UNIVERSITY
OF OSLO

Robustness –
the art of preparing for the unexpected

Version 091120

INF 5150

The exceptional

- Data may have strange syntax or values
  - we apply common data-parsing techniques
- An unexpected signal arrives
  - we explicitly describe every conceivable transition
- No signal arrives
  - we guard our protocols with timers
- Security issues
  - authentication + logging + statistics
- Availability issues
  - self tests

ICUcontroller: the exceptional

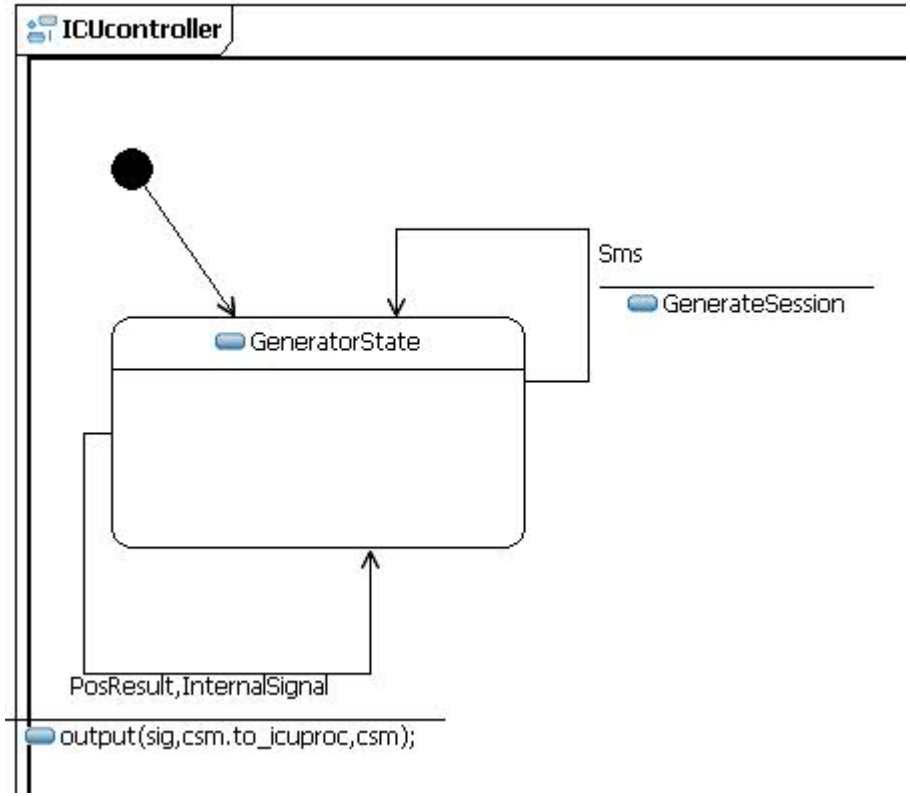
- Data may have strange syntax or values
  - checking for static id already in use
- An unexpected signal arrives
  - ICUcontroller handles Sms, PosResult and InternalSignal in all states
  - We are going to look at unexpected signals for ICUprocess
- No signal arrives
  - ICUcontroller does not have such waiting situations (?)
  - we shall guard our protocols/services with timers (ICUprocess)
- Security issues
  - authentication + logging + statistics
    - Authentication is not needed to enter ICUcontroller
    - we are going to check for registration in ICUprocess
- Availability issues
  - self tests
    - We could use ICUcontroller to test availability of PATS (but don't)
  - we will consider this with the Archive

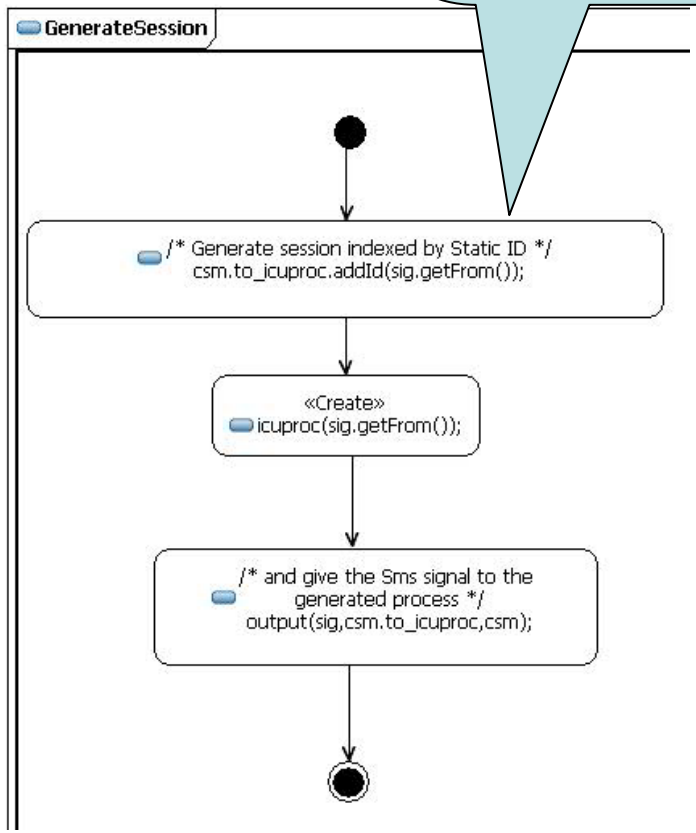
Handling an error or exceptional situation

- The invalid situation is due to an inadequate user input
  - then we know what caused it and the user should be notified
    - ICU: The user is notified by an SMS
- The invalid situation is due to an internal error
  - the reason is unclear, but the situation has become erroneous
  - The correct recovery may be hard to specify, but we believe that terminating the whole program is probably the last resort
    - ICU: different responses:
      - Try and send SMS to the user (if the appropriate user is known)
      - Dump the call stack on console (syserr) (very low level)
      - Terminate the session (and notify the session owner by an SMS)

ICUcontroller's GenerateSession

Data invariant is that Static ID should not already be used





INF 5150

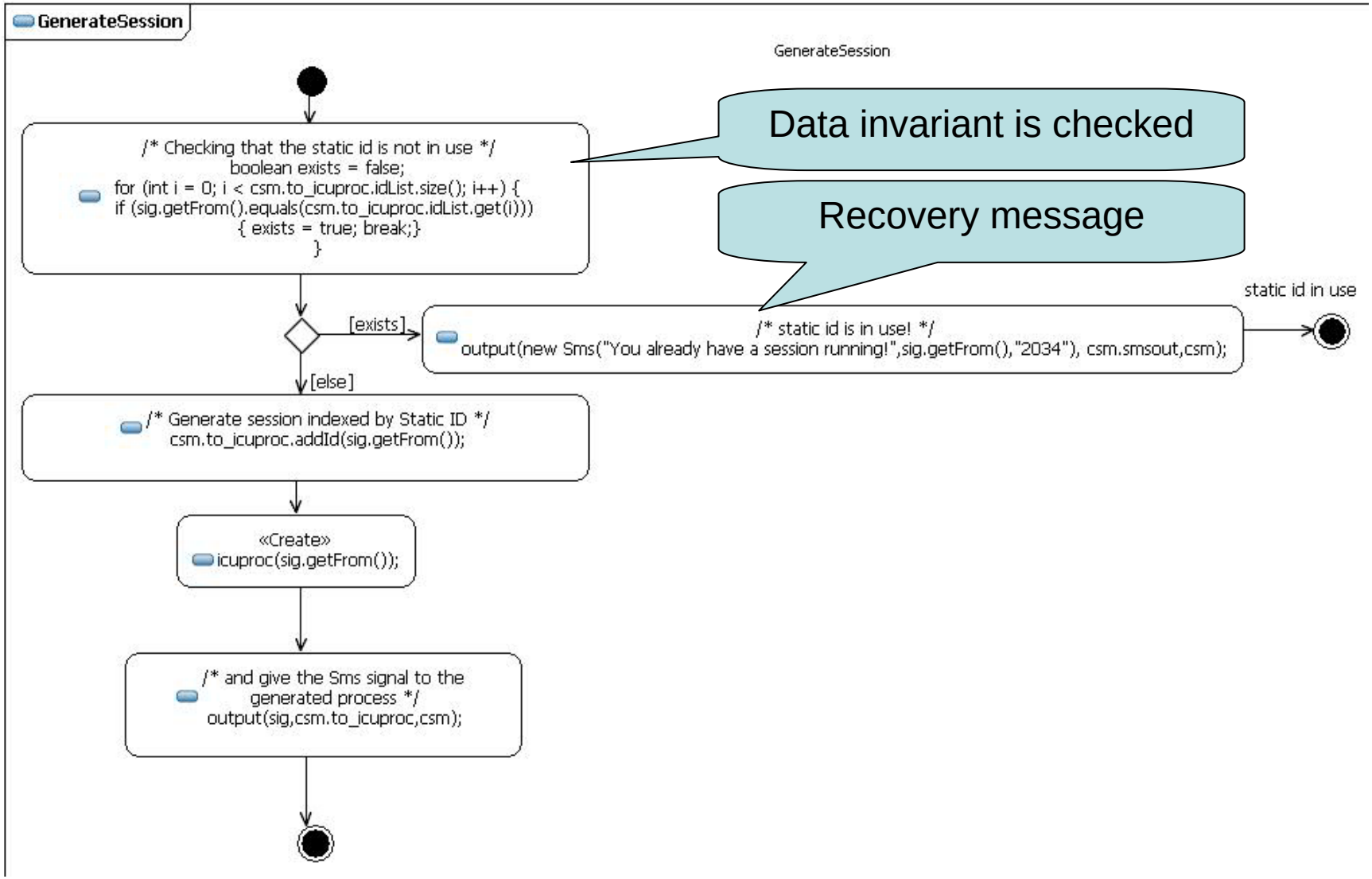
Checking the data invariant

- Our task is to check whether STAT-ID is already the ID of another ICUprocess
- Here are the checking strategies:
  - checking directly the data of the routing port
    - simple, but on low (Java) level
  - sending a probe signal and wait for its possible consequences
    - more protocol needed, and possibly changing the forward() operation
    - if the normal response is that a timer must expire, this will be slow
  - recording which static ids are active (in the Archive)
    - lots of book-keeping, slower, overkill
- We go for the simple java-oriented solution this time

Error recovery for the static id re-use

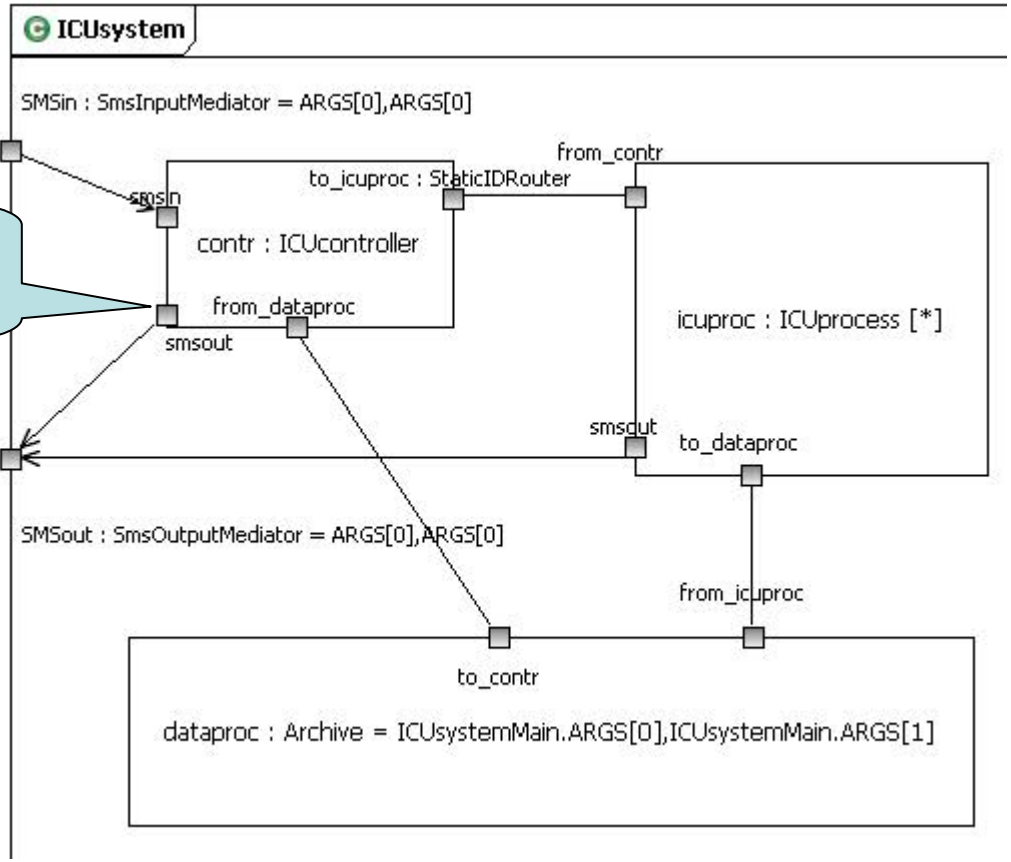
- The cause of the static id re-use is most probably because the user has sent two requests in quick sequence
- We should respond by returning an error message to the user
  - This will imply fixing the composite structure
- Move to a final state
  - in our service-oriented architecture, the service session is the natural unit of recovery, i.e. canceling the current service session is often the best approach

The robust ICUcontroller's GenerateSession



Modified Composite Structure

Port and Connector added



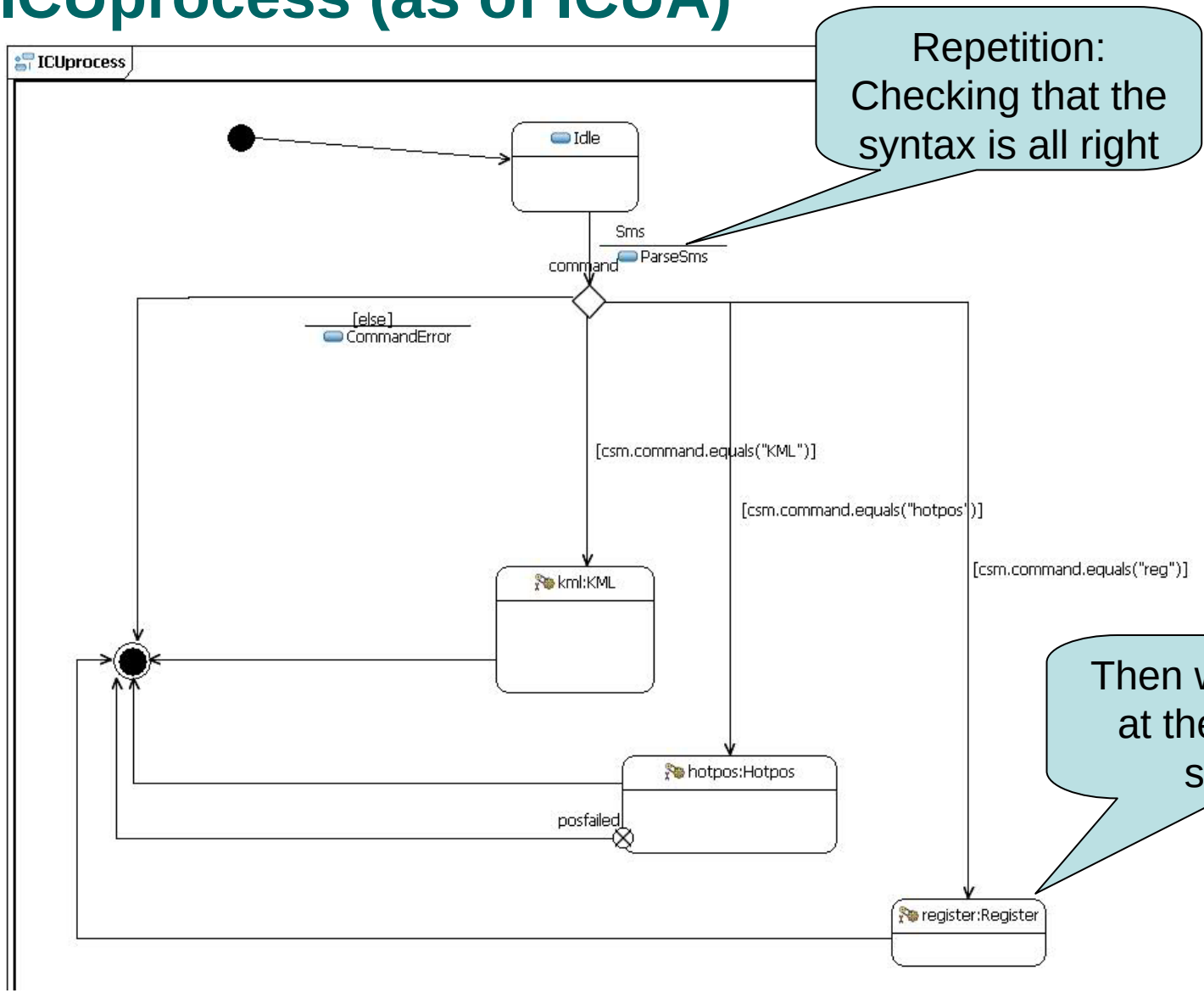
ICUProcess: the exceptional

- Data may have strange syntax or values
  - Checking the correct syntax of the SMS
- An unexpected signal arrives
  - We are going to look at unexpected signals for ICUprocess
- No signal arrives
  - we shall guard our protocols/services with timers (ICUprocess)
- Security issues
  - authentication + logging + statistics
    - Authentication is not needed to enter ICUcontroller
    - we are going to check for registration in ICUprocess
- Availability issues
  - self tests
    - We could use ICUcontroller to test availability of PATS (but don't)
  - we will consider this with the Archive

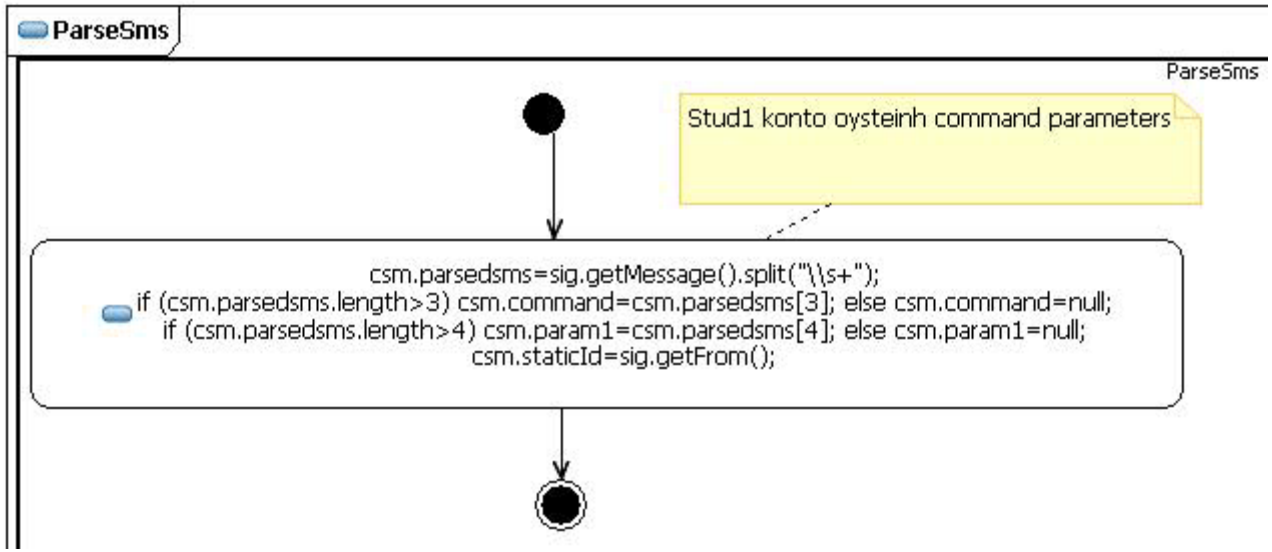
Explicit transitions please!

