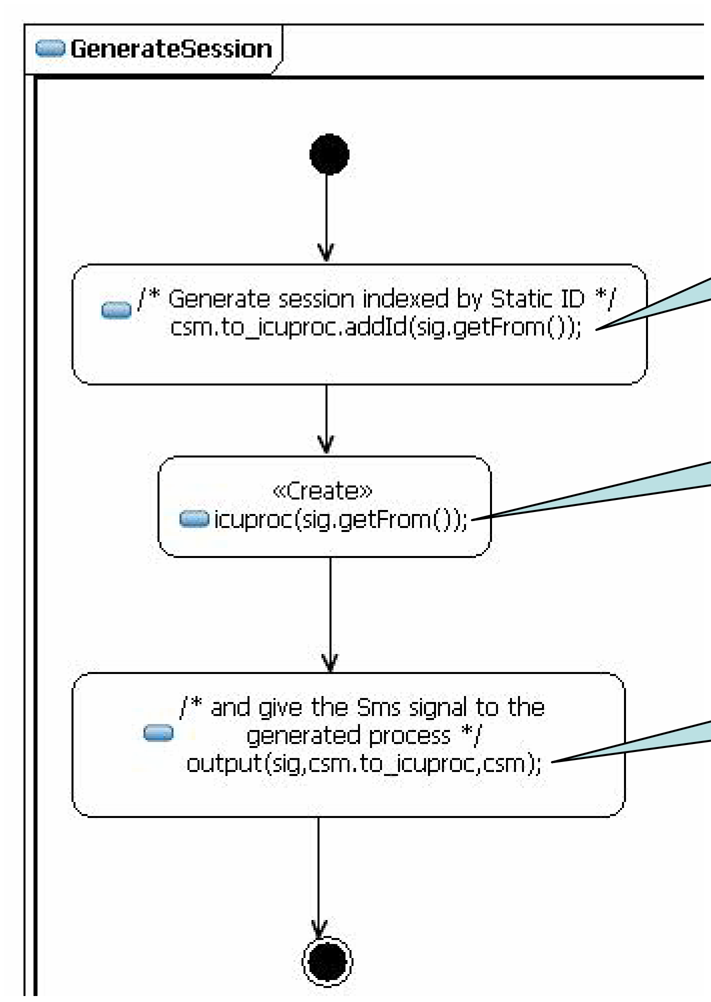

sd icus_hotpos
icuproc : ICUprocess
dataproc : Archive
Sms("Stud1 konto oysteinh hotpos",2034,STAT-ID)
PosRequest
PosResult
GetNearestHotspot(10.73,59.95)
NearestHotspot("Ifi", 1147)
Sms("Hotpos: Ifi is 1147 meters away",STAT-ID,2034)


The receptionist: handling all input signals
ICUprocess: very similar to before
creation!
signal enhancement
routing!
sd icus_hotpos
contr : ICUcontroller
dataproc : Archive
icuproc[STAT-ID] : ICUprocess
Sms("Stud1 konto oysteinh hotpos",2034,STAT-ID)
«create» [STAT-ID]
Sms("Stud1 konto oysteinh hotpos",2034,STAT-ID)
PosRequest
PosResult
PosResult
GetNearestHotspot(10.73,59.95,STAT-ID)
NearestHotspot("Ifi", 1147,STAT-ID)
NearestHotspot("Ifi",1147,STAT-ID)
Sms("Hotpos: Ifi is 1147 meters away",STAT-ID,2034)

ICUcontroller



Creating a session



```
/* Generate session indexed by Static ID */
csm.to_icuproc.addId(sig.getFrom());
```

```
«Create»
icuproc(sig.getFrom());
```

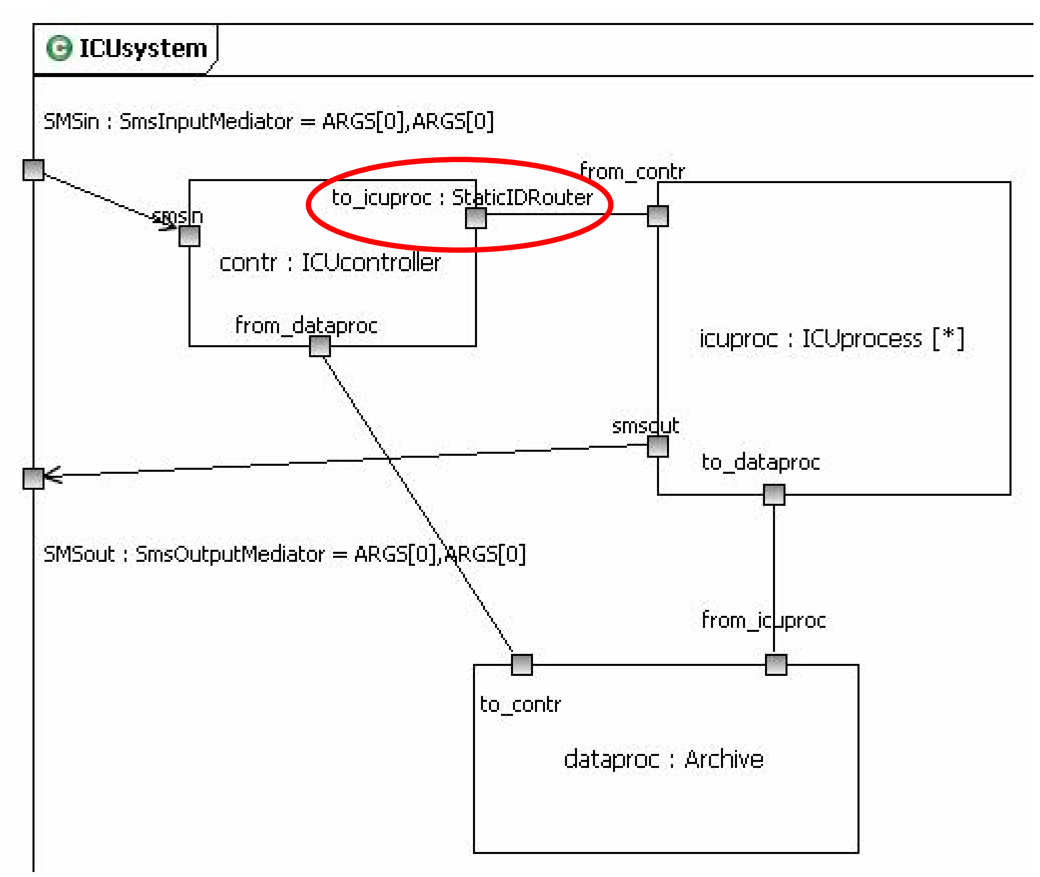
```
/* and give the Sms signal to the
generated process */
output(sig,csm.to_icuproc,csm);
```

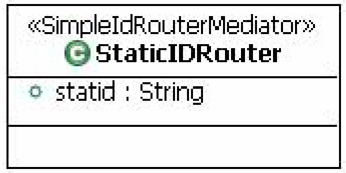
routing information so that the new member to come will ever be found

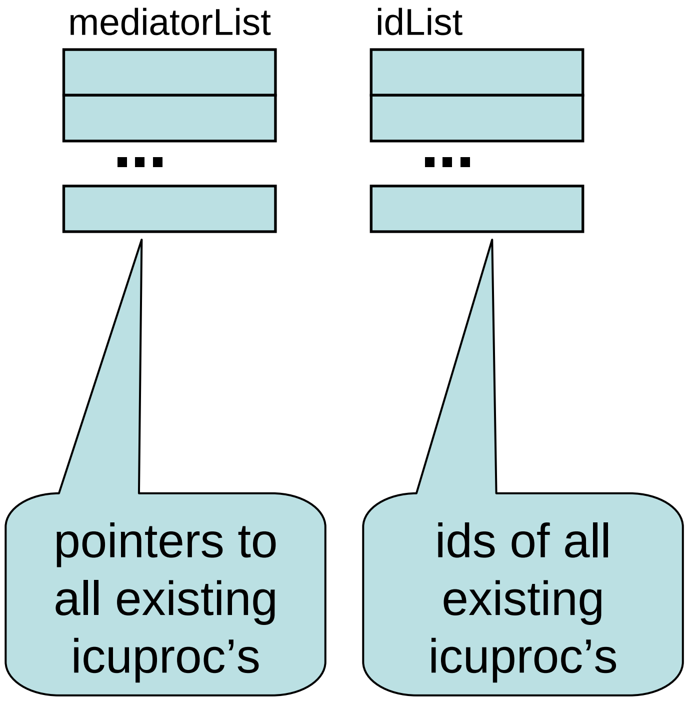
create a new member of the *icuproc* set of parts