- Finite State Machines have a great advantage by being finite!
  - there is a finite set of transitions to execute
  - we can make sure to cover them all
- UML State Machines also define default transitions
  - where the signal is just discarded/consumed
  - We believe that default transitions are a warning of design flaw
- Not all signals can be properly handled at any time
  - We may defer a signal to a state where the signal can be dealt with

ICUprocess (as of ICUA)



ParseSms

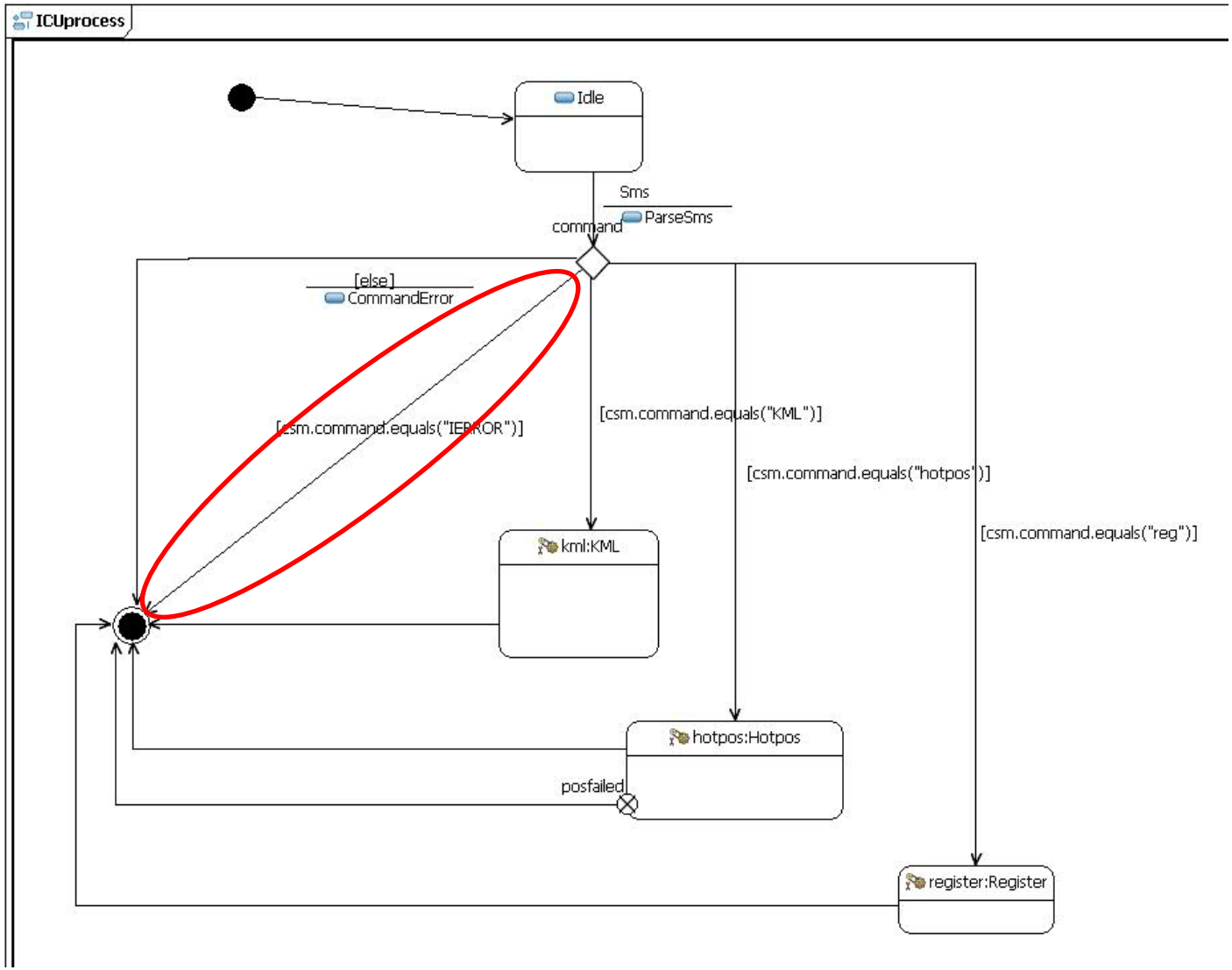


- If the Sms does not start with ”Stud1 konto *username*” it will not come to the program at all
  - Still we may choose to check for it – due to running on FakePats
- If there are more than 1 parameter, there is also an error
  - at least for the set of services of ICU that we have up to now
- We should give user syntax error messages right away
  - and not hide it by letting command be null

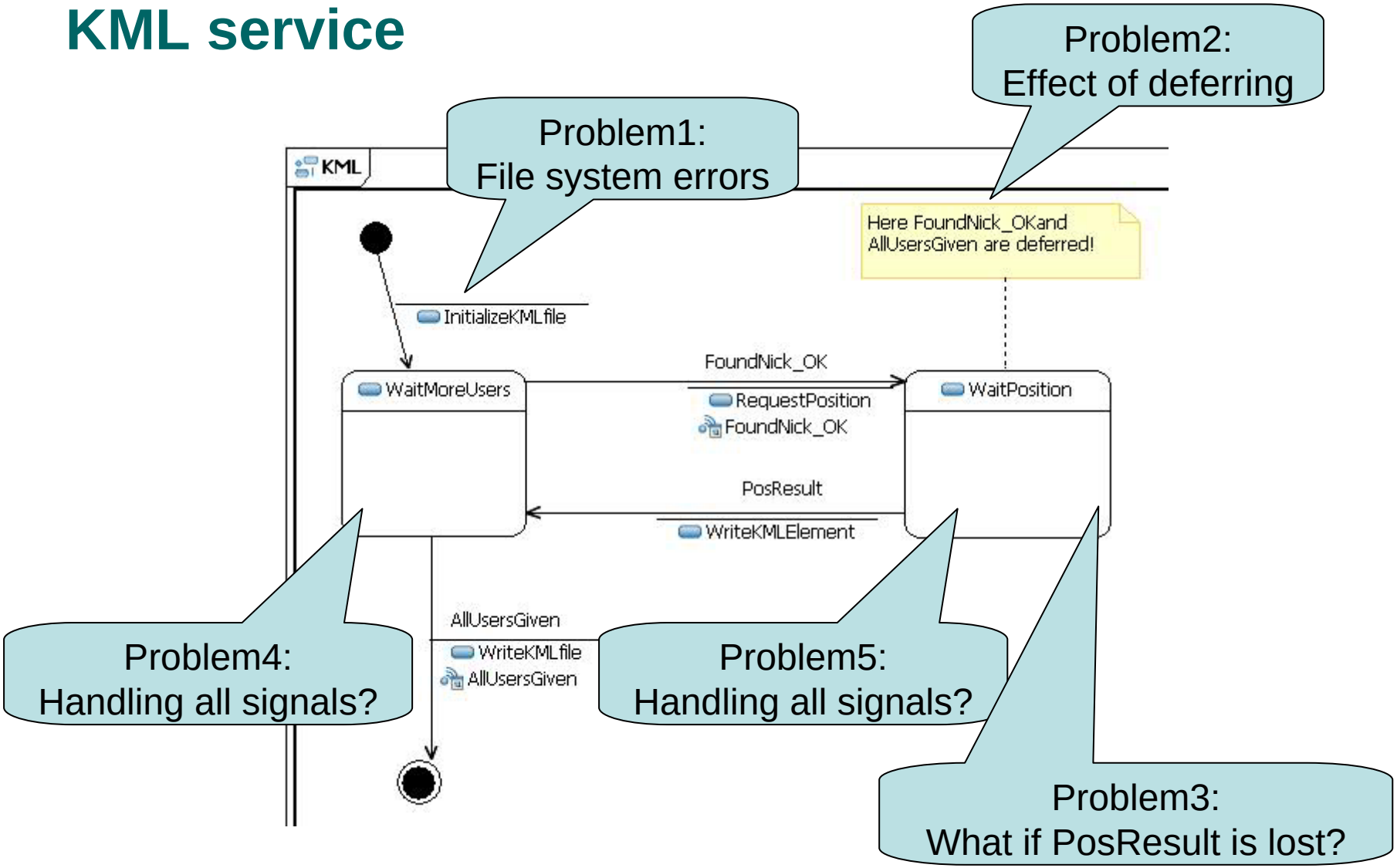
ParseSms robustified (1)

```
csm.parsedsms=sig.getMessage().split("\\s+");

/* check for existence of necessary prefix - very rudimentary */
if (csm.parsedsms.length<=3)
{ output(new Sms("ICU: Syntax error - no command",sig.getFrom(),"2034"), csm.smsout,csm);
   csm.command = "IERROR";
}
else
{ csm.command=csm.parsedsms[3];

  /* check for only one parameter */
  if (csm.parsedsms.length>5)
  { output(new Sms("ICU: Too many parameters!",sig.getFrom(),"2034"), csm.smsout,csm);
    csm.command = "IERROR";
  }
  else
  { if (csm.parsedsms.length>4) csm.param1=csm.parsedsms[4]; else csm.param1=null;
  }
}
csm.staticId=sig.getFrom();
```

ParseSms robustified (2)



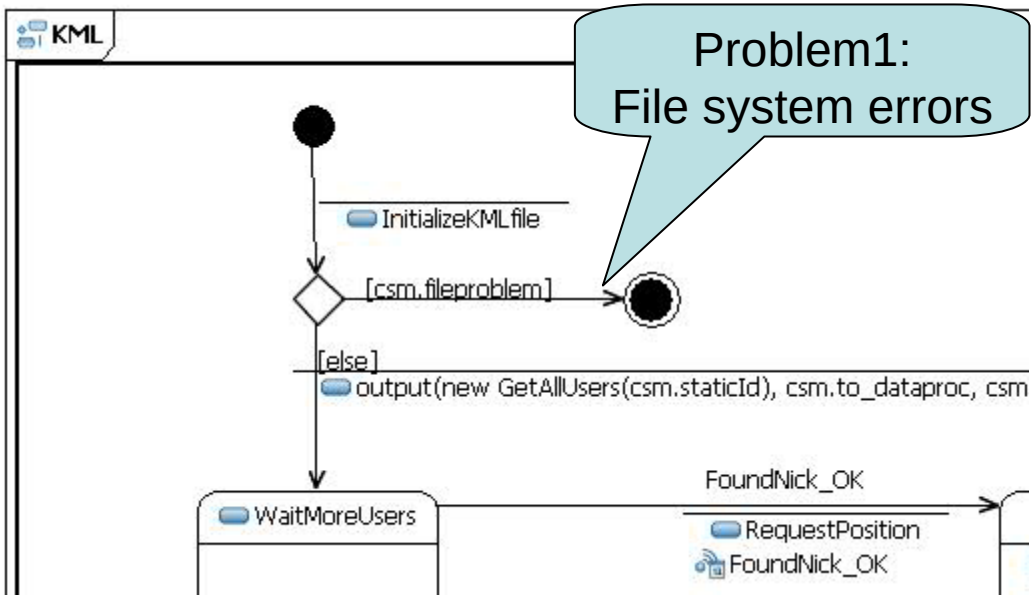
KML service


Problem2:
Effect of deferring
Problem1:
File system errors
KML
Here FoundNick_OKand
AllUsersGiven are deferred!
InitializeKMLfile
FoundNick_OK
WaitMoreUsers
RequestPosition
FoundNick_OK
WaitPosition
PosResult
WriteKMLElement
AllUsersGiven
WriteKMLfile
AllUsersGiven
Problem4:
Handling all signals?
Problem5:
Handling all signals?
Problem3:
What if PosResult is lost?

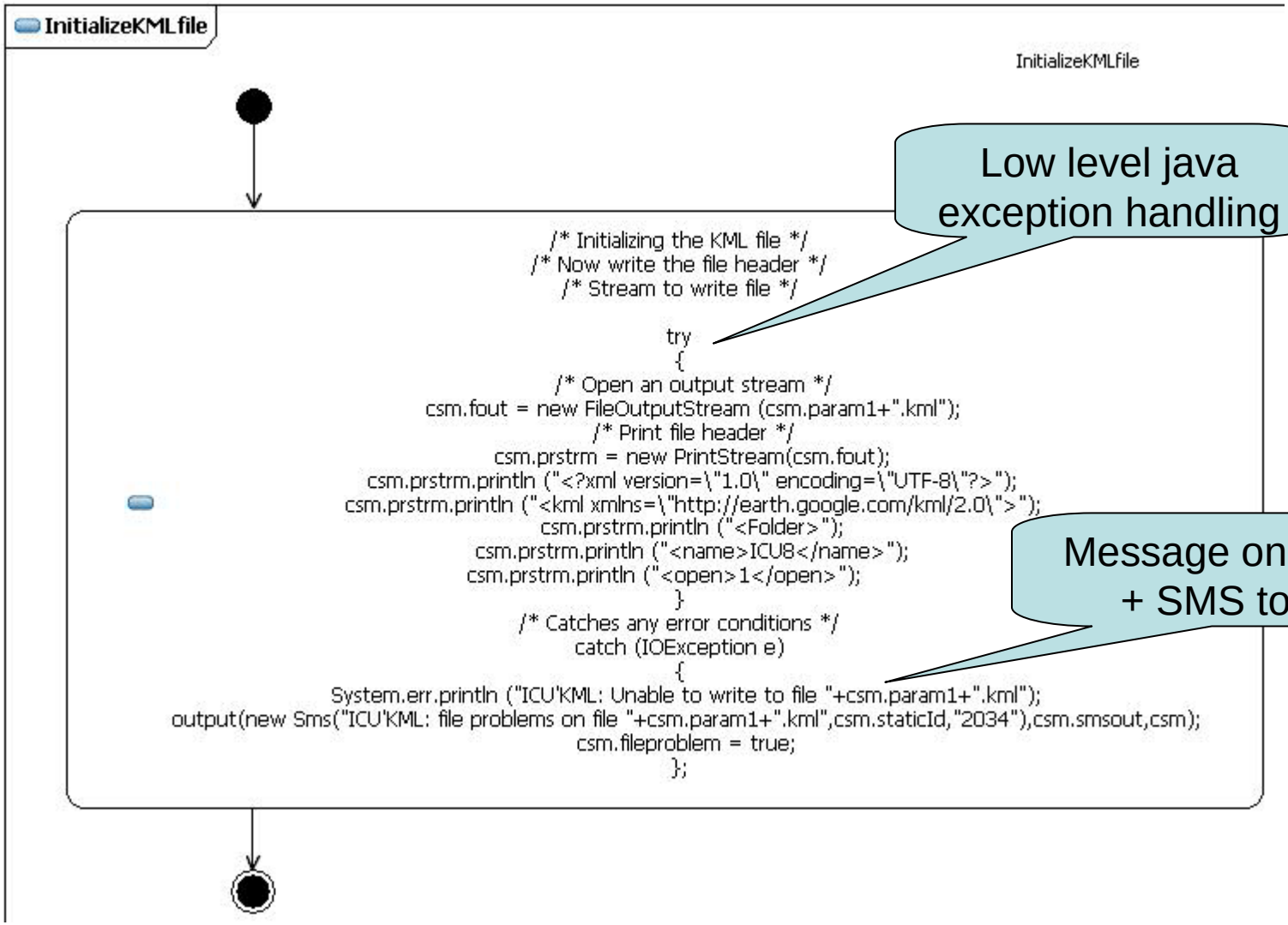
KML problems (1)

- 1: File writing problems
  - currently error dumped on console, and proceed as if no problem has arisen
    - not adequate: if the initialization fails to write on the file, the session should terminate, and double messages given (to console and user)

1: File writing problems – New KML machine



1: File writing problems (2)



KML problems (2)

- 2: The deferring of FoundNick_OK is motivated by wanting to handle one positioning at the time
  - but the effect is the need to handle many *defers*
    - since the database produces users faster than PATS positions them
    - actually the #defers are in the order of #users^2
  - and decreased efficiency due to this defer-handling and since positioning requests may be done in parallel (possibly)
    - but in fact sending too many positioning requests very quickly seems to stress PATS such that sometimes requests are lost
  - The optimal solution may be to introduce a little more protocol to sequentialize such that the Archive is explicitly asked to give the next user
    - rather than giving all users in a stream of messages
    - ... but we keep to the *defer* solution – to show in detail how *defer* is

JFTrace of the deferring KML

Filtered Trace from /127.0.0.1:54321 at 2007-04-29 23:48:08.759

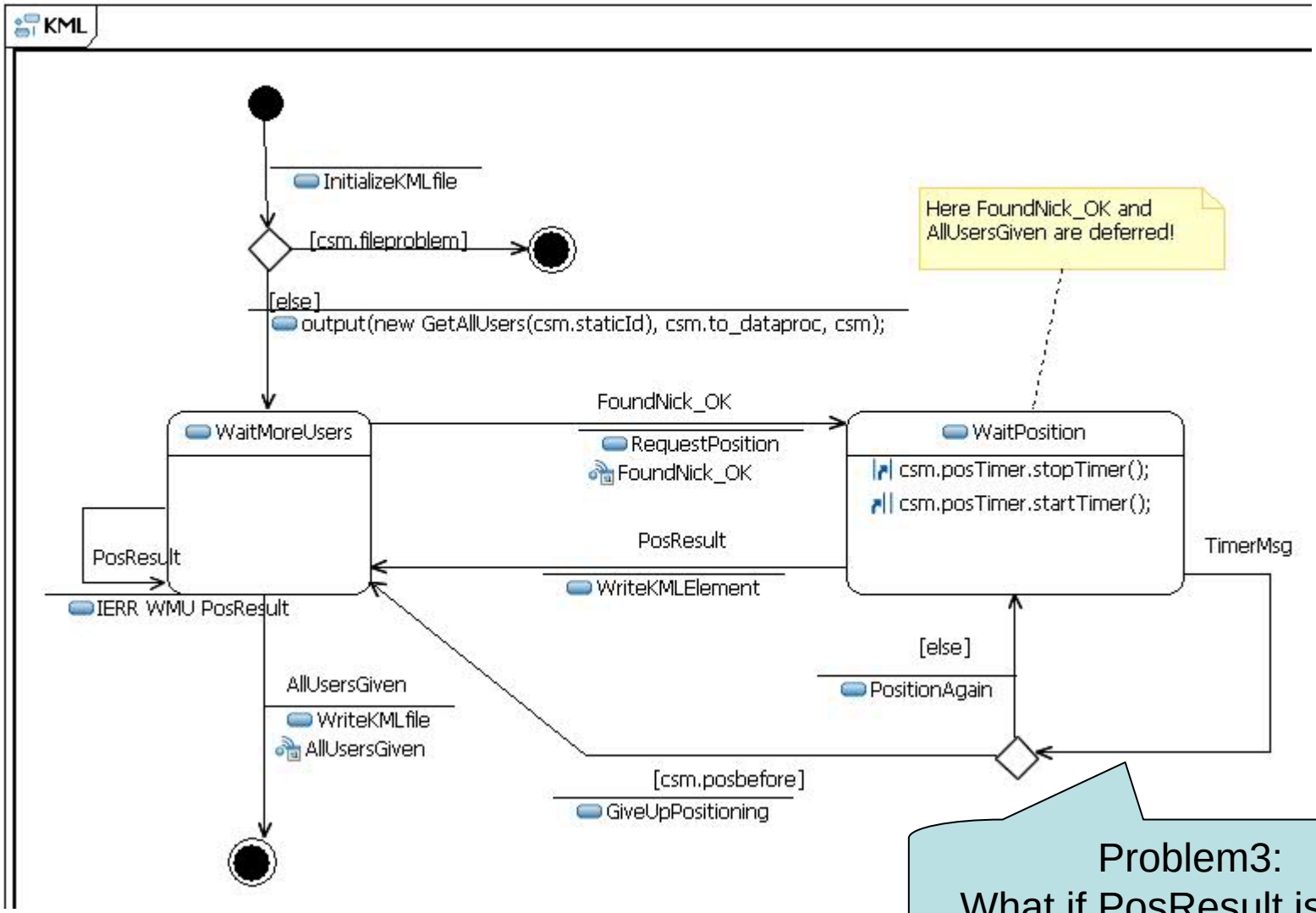
Table View

Time	State Machine	Current State	Input	Transition Behaviour	Next State
0	New ICUsystem_Archive@4e224cb9				
0	New ICUsystem_ICUcontroller@40...				
1362	ICUsystem_Archive@4e224cb9	null	StartMessage@4f430cb9		Idle
3125	ICUsystem_ICUcontroller@406f0cb9	null	StartMessage@41278cb9		GeneratorState
28541	ICUsystem_ICUcontroller@406f0cb9	GeneratorState	Sms@3e18cba (Stud1 konto oysteinh KML,2034,91390900)	New ICUsystem_ICUprocess@1dadccba Output Sms@3e18cba (Stud1 konto oysteinh KML,2034,91390900)	GeneratorState
28872	ICUsystem_ICUprocess@1dadccba	null	StartMessage@1d884cba		Idle
28872	ICUsystem_ICUprocess@1dadccba	Idle	Sms@3e18cba (Stud1 konto oysteinh KML,2034,91390900)	Output GetAllUsers@298accba (91390900)	WaitMoreUsers^kml
29032	ICUsystem_Archive@4e224cb9	Idle	GetAllUsers@298accba (91390900)	Output FoundNick_OK@2fc1ccba (No91390900, 91390900, 91390900) Output FoundNick_OK@2e7c8cba (No66688899, 66688899, 91390900) Output FoundNick_OK@3bad8cba (No09090909, 09090909, 91390900) Output AllUsersGiven@3a474cba (91390900)	Idle
29112	ICUsystem_ICUcontroller@406f0cb9	GeneratorState	FoundNick_OK@2fc1ccba (No91390900, 91390900, 91390900)	Output FoundNick_OK@2fc1ccba (No91390900, 91390900, 91390900)	GeneratorState
29112	ICUsystem_ICUcontroller@406f0cb9	GeneratorState	FoundNick_OK@2e7c8cba (No66688899, 66688899, 91390900)	Output FoundNick_OK@2e7c8cba (No66688899, 66688899, 91390900)	GeneratorState
29523	ICUsystem_ICUcontroller@406f0cb9	GeneratorState	FoundNick_OK@3bad8cba (No09090909, 09090909, 91390900)	Output FoundNick_OK@3bad8cba (No09090909, 09090909, 91390900)	GeneratorState
30083	ICUsystem_ICUcontroller@406f0cb9	GeneratorState	AllUsersGiven@3a474cba (91390900)	Output AllUsersGiven@3a474cba (91390900)	GeneratorState
30134	ICUsystem_ICUprocess@1dadccba	WaitMoreUsers^kml	FoundNick_OK@2fc1ccba (No91390900, 91390900, 91390900)	Output PosRequest@be80cb9	WaitPosition^kml
30204	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	FoundNick_OK@2e7c8cba (No66688899, 66688899, 91390900)		Saved
30204	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	FoundNick_OK@3bad8cba (No09090909, 09090909, 91390900)		Saved
30204	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	AllUsersGiven@3a474cba (91390900)		Saved
32176	ICUsystem_ICUcontroller@406f0cb9	GeneratorState	PosResult@31584cb9	Output PosResult@31584cb9	GeneratorState
32207	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	PosResult@31584cb9		WaitMoreUsers^kml
32397	ICUsystem_ICUprocess@1dadccba	WaitMoreUsers^kml	FoundNick_OK@2e7c8cba (No66688899, 66688899, 91390900)	Output PosRequest@53838cb9	WaitPosition^kml
32557	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	FoundNick_OK@3bad8cba (No09090909, 09090909, 91390900)		Saved
32637	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	AllUsersGiven@3a474cba (91390900)		Saved
34390	ICUsystem_ICUcontroller@406f0cb9	GeneratorState	PosResult@54868cb9	Output PosResult@54868cb9	GeneratorState
34430	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	PosResult@54868cb9		WaitMoreUsers^kml
34670	ICUsystem_ICUprocess@1dadccba	WaitMoreUsers^kml	FoundNick_OK@3bad8cba (No09090909, 09090909, 91390900)	Output PosRequest@6c4dccb9	WaitPosition^kml
34750	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	AllUsersGiven@3a474cba (91390900)		Saved
35872	ICUsystem_ICUcontroller@406f0cb9	GeneratorState	PosResult@78c88cb9	Output PosResult@78c88cb9	GeneratorState
35902	ICUsystem_ICUprocess@1dadccba	WaitPosition^kml	PosResult@78c88cb9		WaitMoreUsers^kml
35902	ICUsystem_ICUprocess@1dadccba	WaitMoreUsers^kml	AllUsersGiven@3a474cba (91390900)	Output Sms@6d174cb9 (null.kml:E0104541,N595627,91390900,2034)	FinalState

KML problems (3) – needing timers

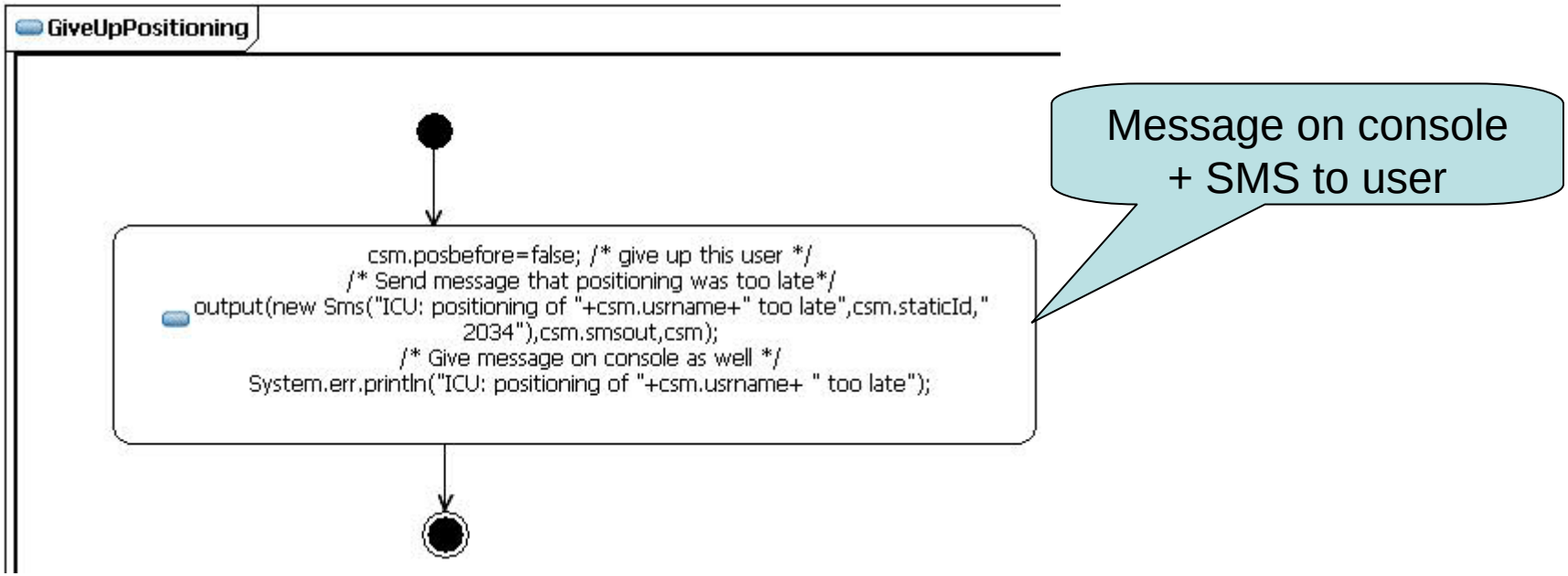
- 3: We have no guarantee that the PosRequest eventually results in a corresponding PosResult
  - We shall have to guard the PosResult by a timer
  - What then to do if the guarding timer expires?
    - Giving an SMS to the user for every non-positioned phone may be too many SMSes
      - and we could cut off after a small number of such messages (say 3)
        - and then give a more general error message and terminate KML session
    - We could try again to position the failed one (one retry)
  - What if the timer has expired, recovery has been done, and then the PosResult appears very late?
    - In our case this will have a cascading effect of PosResult appearing when it should not
      - this actually becomes rather tricky! (will be covered later)

3: Including the timer in KML



Problem3:
What if PosResult is lost?

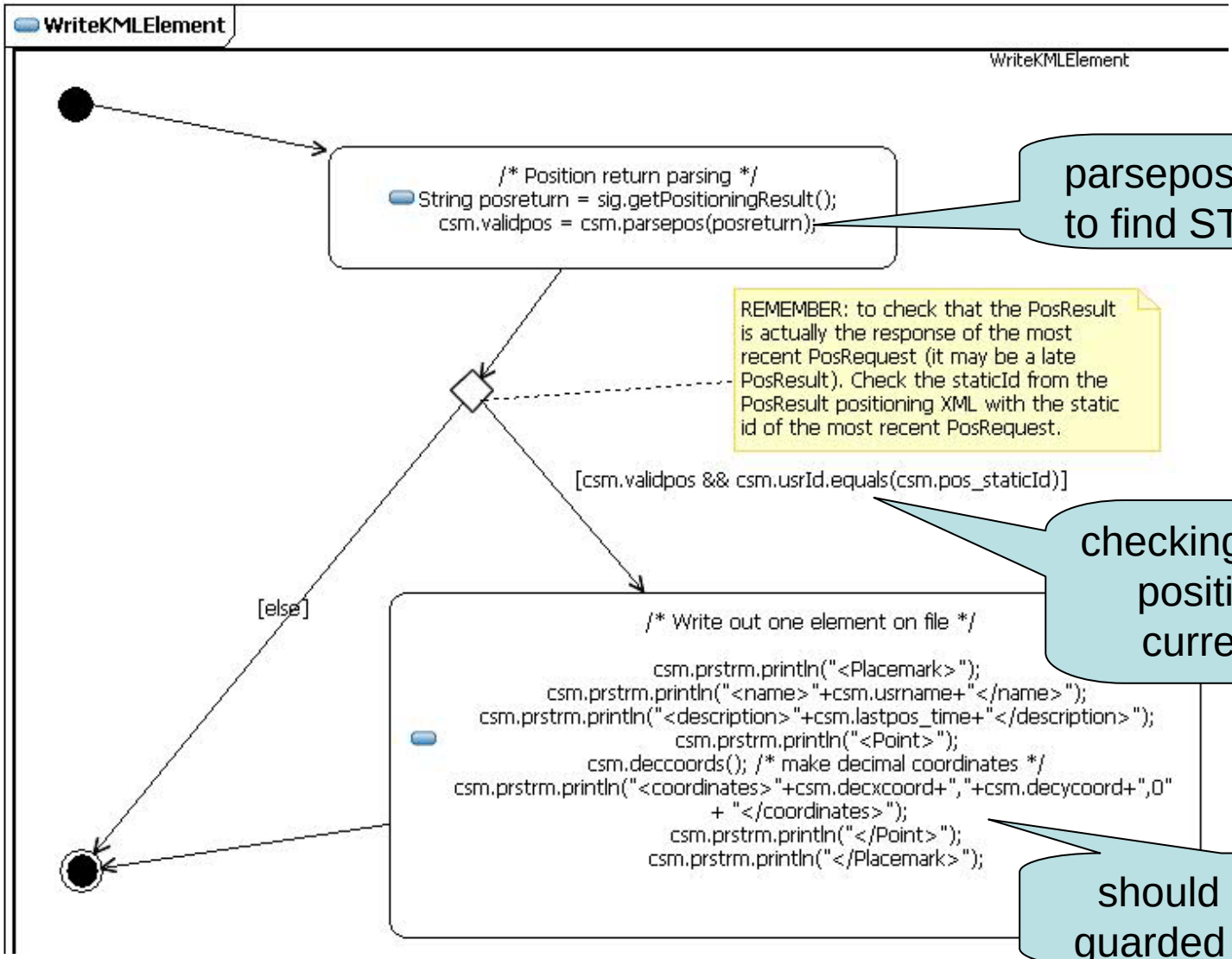
3: Giving up positioning after one re-try



Timer expires

- To give PATS one more chance with this user
  - we need to define a variable to control re-positioning
- Having given up we must still cope with the PosResult coming later
  - This is more tricky than meets the eye since
    - when positioning is given up there is normally several FoundNick_OK signals in the queue
    - and a late PosResult will follow those, but
    - that PosResult may come before any PosResult that is the result of new PosRequests
  - Thus, we must make sure that the PosResult is actually matched with the right nickname
    - We need to check the static id of the PosResult with that of the most recent PosRequest