sending a message to the generated process

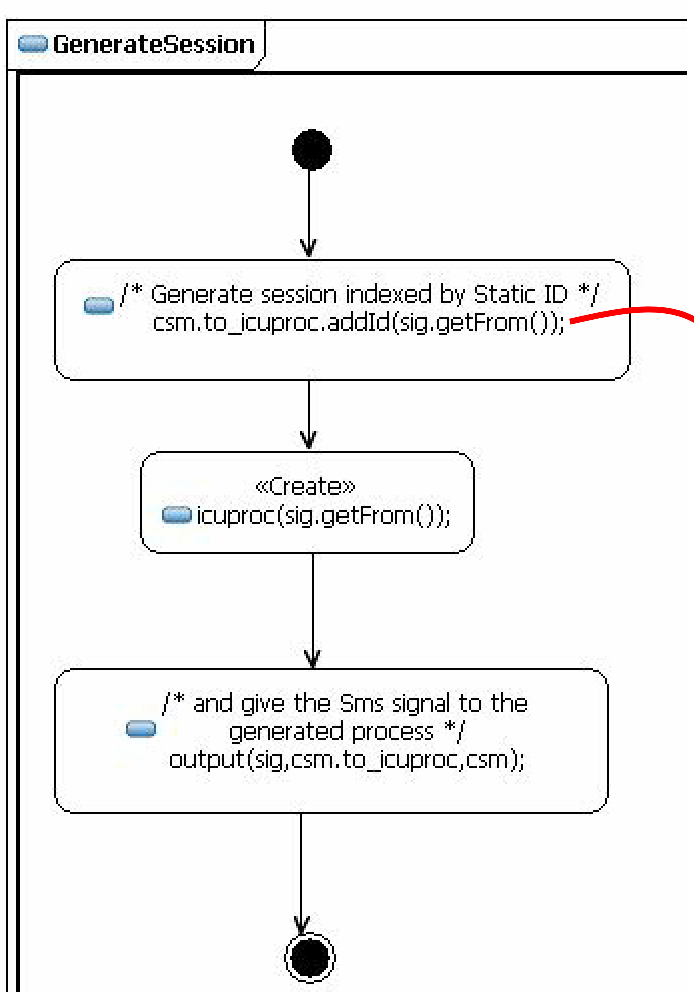
Simple Routing (1) One-to-many Port

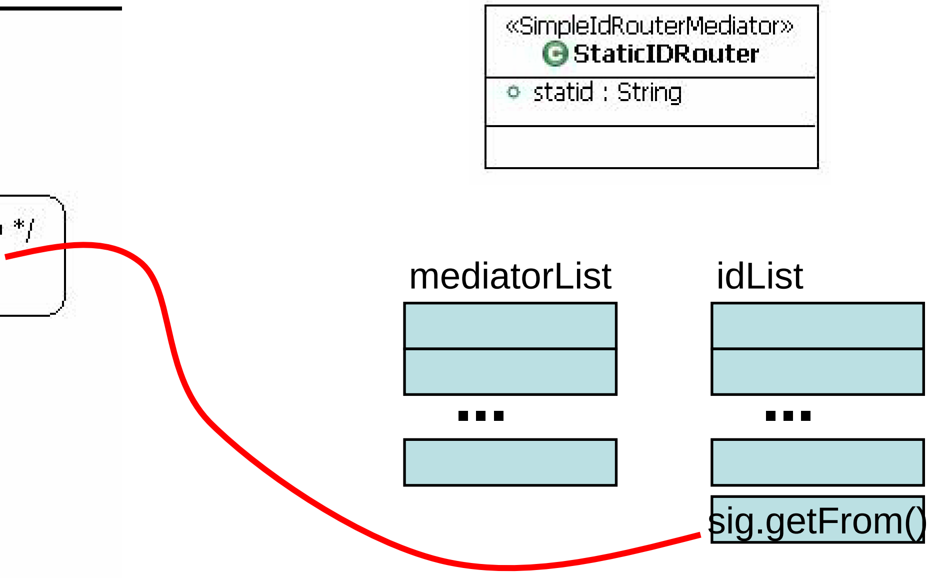




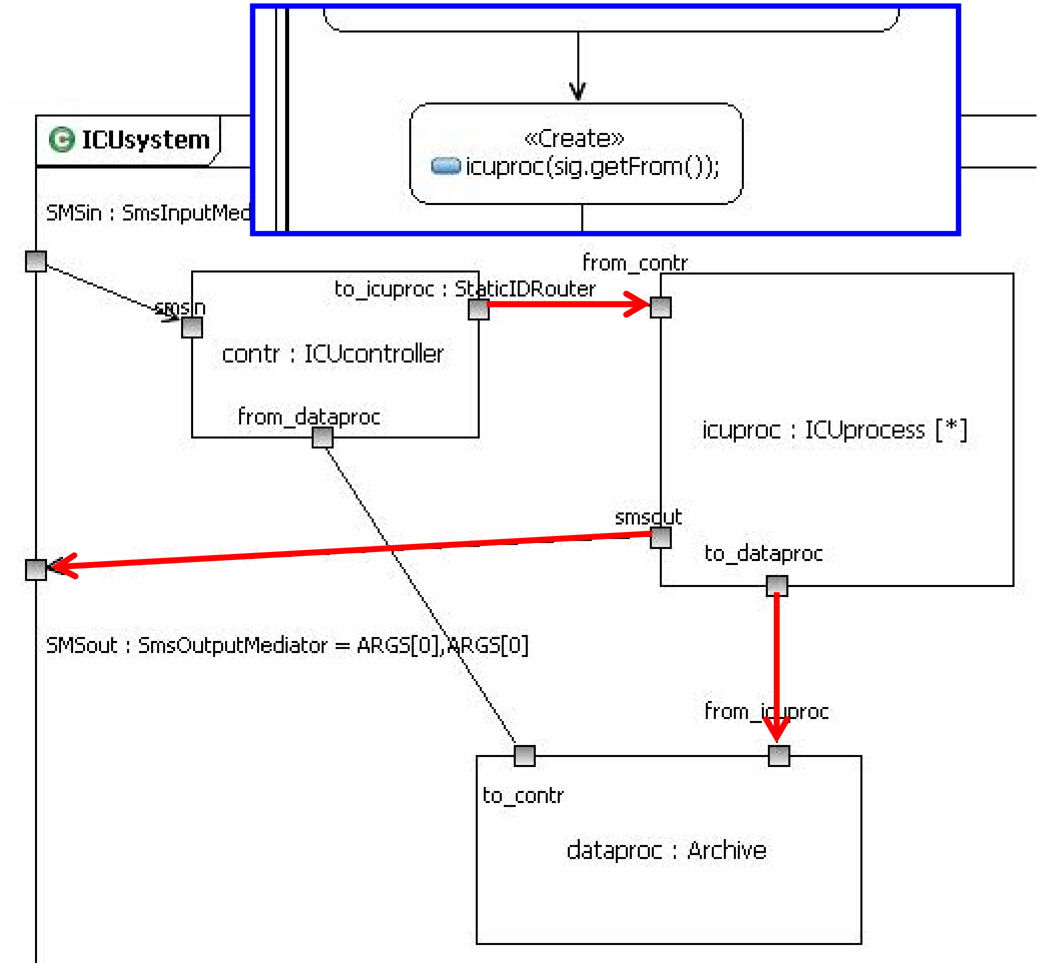

mediatorList
idList
...
...
pointers to all existing icuproc's
ids of all existing icuproc's

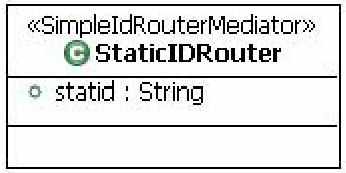
Simple Routing (2) Adding the ID

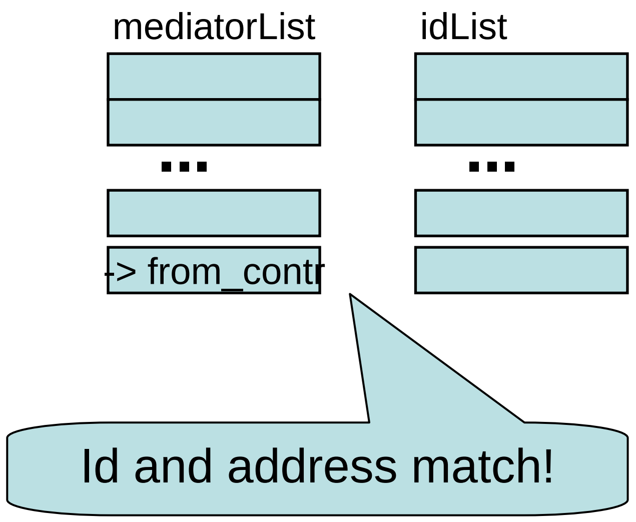

GenerateSession
/* Generate session indexed by Static ID */
csm.to_icuproc.addId(sig.getFrom());
«Create»
icuproc(sig.getFrom());
/* and give the Sms signal to the
generated process */
output(sig,csm.to_icuproc,csm);