Checking the static id of the PosResult



parsepos is extended to find STATICID, too

checking the received position against current request

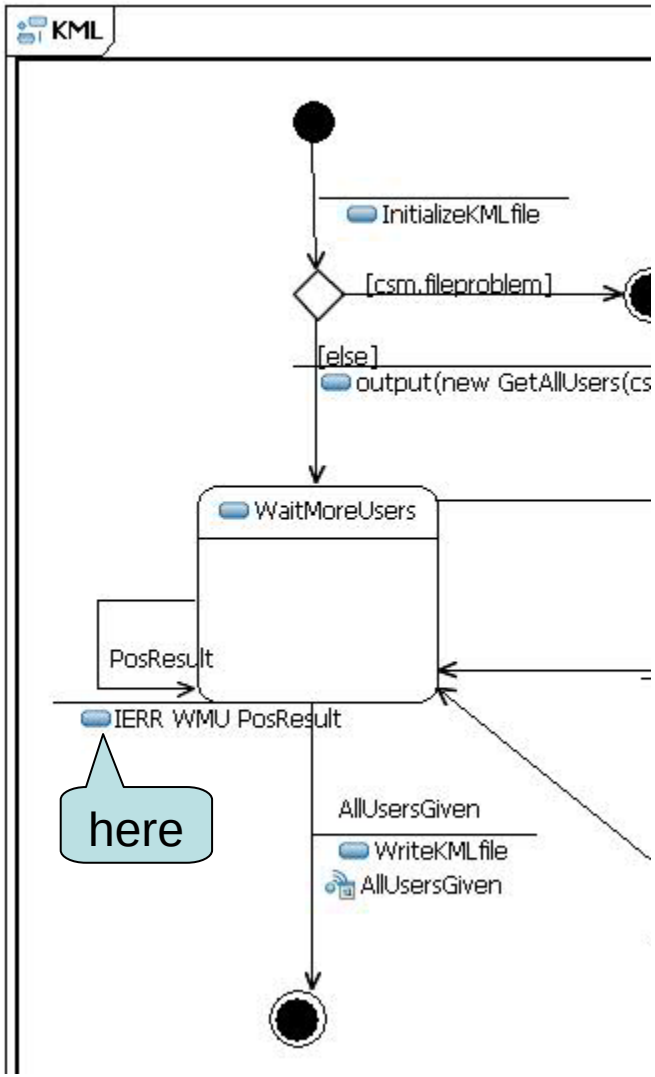
should have been guarded by try/catch

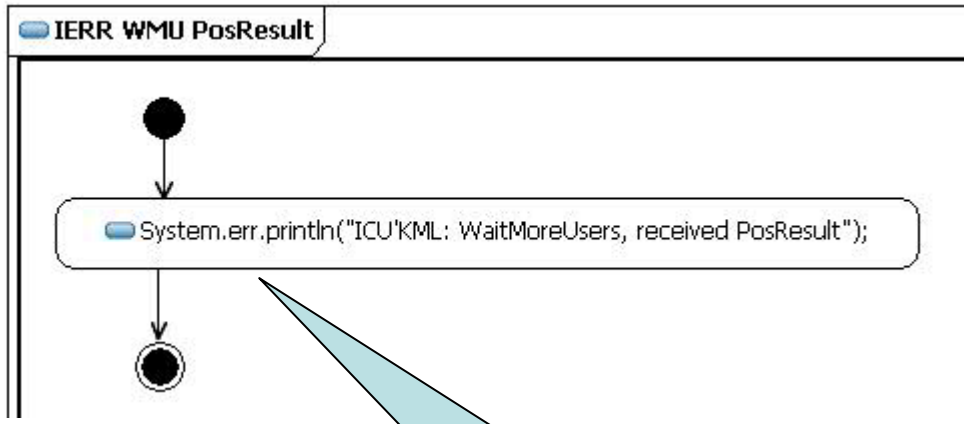
INF 5150

KML problems (4 & 5)

- 4: Default transitions of WaitMoreUsers
  - PosResult and Sms are not handled
    - Sms cannot come to KML =>
      - internal error, handled on enclosing level
    - PosResult should (normally) not come =>
      - internal sequencing error, give message on console and ignore signal
- 5: Default transitions of WaitPosition
  - There are non-KML signals that should be covered (as Sms)
    - we will cover that on enclosing level

4: PosResult received at WaitMoreUsers





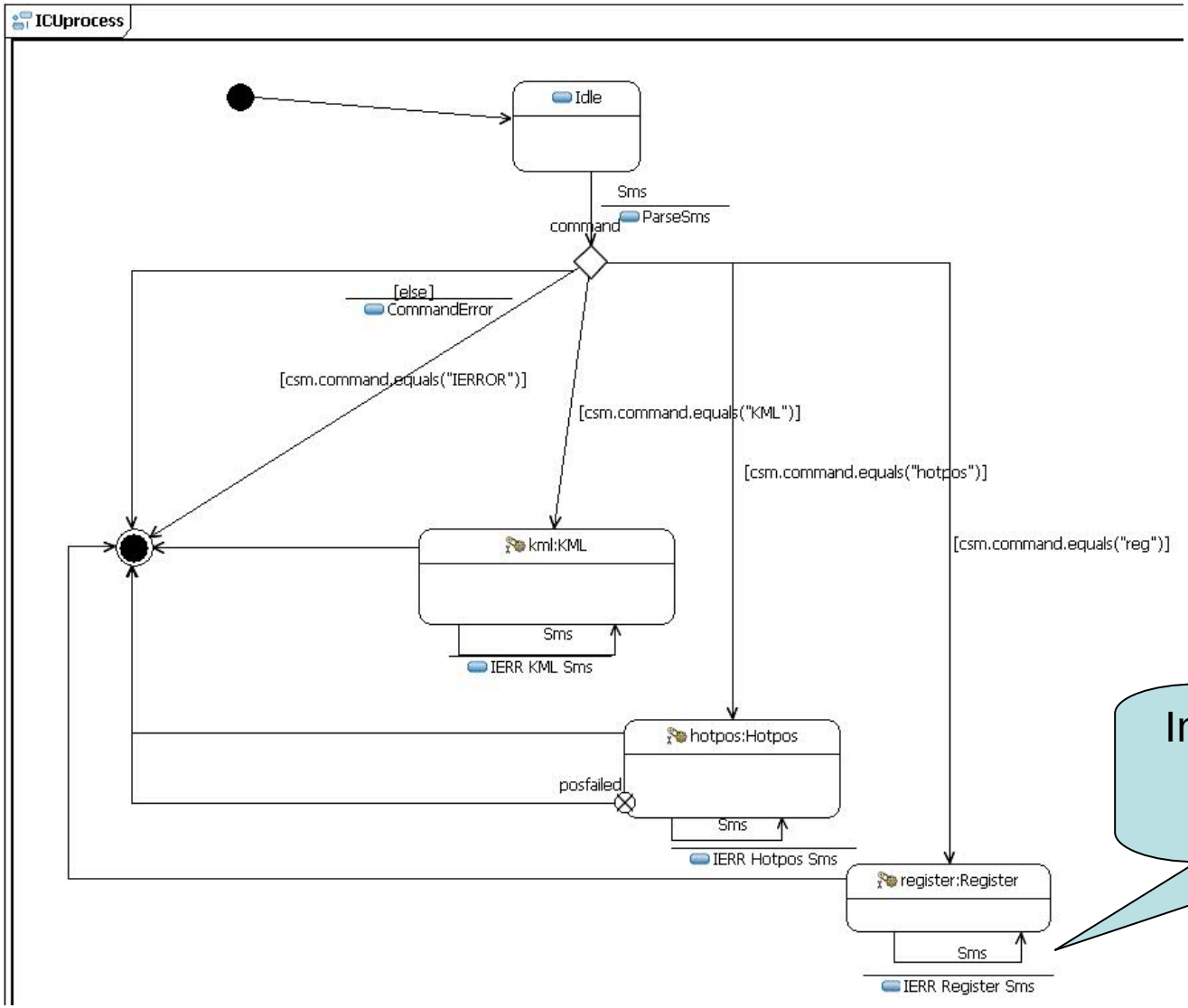
Internal error and no obvious reason to send Sms to user

INF 5150

4&5: Problems best solved on ICUprocess level

- The unexpected Sms signal
  - Neither KML, Hotpos or Register cater for receiving Sms
  - ...but they do not need to since Sms is always handled by ICUcontroller by creating a new ICUprocess
    - true, but will it always be that way?
- Covering the unexpected also makes the software more robust for the future
- The normal situation being that Sms will not occur in ICUprocess it may be handled on its top level

The modified ICUprocess



Internal error and no obvious reason to send Sms to user

INF 5150

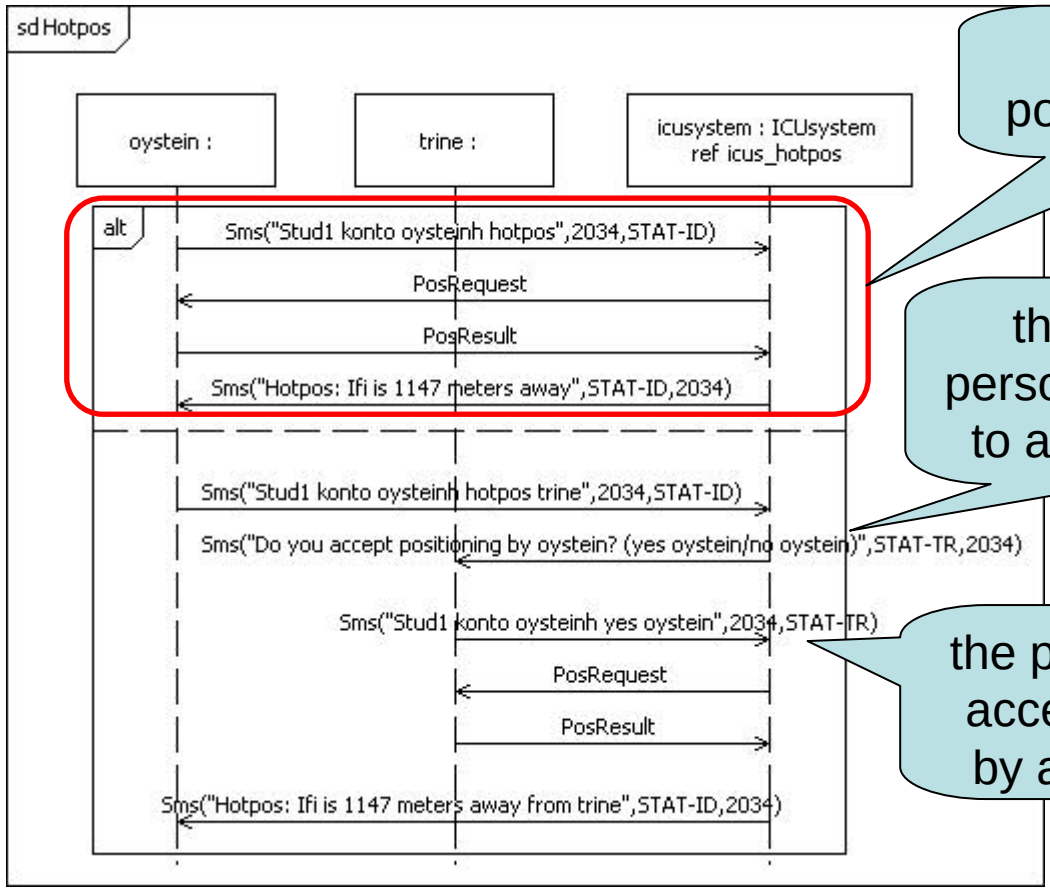
The exceptional

- Data may have strange syntax or values
  - We have looked at data checks for ICUcontroller
- An unexpected signal arrives
  - we explicitly describe every conceivable transition
  - We have looked at this for ICUprocess'KML
- No signal arrives
  - we guard our protocols/services with timers (ICUprocess'KML)
- Security issues
  - authentication + logging + statistics
  - Check for registration in ICUprocess'Hotpos
- Availability issues
  - self tests (we shall improve the Archive)

Services revisited

- Hotpos
  - Only registered users should be able to position others
  - Positioning must be accepted by the positioned user
    - for the sake of showing more advance protocol for authentication
- KML
  - will not get the same full treatment
    - because asking every registered user is too tedious
  - This shows that a ”buddy group” concept probably needs to be introduced to continue to offer KML service
- Register
  - will of course not require that users are registered!

Hotpos revisited



anybody can position themselves

the positioned person will be asked to accept or reject

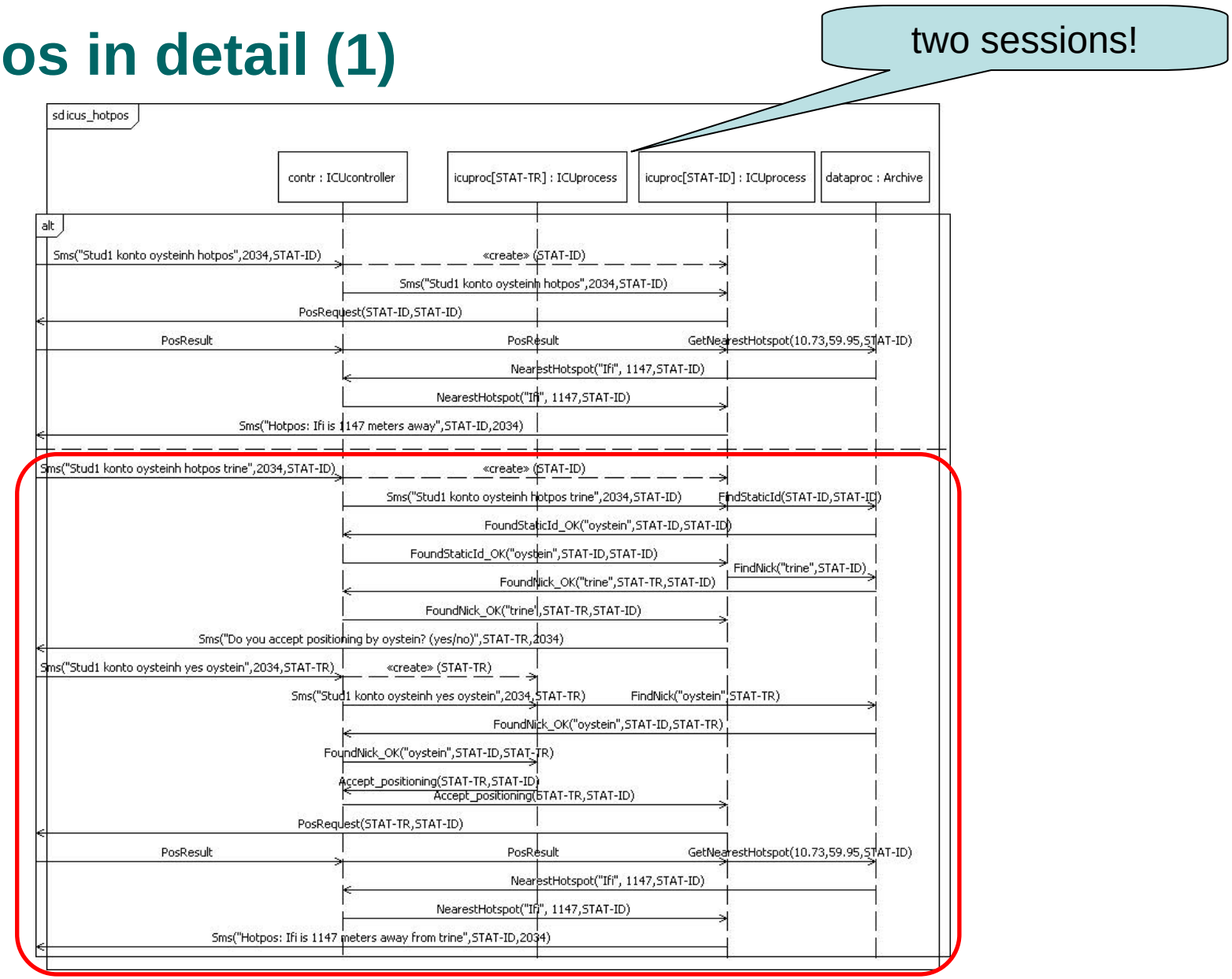
the positioned must accept positioning by a specific user

INF 5150

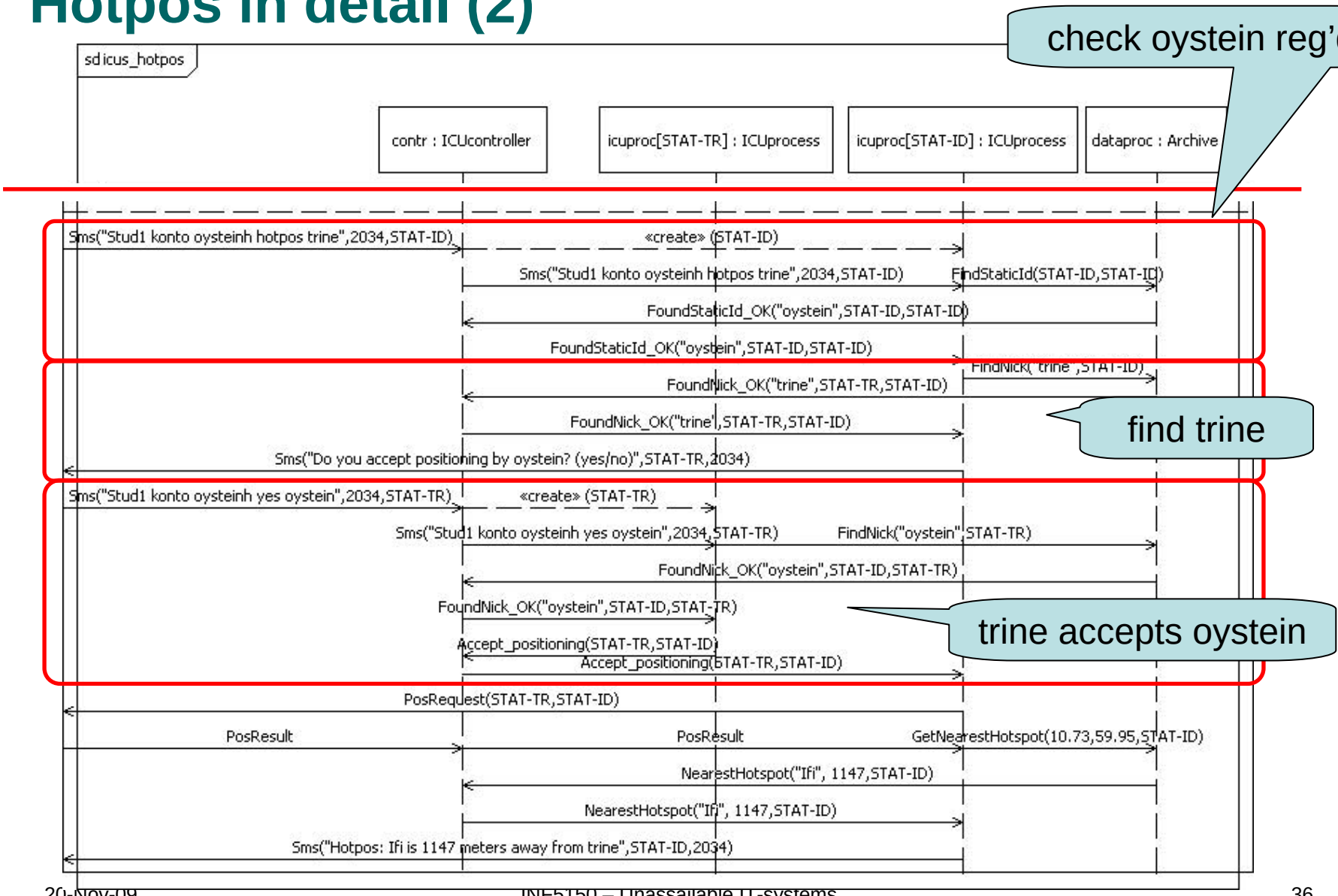
Problems when Trine should accept Oystein

- We need to know that Trine really accepts Oystein and not somebody else
  - we need to connect Trine's response to Oystein's session
- Trine's response is an Sms and that will in our design spawn another session!
  - which may not be a bad idea!
- Let us make a new service – a *yesno* service
  - The yesno service will take an Sms with the following syntax:
    - "yes nickname" or "no nickname"
  - The yesno service will send a signal to the session identified by the nickname
    - Accept_positioning or Reject_positioning depending on yes/no

Hotpos in detail (1)

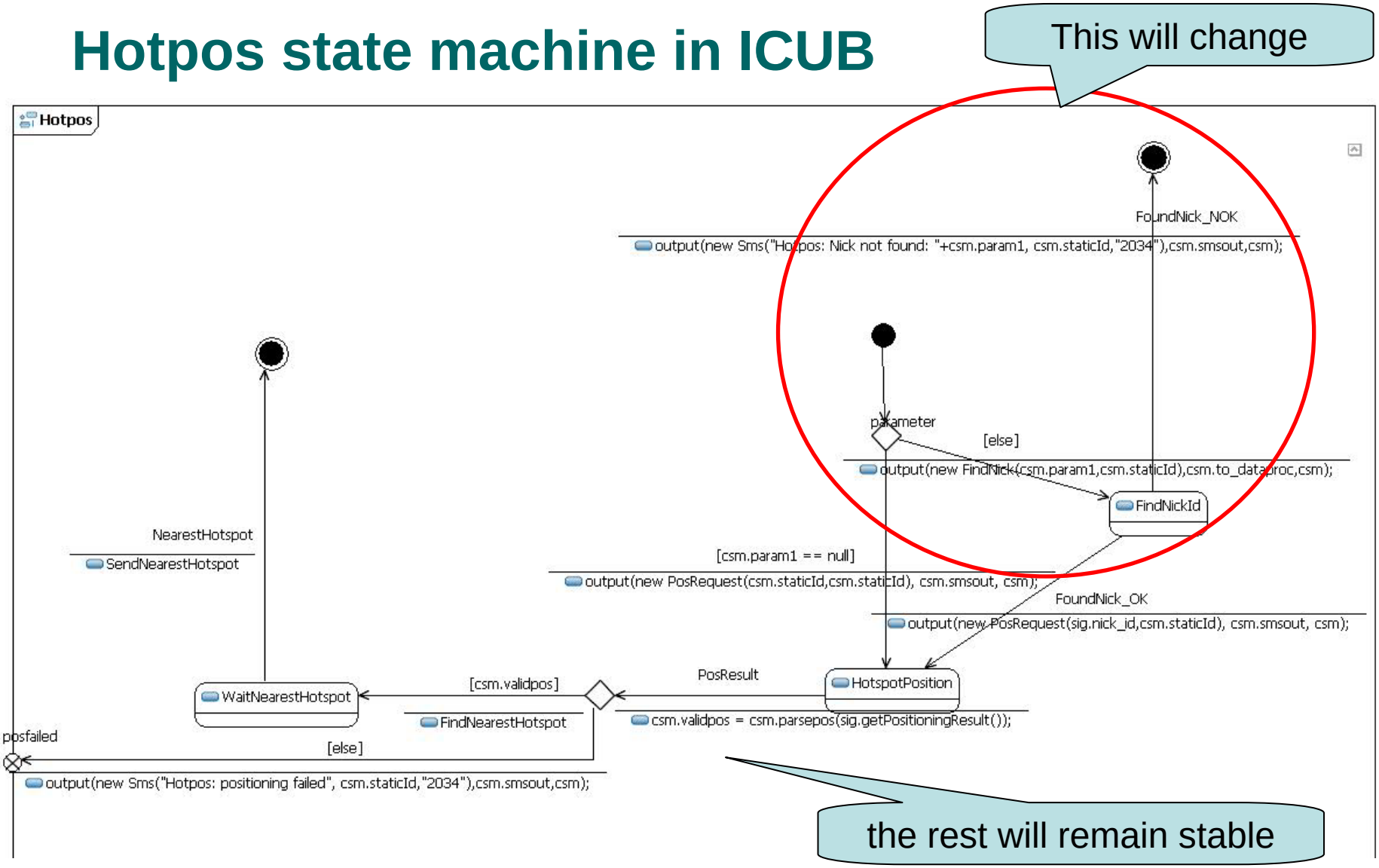


Hotpos in detail (2)