Simple Routing (3) Connecting connectors

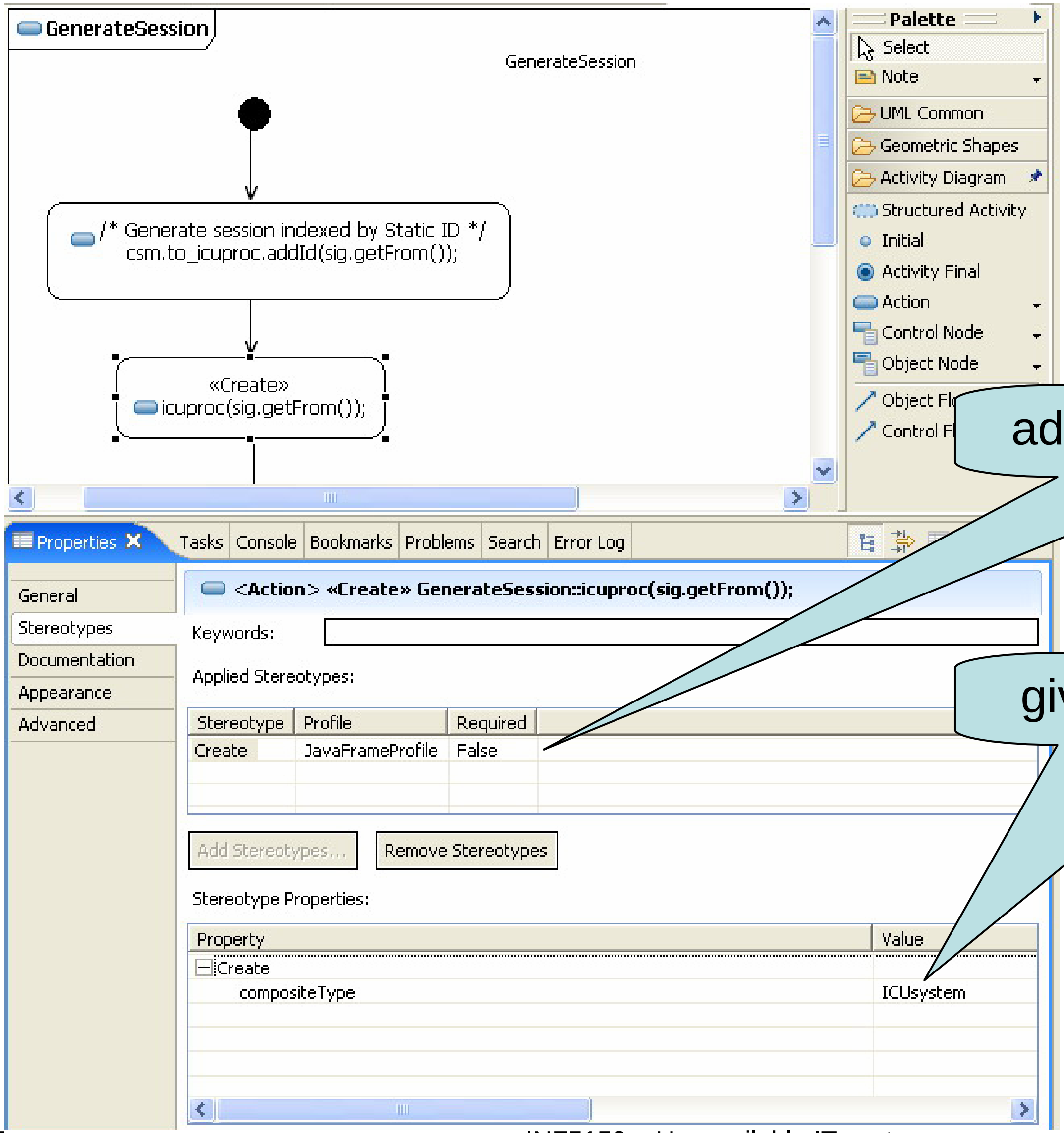






UNIVERSITY OF OSLO

Technicalities of the Create-stereotype



```
/* Generate session indexed by Static ID */
csm.to_icuproc.addId(sig.getFrom());
```

«Create»
icuproc(sig.getFrom());

<Action> «Create» GenerateSession::icuproc(sig.getFrom());

Keywords:

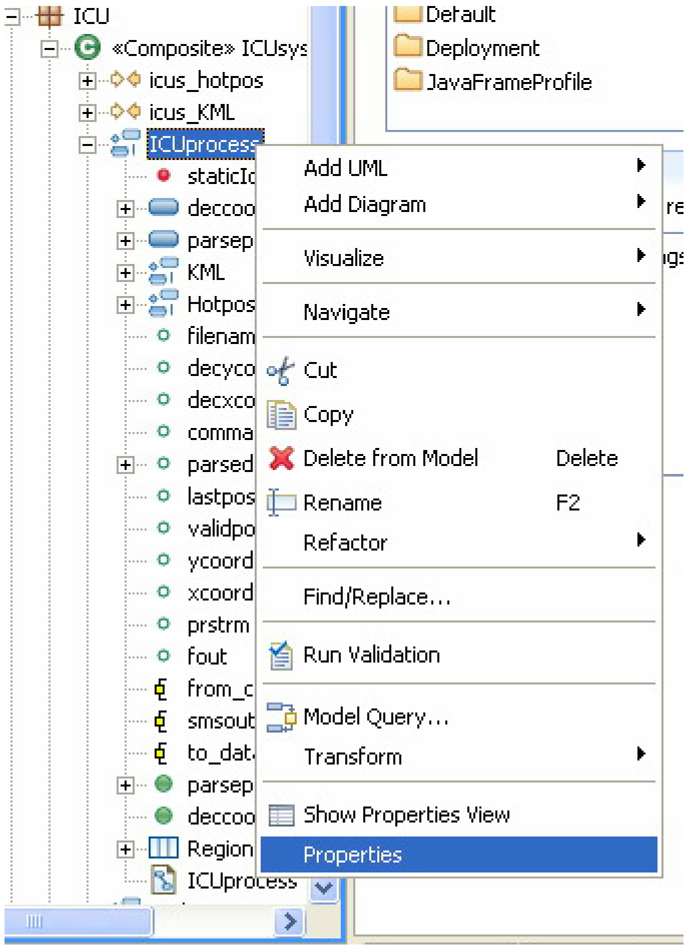
Applied Stereotypes:

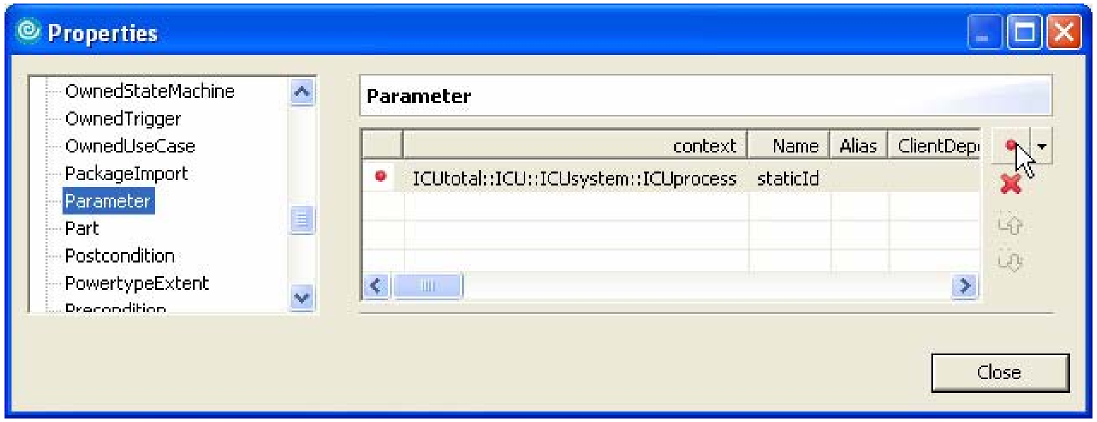
Stereotype	Profile	Required
Create	JavaFrameProfile	False