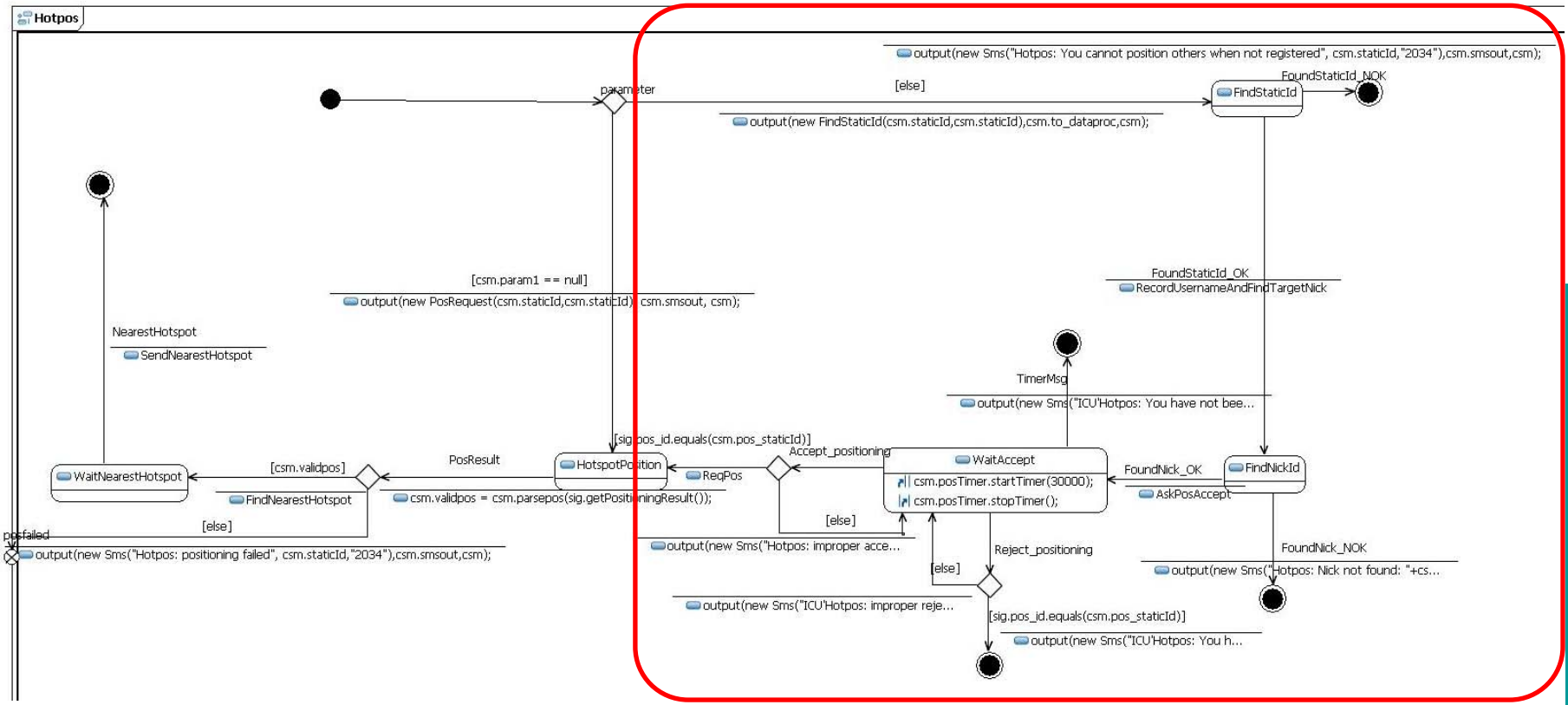


INF 5150

Hotpos state machine in ICUB

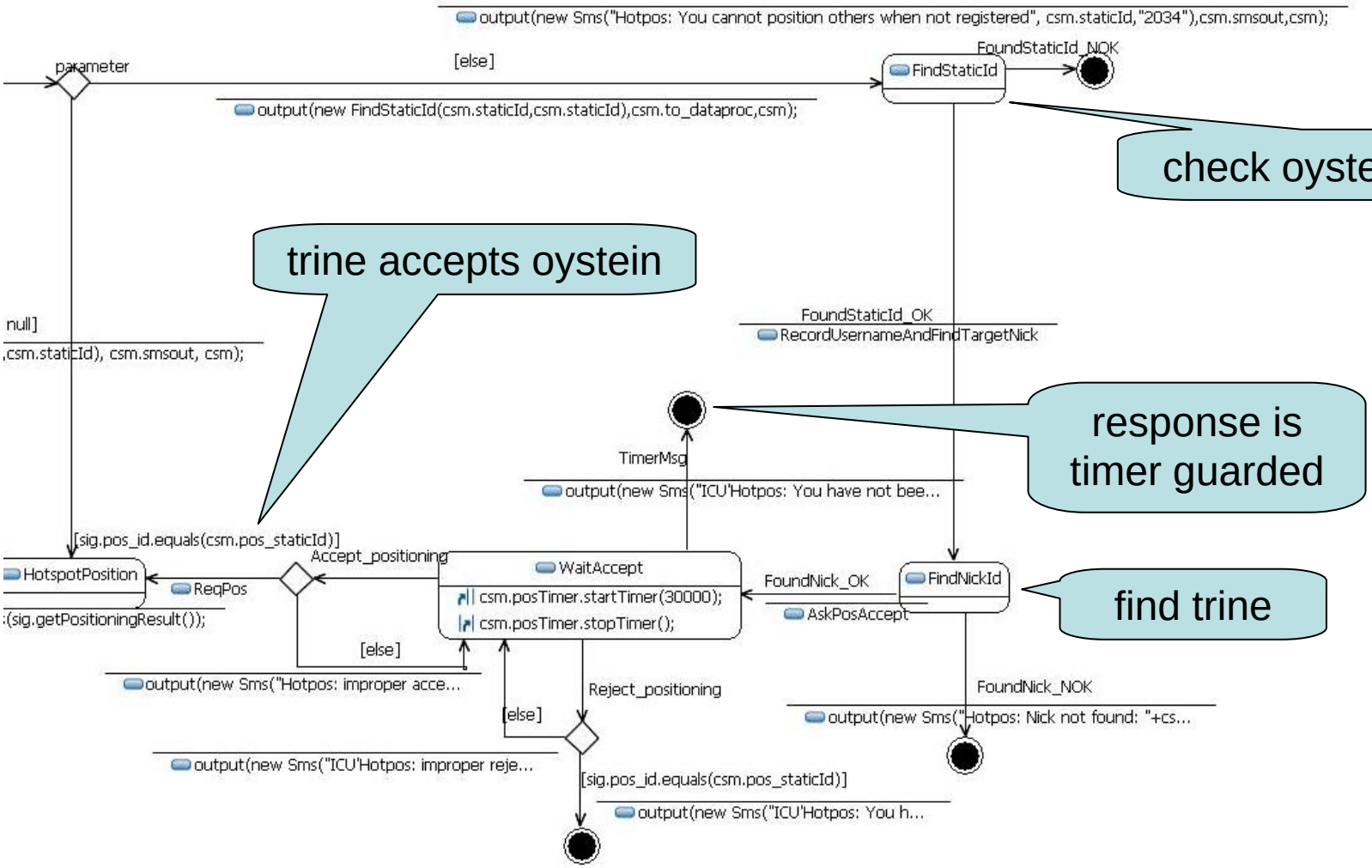


Hotpos in ICUC

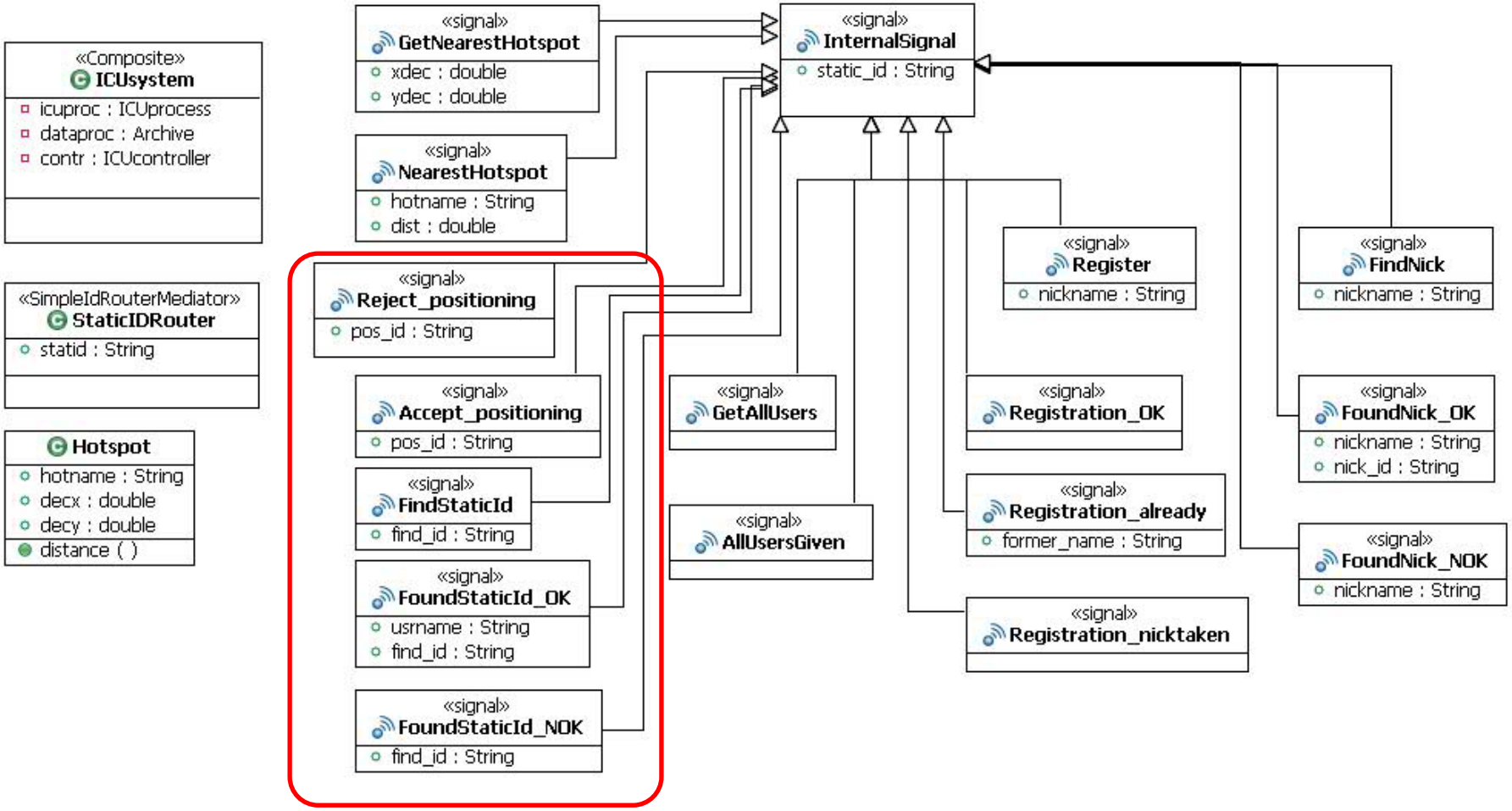


INF 5150

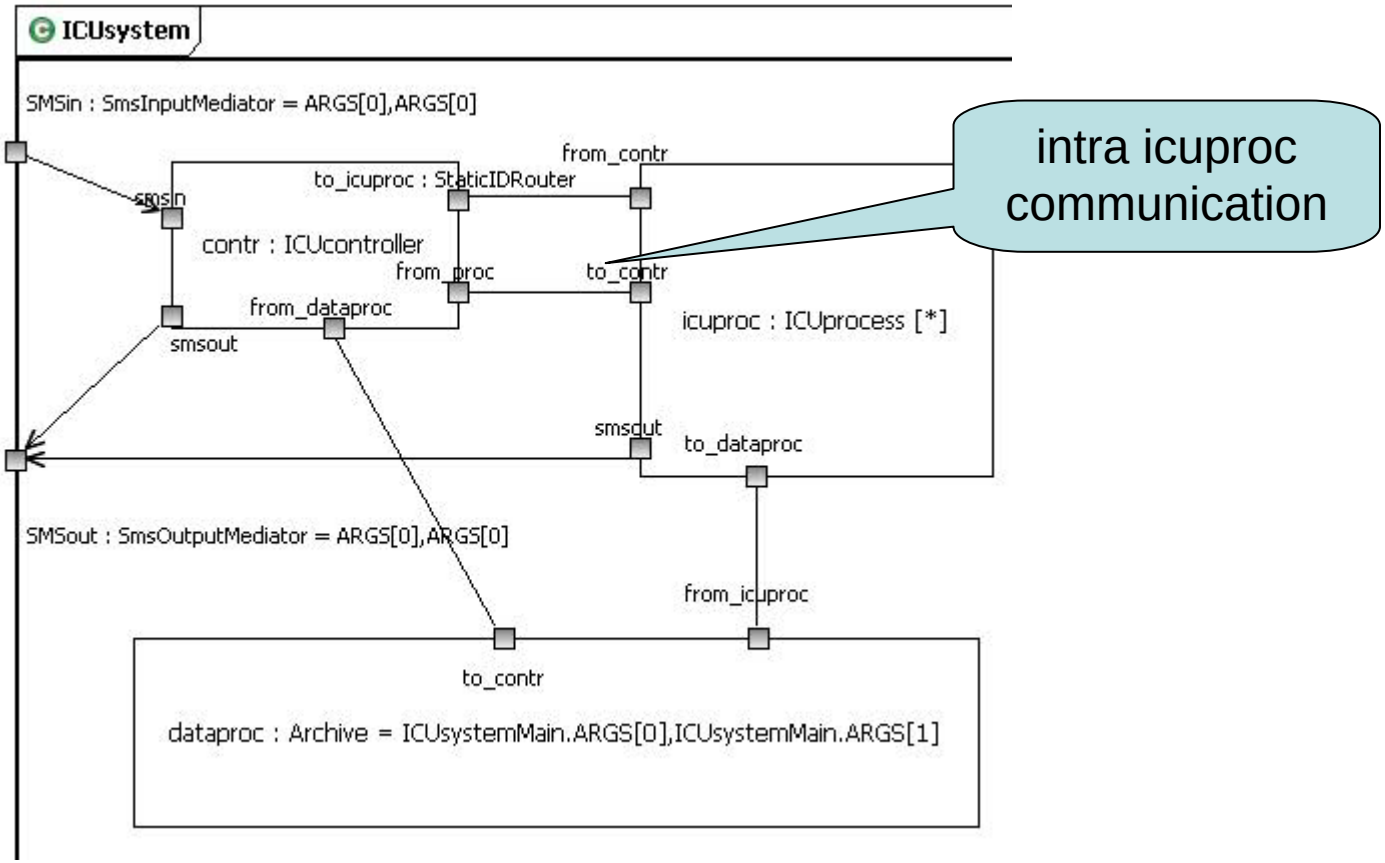
Hotpos – the new features



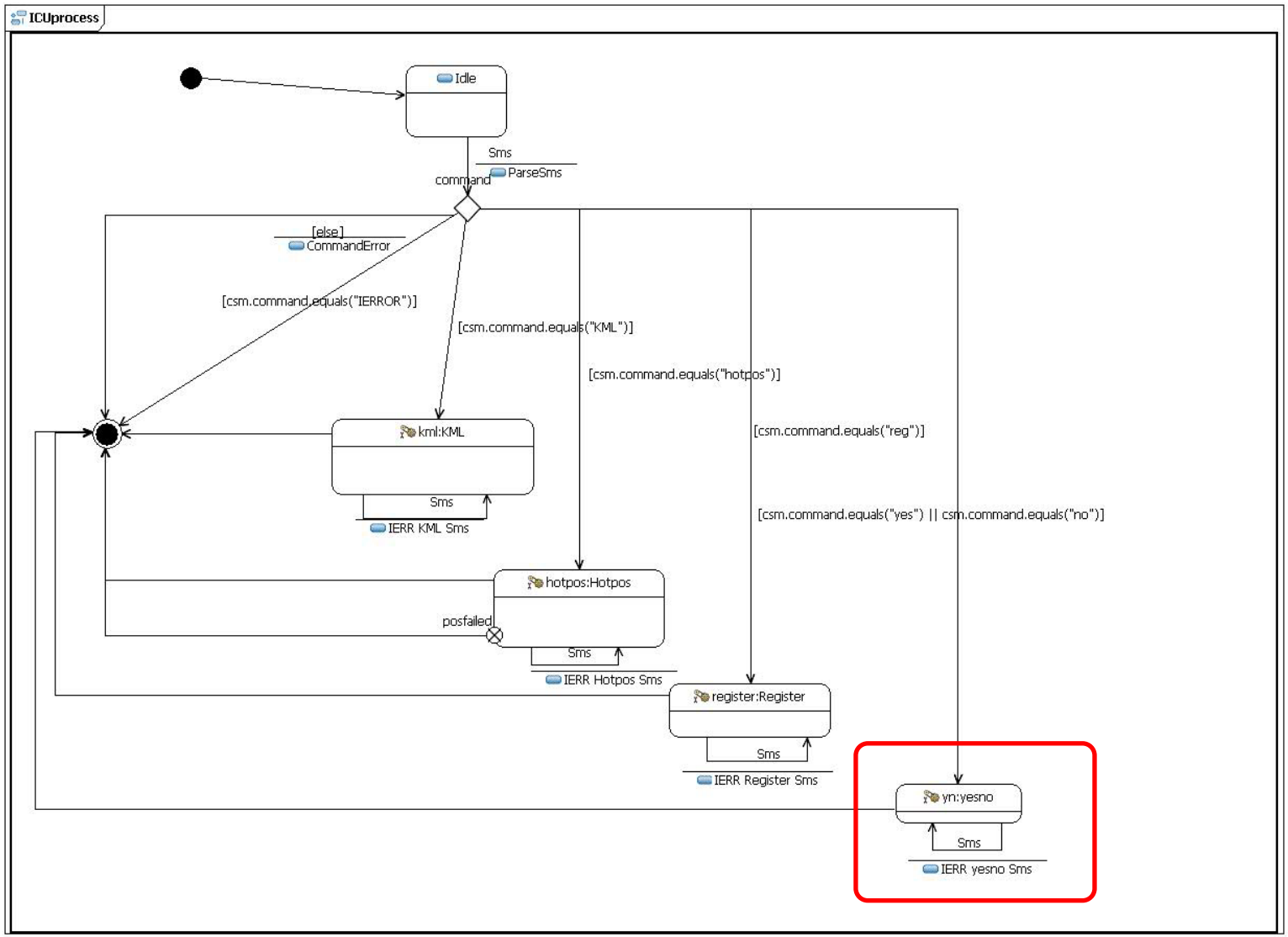
New internal signals



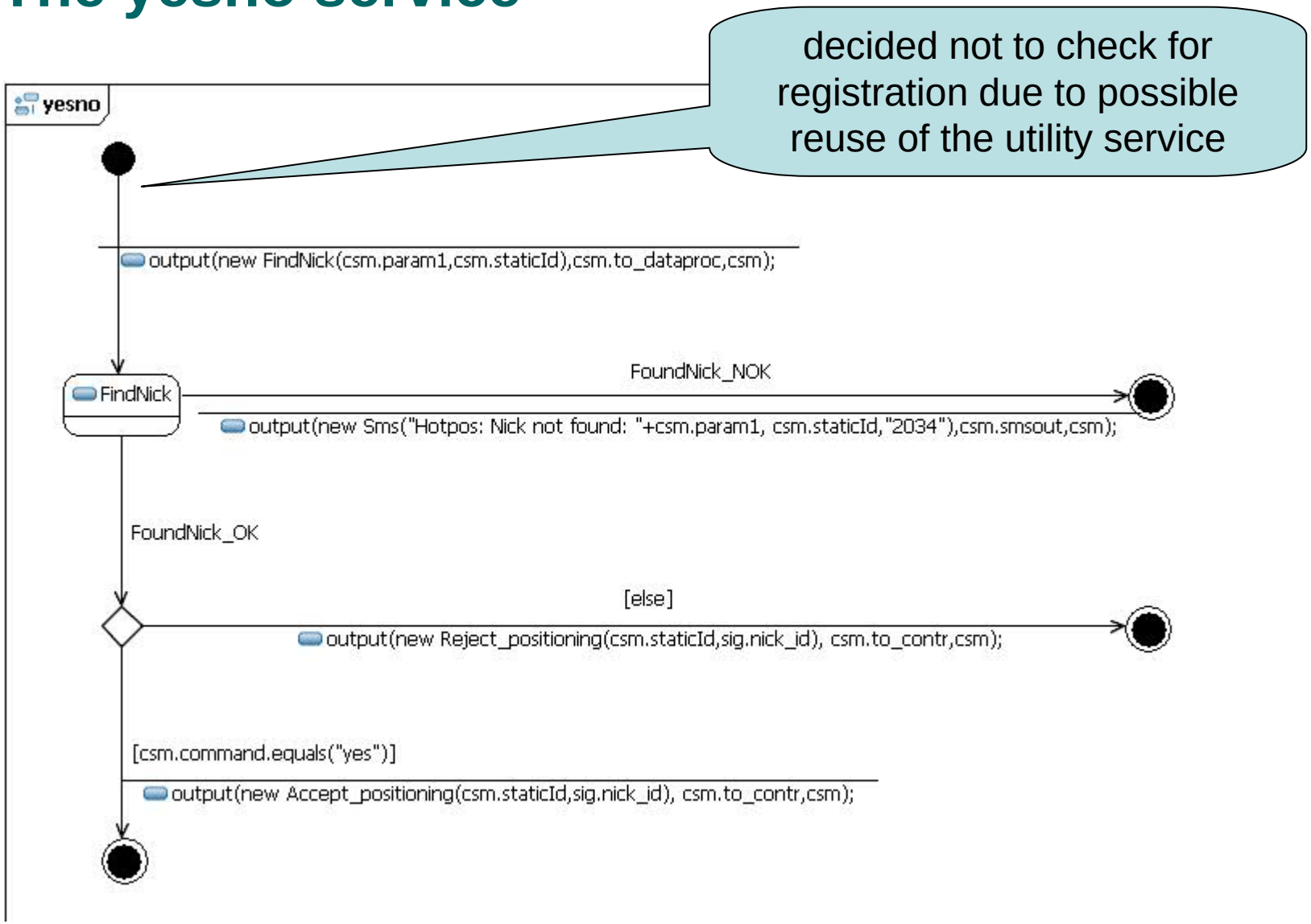
New communication path must be added



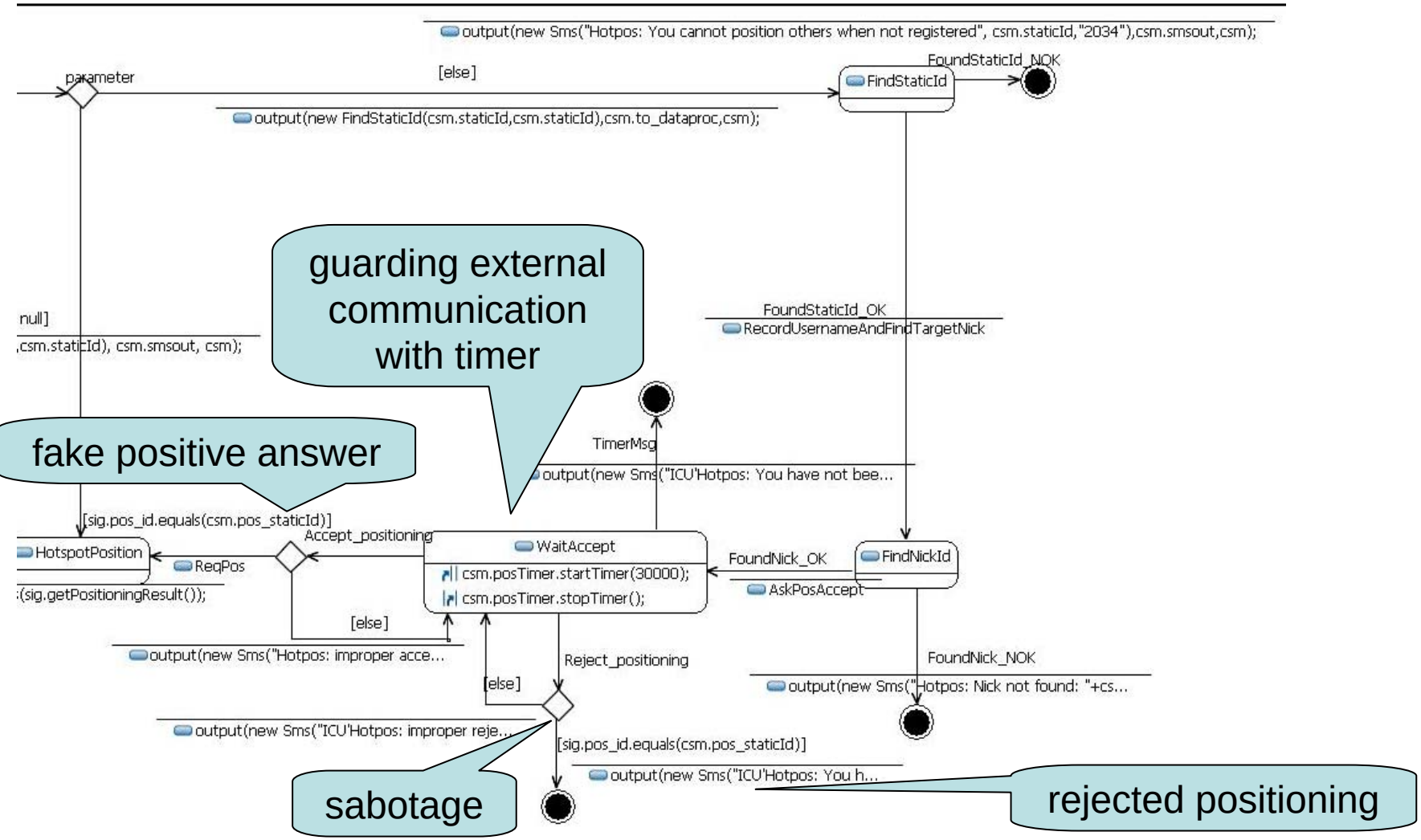
Adding the *yesno* service



The yesno service



Hotpos – more issues



Points to make

- trivial additions to the Archive
  - finding the registered username from static id
- n+1
  - new signals introduced means new signals to cope with everywhere
- stability for parts of the state machine
  - emphasizing that a state is enough to determine the history
- services that use other services
  - Hotpos uses yes-no service
  - therefore we need new connection (and new ports) between *icuproc* and *contr*

More points to make

- Guarding the external communication with a timer
  - WaitAccept where the positioning must be confirmed
- what about yes-no service?
  - out of protocol – we must check on receiving side that the yes-no has the appropriate static id
    - since otherwise anybody (*or even Oystein himself*) could just send a ”yes oystein” in place of the reply from Trine
  - also a reject must be checked against the static id
    - since otherwise anybody could just send a ”no oystein” in place of the reply from Trine!
  - we will not require that yes-no needs registration
    - it is a utility, and may be used more freely at a later stage

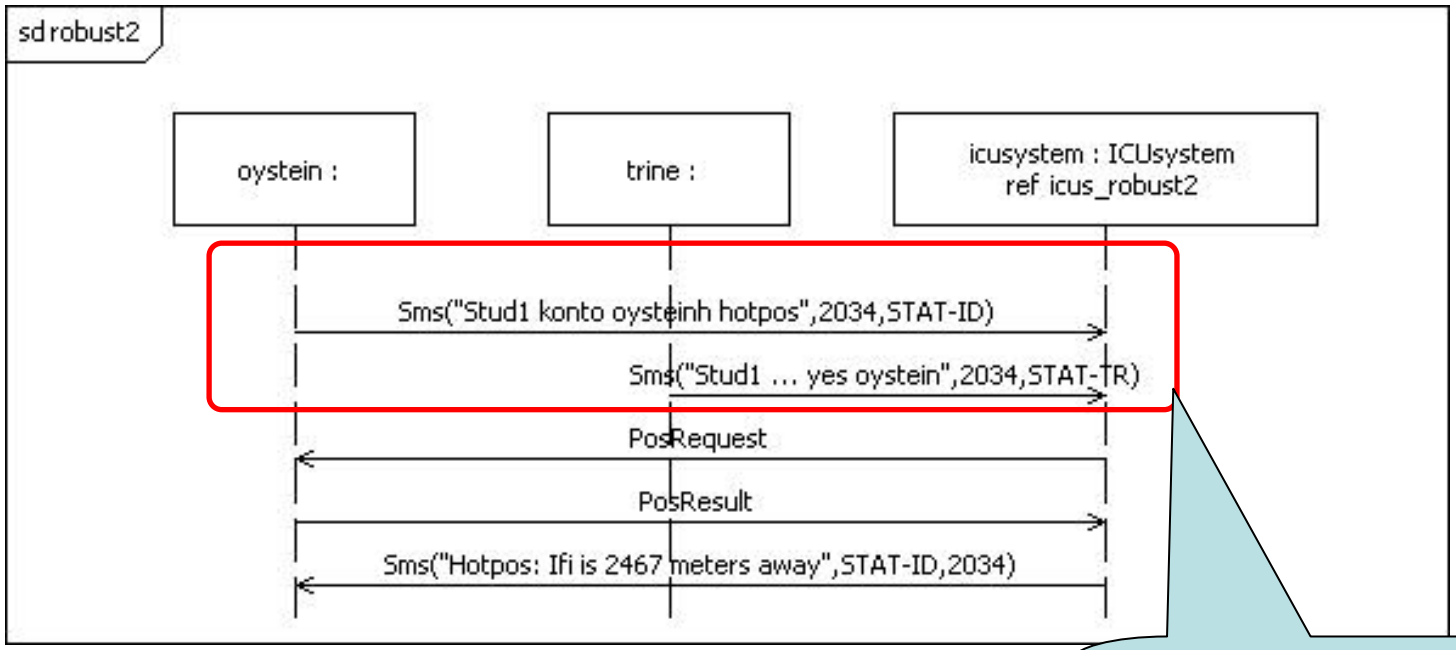
The exceptional applied to new Hotpos

- Data may have strange syntax or values
  - checking static ids of the accept/reject messages
- An unexpected signal arrives
  - We should probably gone through the new signals everywhere
    - especially the accept and reject signals
- No signal arrives
  - we guard our external communication with timers (WaitAccept)
- Security issues
  - authentication (+ logging + statistics)
  - Check that user is registered
  - Check expected static id
    - prevents faked positive acceptance or negative service sabotage
  - Denial of service
    - keep faking will give resetting of the timer

n+1

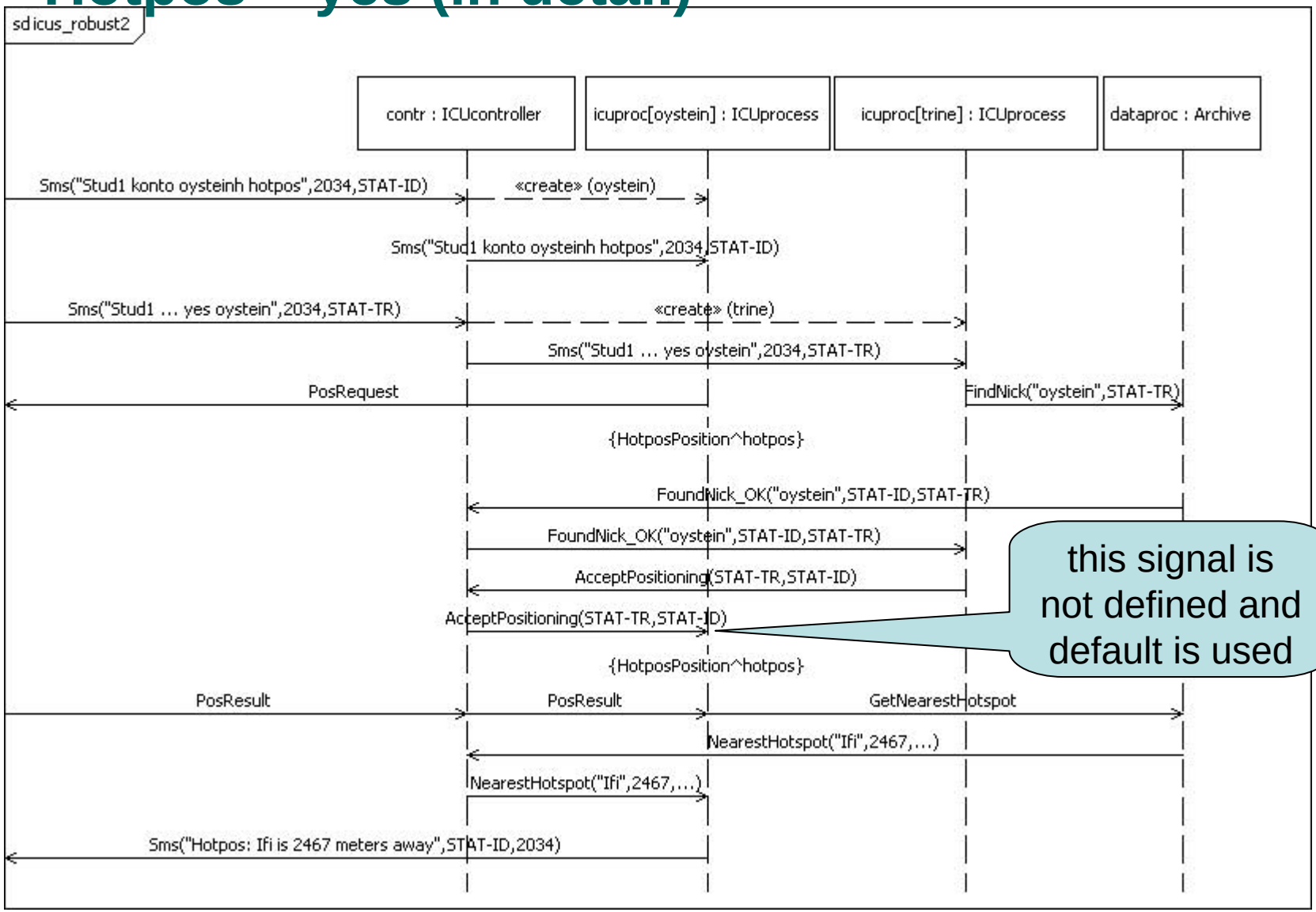
- When we add functionality, we add signals
  - and those added signals should be covered in all states
  - in ICUC this has not been done yet!
- We have added external legal services *yes* and *no*
  - These services may produce internal signals *Accept_positioning* or *Reject_positioning* to other *ICUprocesses*
  - Those services may not be ready for those inputs!
    - if yes/no has been sent for no purpose or the nickname is misspelled
      - and the misspelled person really has a service going (rather improbable)

Hotpos + yes (resulting in a default transition)

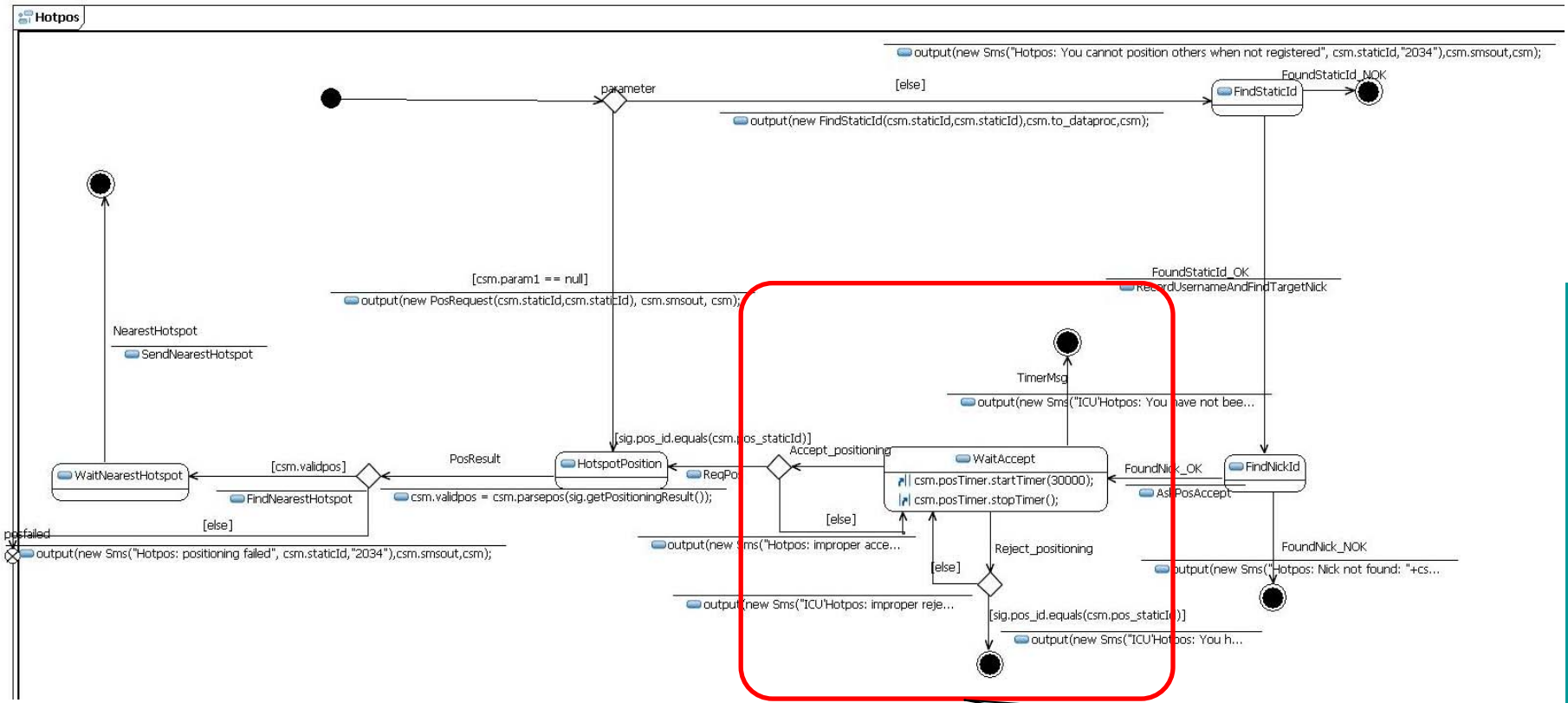


competing initiatives and yes is really wrong

Hotpos + yes (in detail)

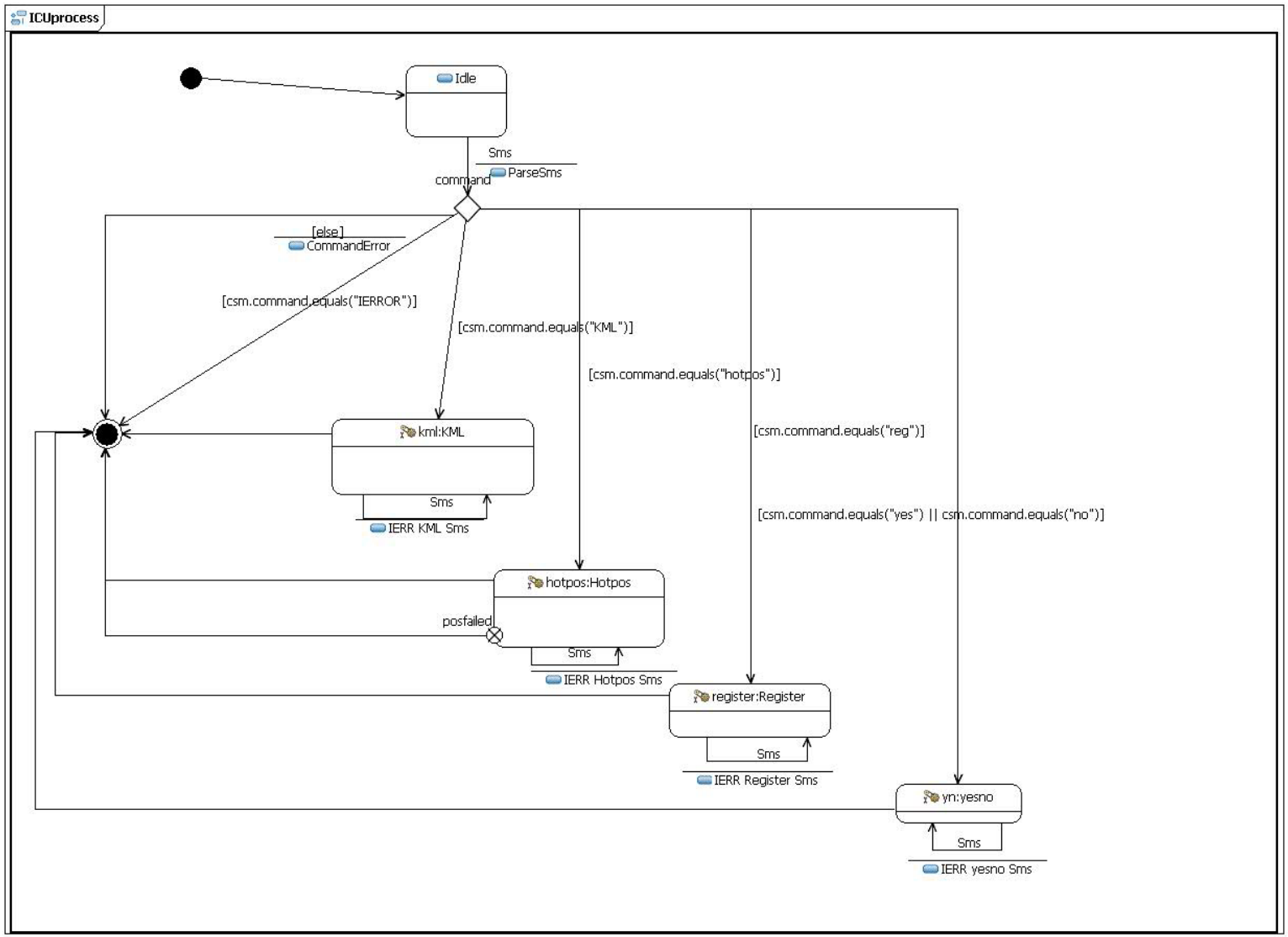


Misplaced Internal Signals



In Hotpos we only expect Accept_positioning or Reject_positioning here

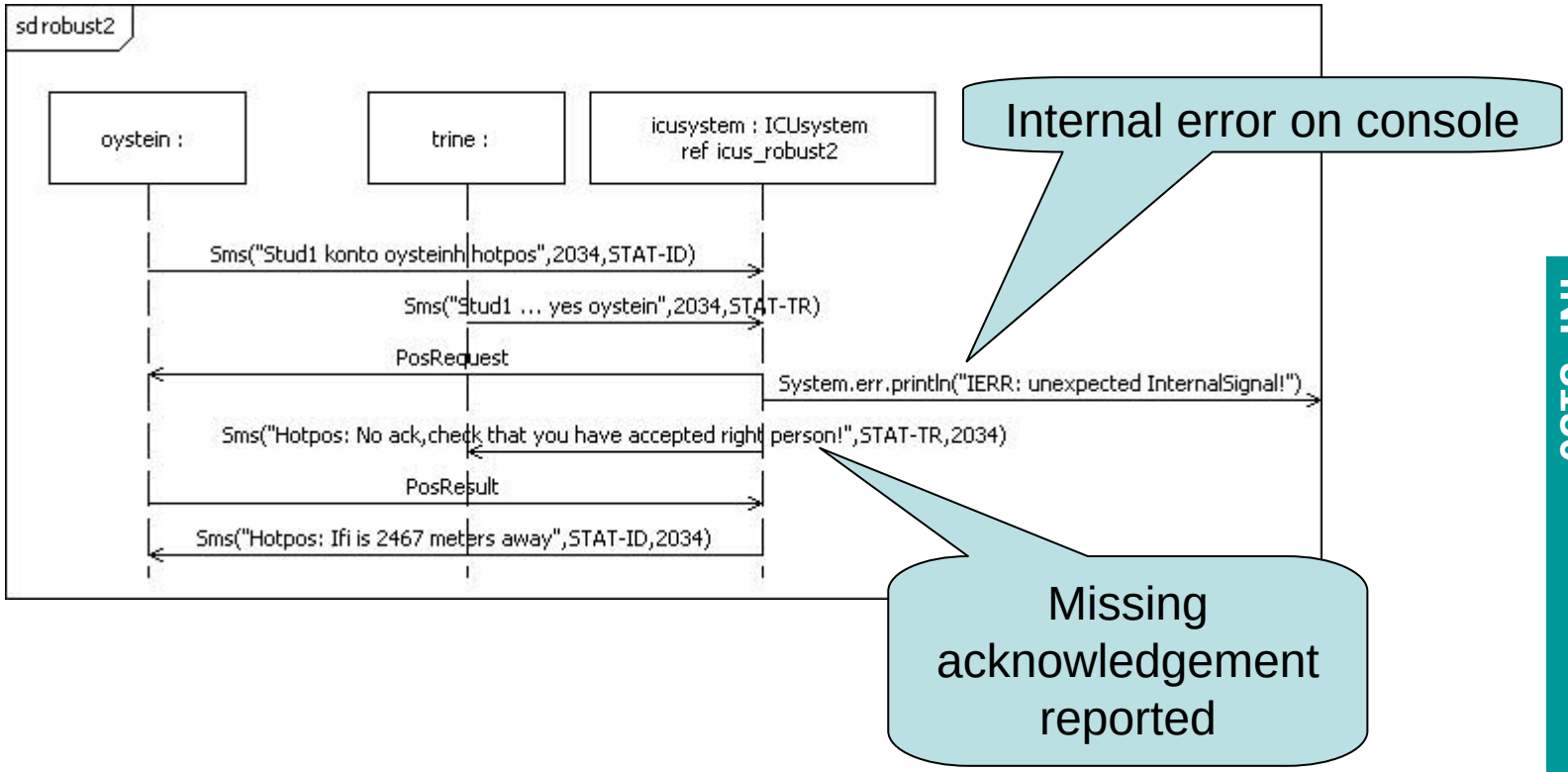
No capture of InternalSignal on top level