Stereotype Properties:

Property	Value
Create	
compositeType	ICUsystem

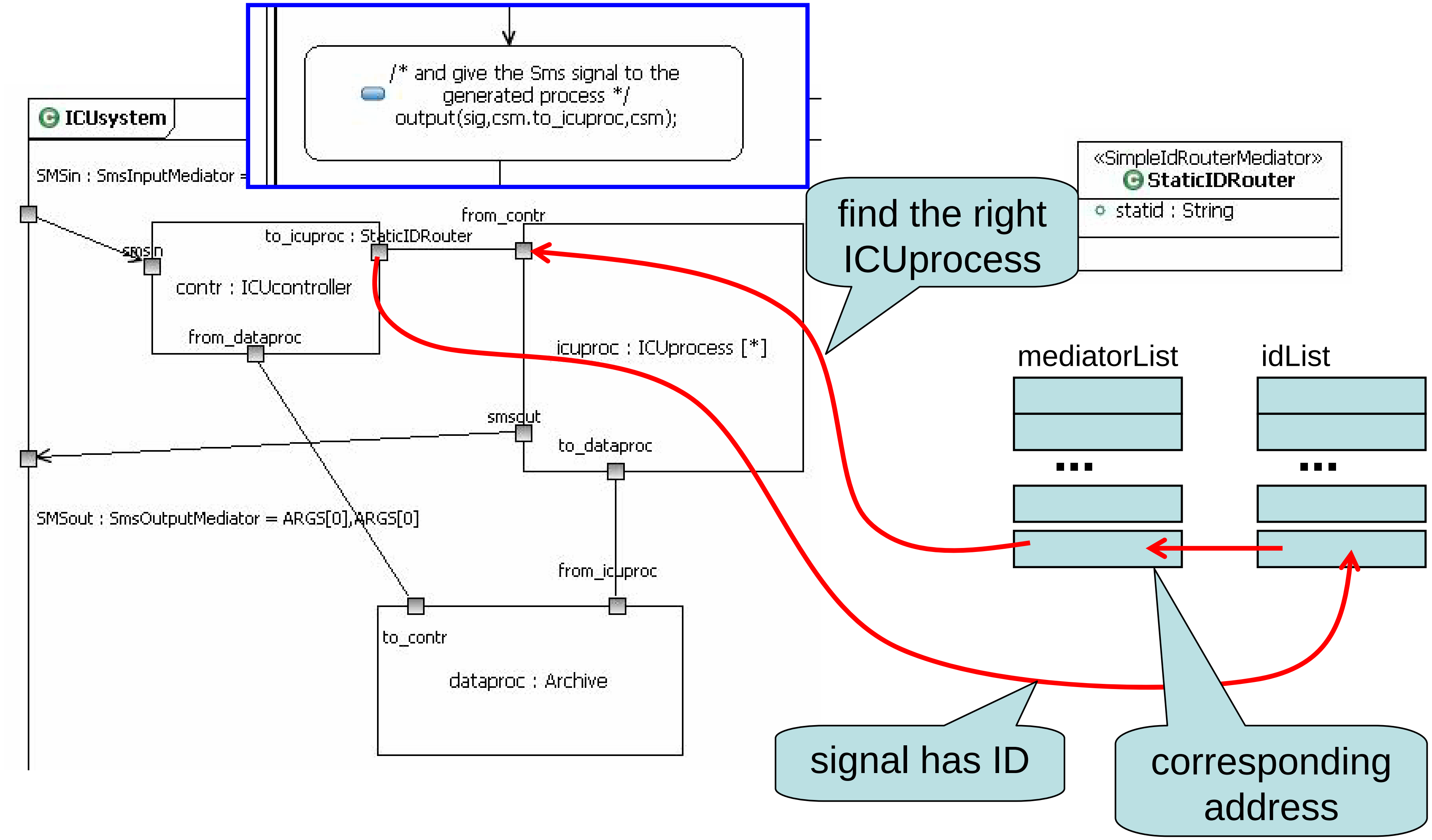
Adding a parameter to the dynamic process



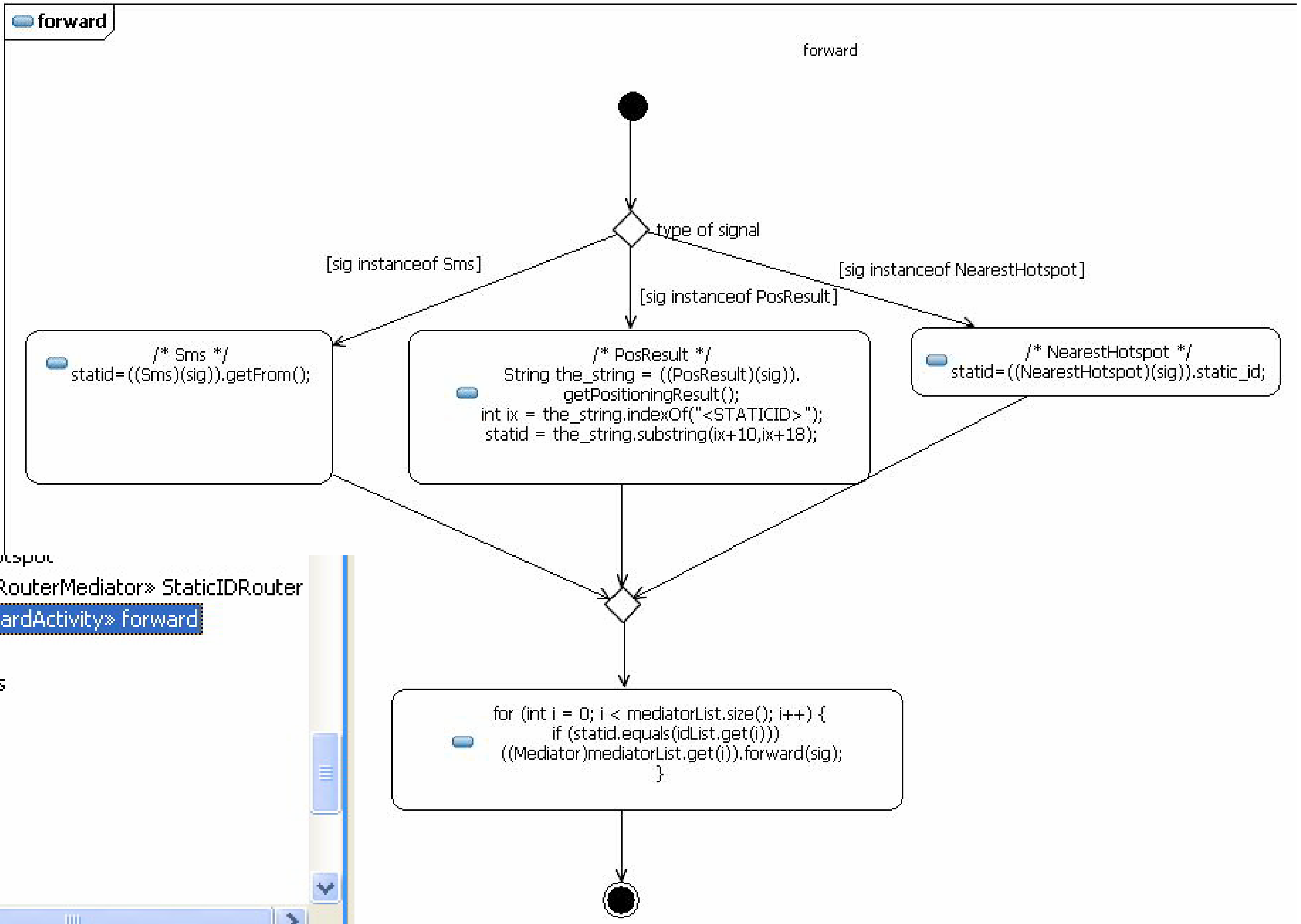


	context	Name	Alias	ClientDep
●	ICUtotal::ICU::ICUsystem::ICUprocess	staticId		

Simple Routing (4) Forwarding from Port



Simple Routing (5) forward() is programmed!

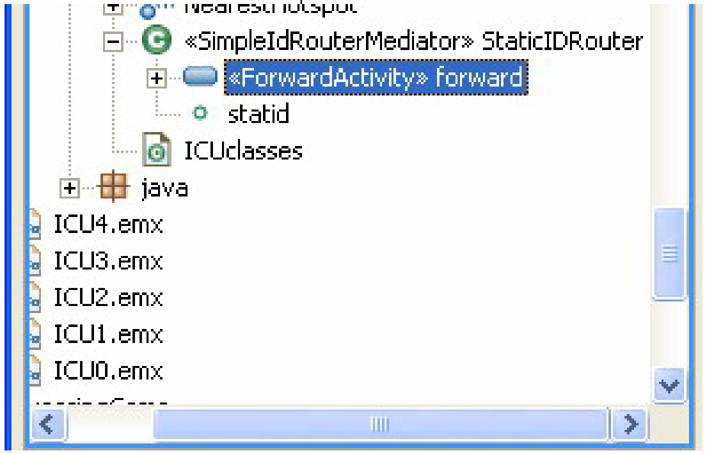


```
/* Sms */
statid=((Sms)(sig)).getFrom();
```

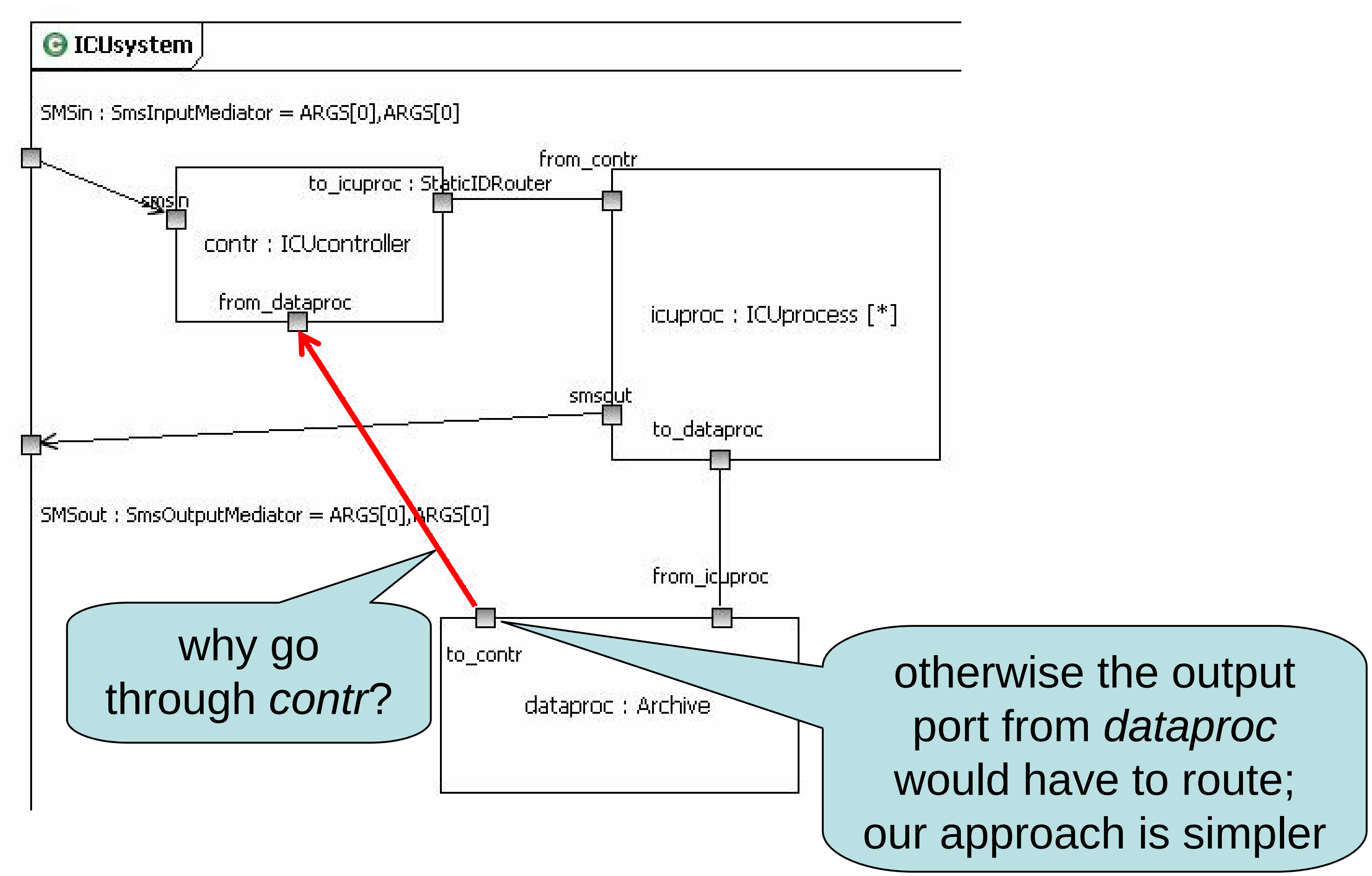
```
/* PosResult */
String the_string = ((PosResult)(sig)).
getPositioningResult();
int ix = the_string.indexOf("<STATICID>");
statid = the_string.substring(ix+10,ix+18);
```

```
/* NearestHotspot */
statid=((NearestHotspot)(sig)).static_id;
```

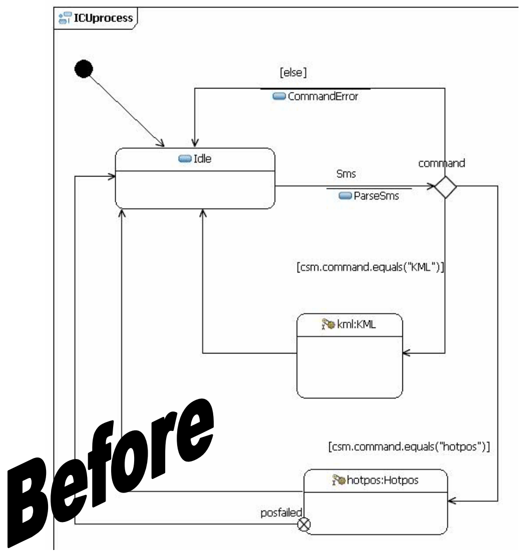
```
for (int i = 0; i < mediatorList.size(); i++) {
  if (statid.equals(idList.get(i)))
    ((Mediator)mediatorList.get(i)).forward(sig);
}
```

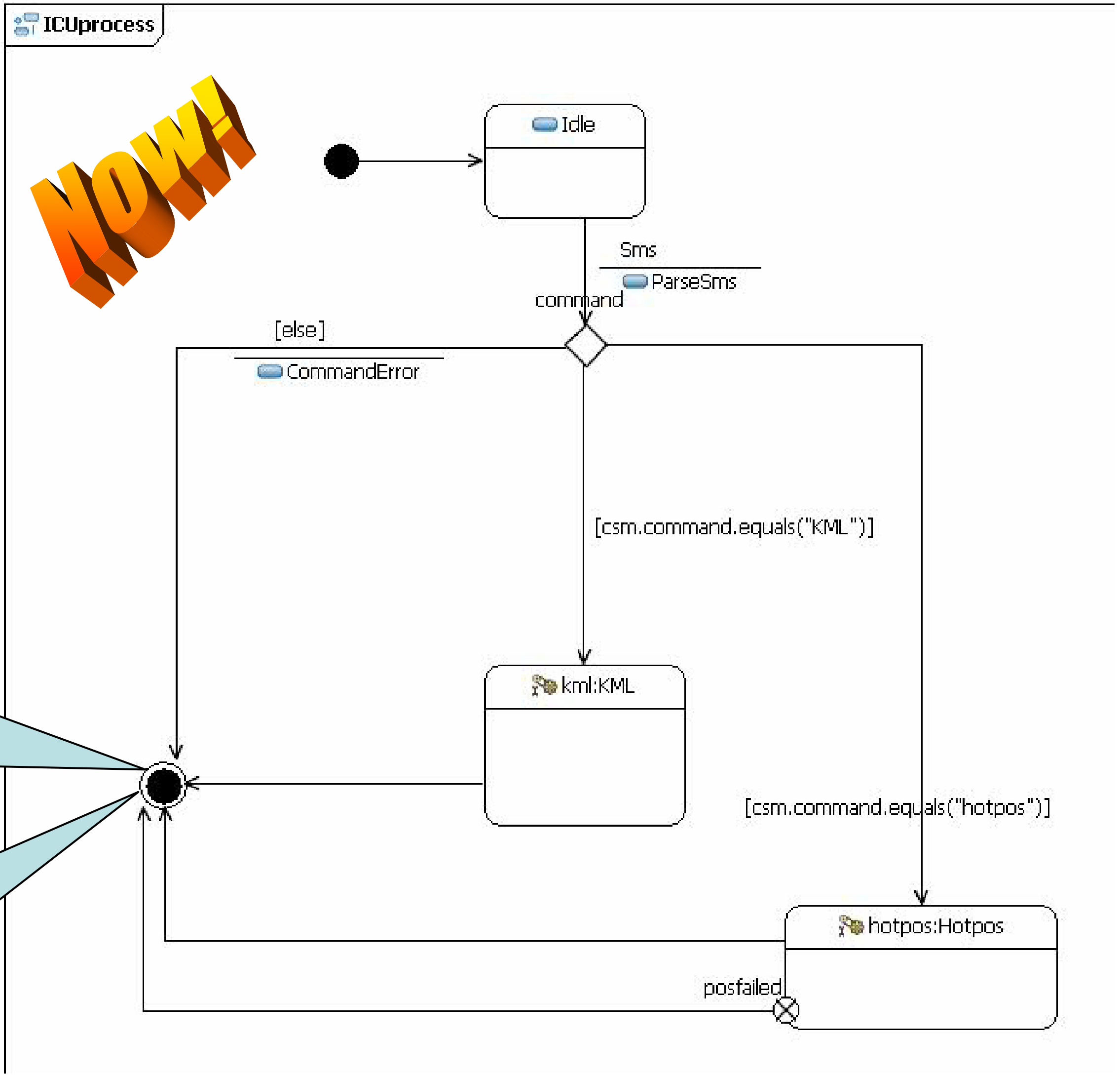


Simple Routing (6) The routing central


ICUsystem
SMSin : SmsInputMediator = ARGS[0],ARGS[0]
smsin
to_icuproc : StaticIDRouter
from_contr
contr : ICUcontroller
from_dataproc
icuproc : ICUprocess [*]
smsout
to_dataproc
SMSout : SmsOutputMediator = ARGS[0],ARGS[0]
from_icuproc
to_contr
dataproc : Archive
why go through *contr*?
otherwise the output port from *dataproc* would have to route; our approach is simpler

Terminating a session



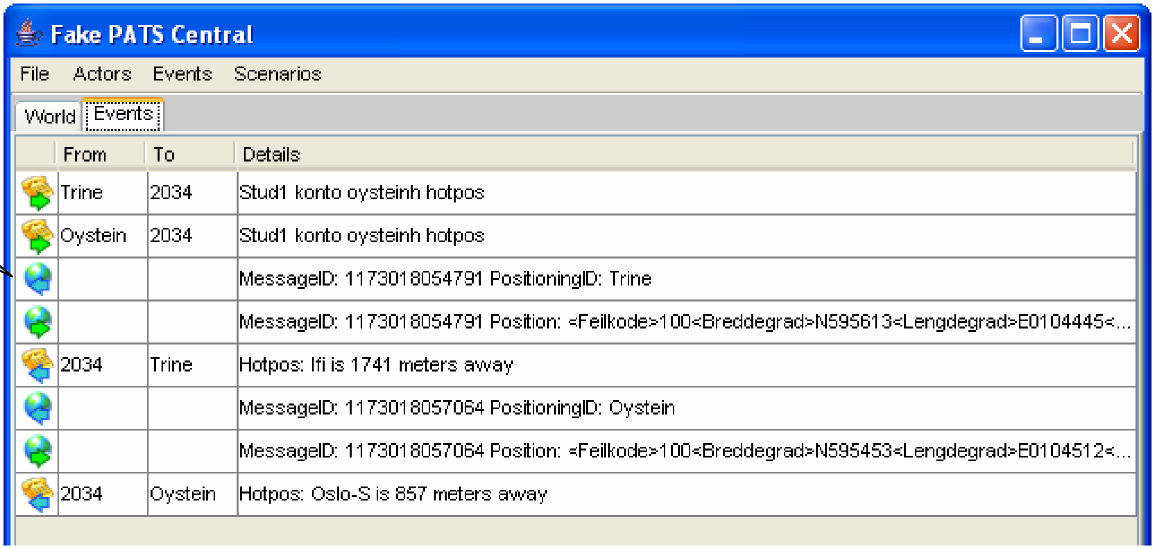


At the final state the *ICUprocess* representing the session will terminate

The compiler and JavaFrame makes sure that the implementation gets rid of the session

Executing ICU5 (with Sessions)

ICU4-DEFER: sequentialized



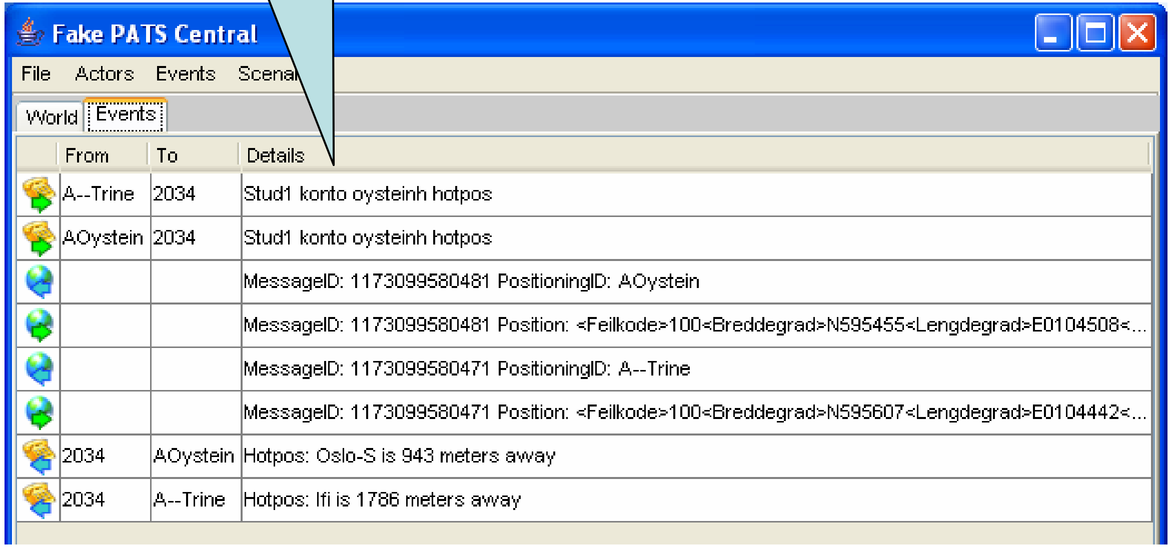
Fake PATS Central

File Actors Events Scenarios

World Events

From	To	Details
Trine	2034	Stud1 konto oysteinh hotpos
Oystein	2034	Stud1 konto oysteinh hotpos
		MessageID: 1173018054791 PositioningID: Trine
		MessageID: 1173018054791 Position: <Feilkode>100<Breddegrad>N595613<Lengdegrad>E0104445<...
2034	Trine	Hotpos: Ifi is 1741 meters away
		MessageID: 1173018057064 PositioningID: Oystein
		MessageID: 1173018057064 Position: <Feilkode>100<Breddegrad>N595453<Lengdegrad>E0104512<...
2034	Oystein	Hotpos: Oslo-S is 857 meters away

ICU5: more concurrency



Fake PATS Central

File Actors Events Scenarios

World Events

From	To	Details
A--Trine	2034	Stud1 konto oysteinh hotpos
AOystein	2034	Stud1 konto oysteinh hotpos
		MessageID: 1173099580481 PositioningID: AOystein
		MessageID: 1173099580481 Position: <Feilkode>100<Breddegrad>N595455<Lengdegrad>E0104508<...
		MessageID: 1173099580471 PositioningID: A--Trine
		MessageID: 1173099580471 Position: <Feilkode>100<Breddegrad>N595607<Lengdegrad>E0104442<...
2034	AOystein	Hotpos: Oslo-S is 943 meters away
2034	A--Trine	Hotpos: Ifi is 1786 meters away

Technicality: StaticID **must** be 8 chars

Summary of Sessions

- One session per concurrent user initiative
  - The state machine type *ICUprocess* describes the session
- One receptionist state machine creates the sessions
  - when the session initiation arrives
  - here: Sms-message
- Centralized routing through the receptionist *contr*
  - one routing port (SimpleIdRouterMediator)
  - all signals aiming for a session are sent through *contr*
- Terminating the session by reaching the final state
  - and the runtime system machinery takes care of the rest