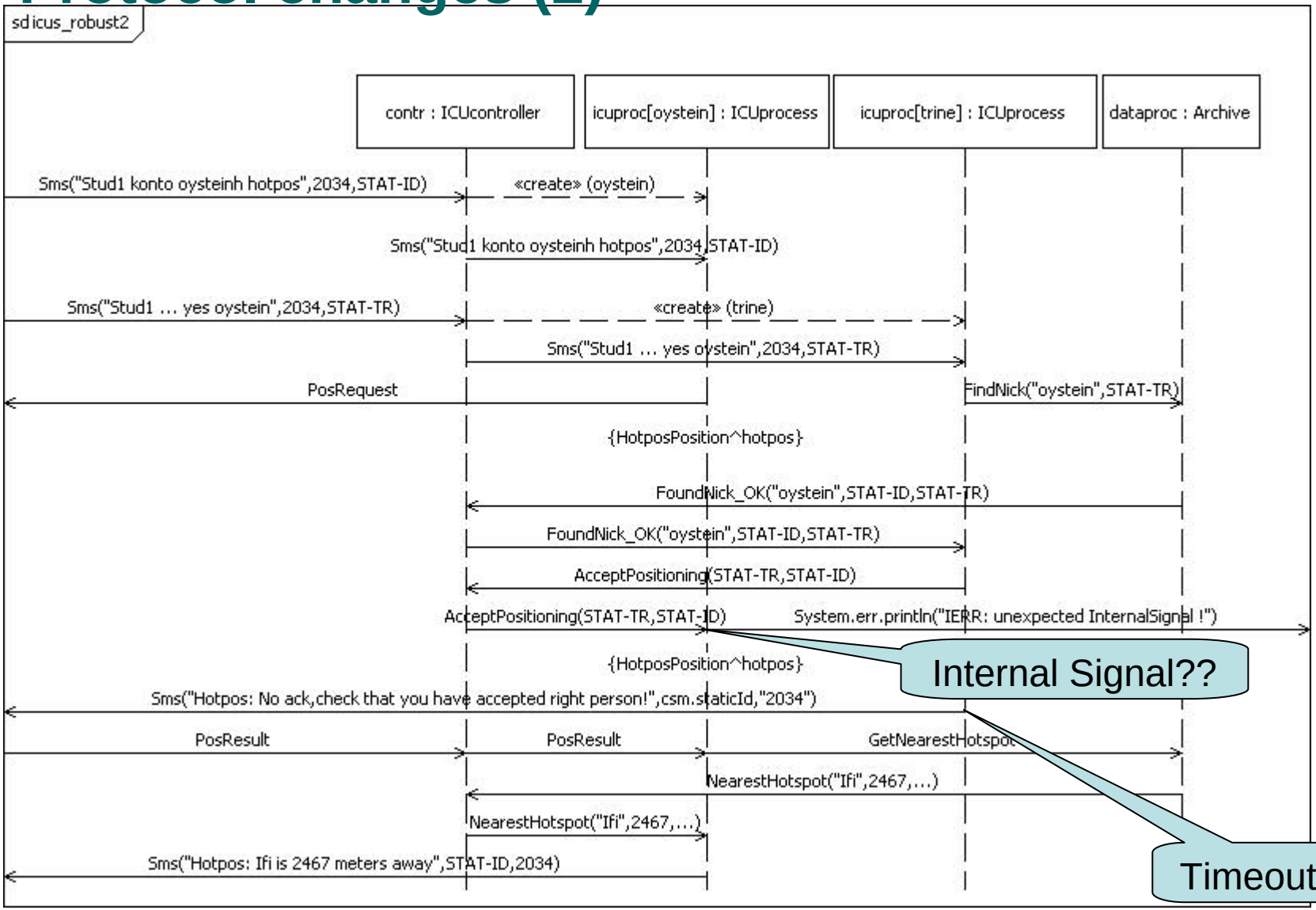
Using the yes/no service

- Normal (sd ICUC'Hotpos):
  - oystein asks "hotpos trine"
  - trine accepts (or rejects) by saying "yes oystein" or "no oystein"
- Exceptional 1(sd ICUC'robust2):
  - oystein positions himself by "hotpos"
  - trine for no reason concurrently says "yes oystein"
- Exceptional 2:
  - oyvind asks "hotpos trine"
  - trine misreads oyvind's nick and says "yes oystein"
    - trine gets no message that her supposed acceptance fails!
    - oyvind will time out waiting for trine's approval
  - Possibly we should need double acknowledgment protocol
    - trine should be confirmed that her acceptance succeeded?!
    - or she should get an error message back when not acknowledged

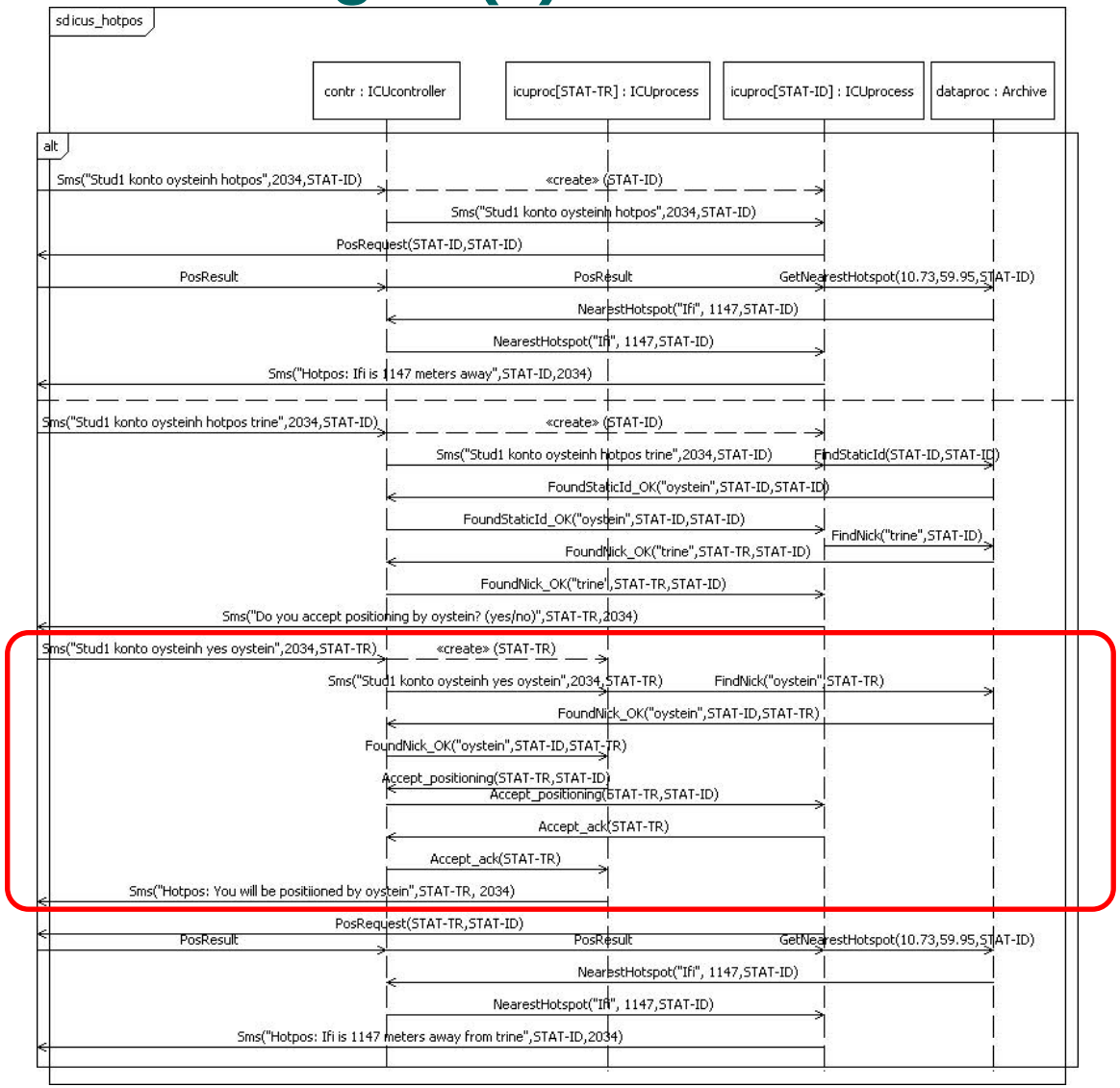
Protocol changes in detail (sequence diagrams)



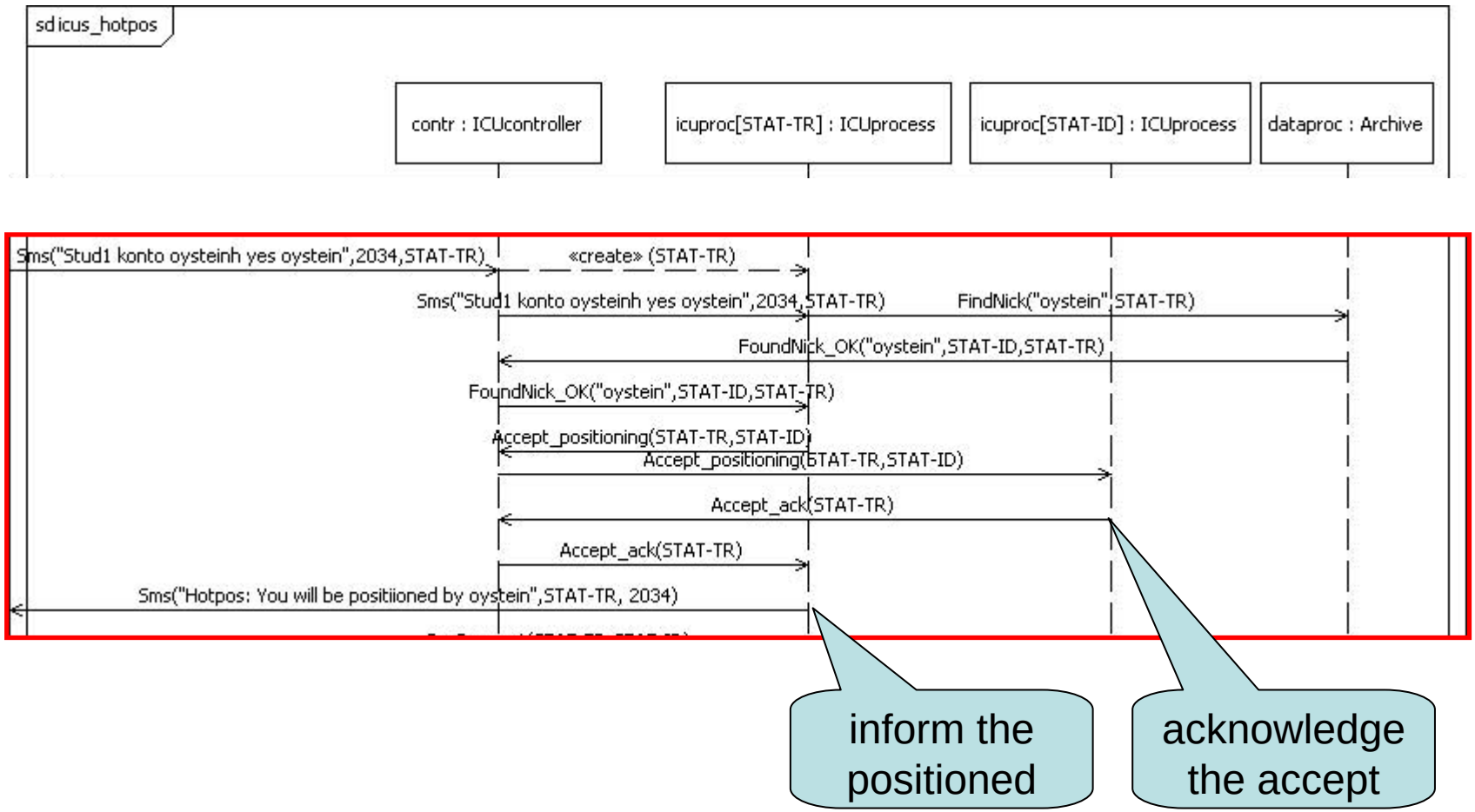
Protocol changes (2)



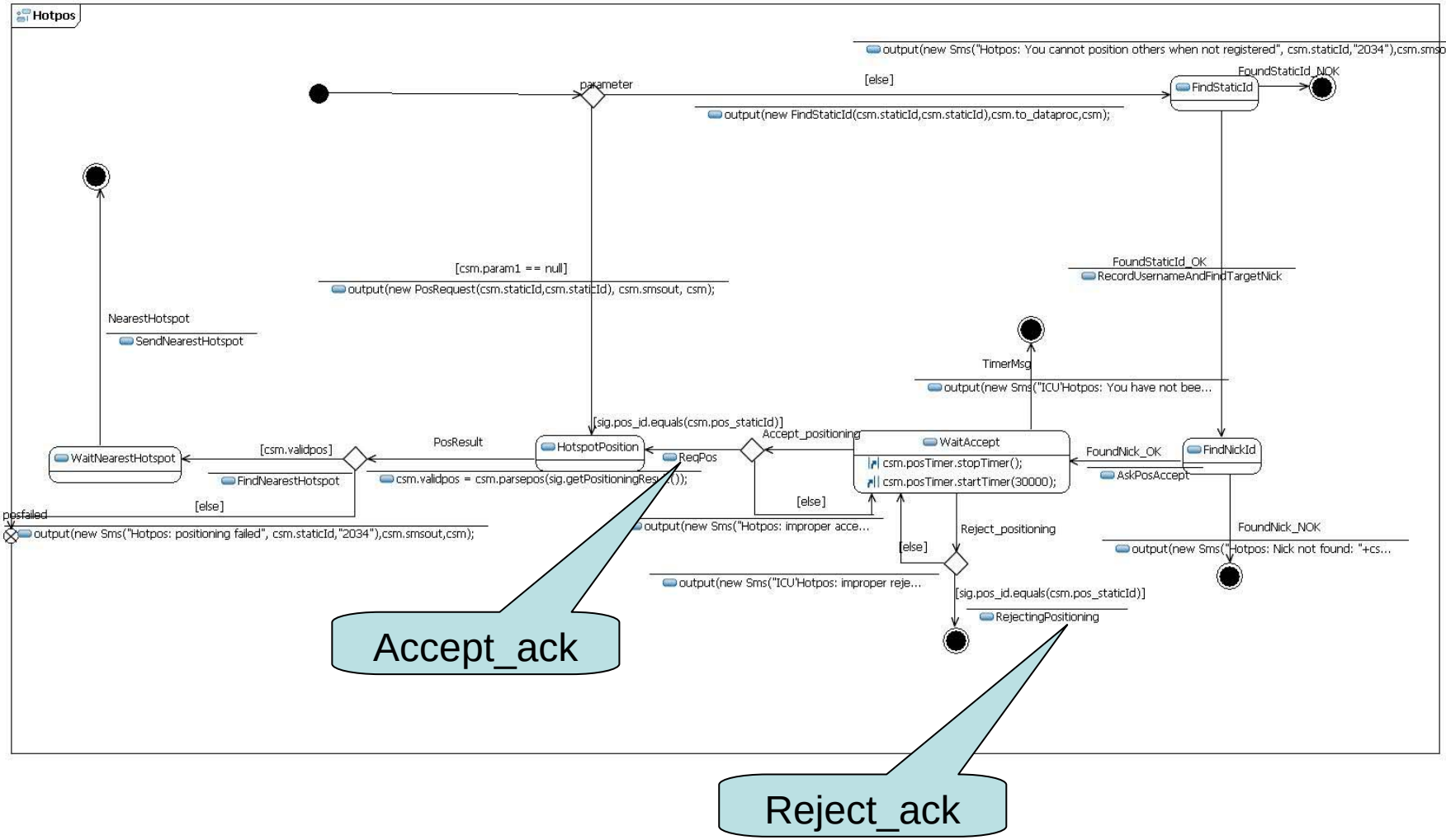
Protocol changes (3) – the normal Hotpos



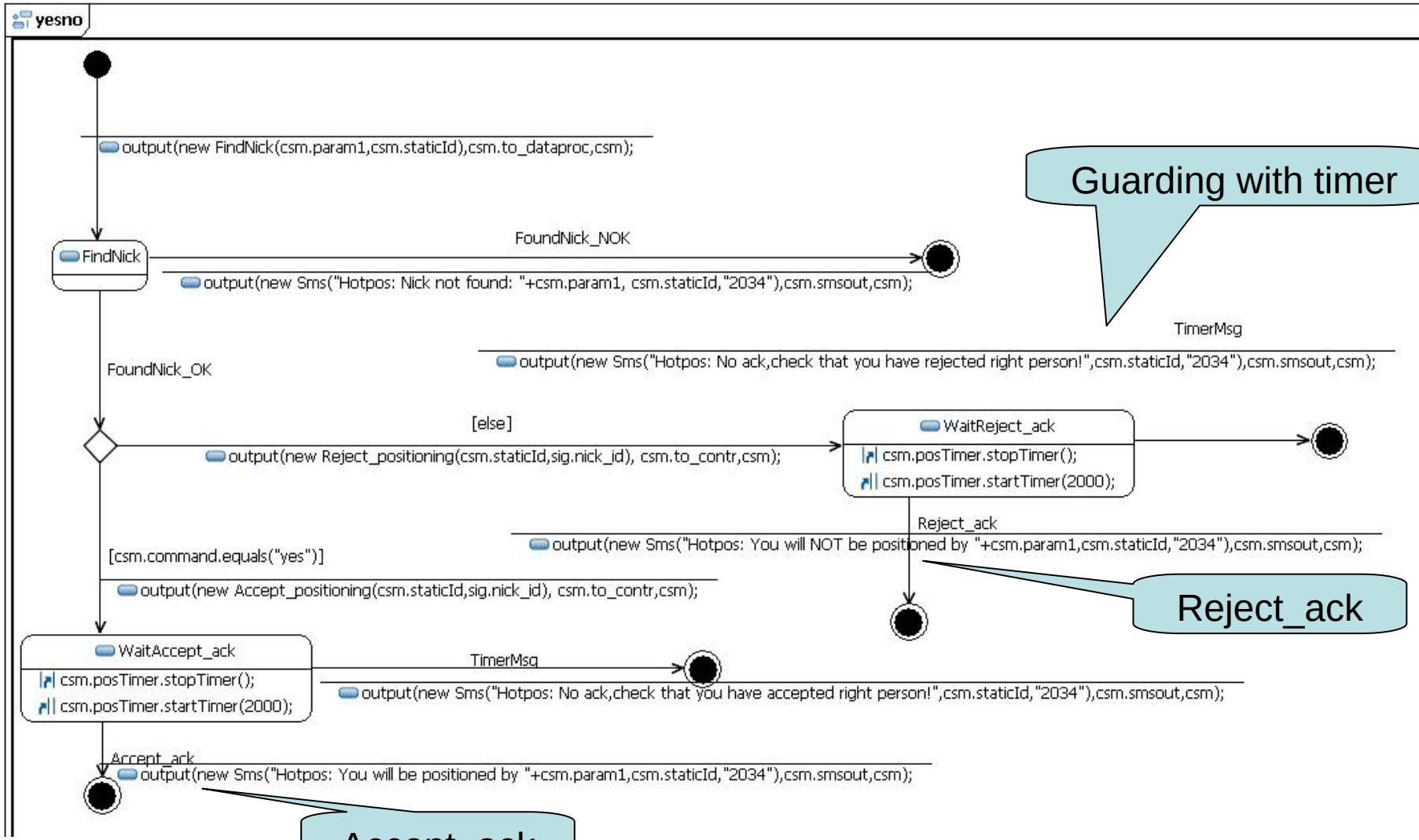
Protocol changes (4) – the big view



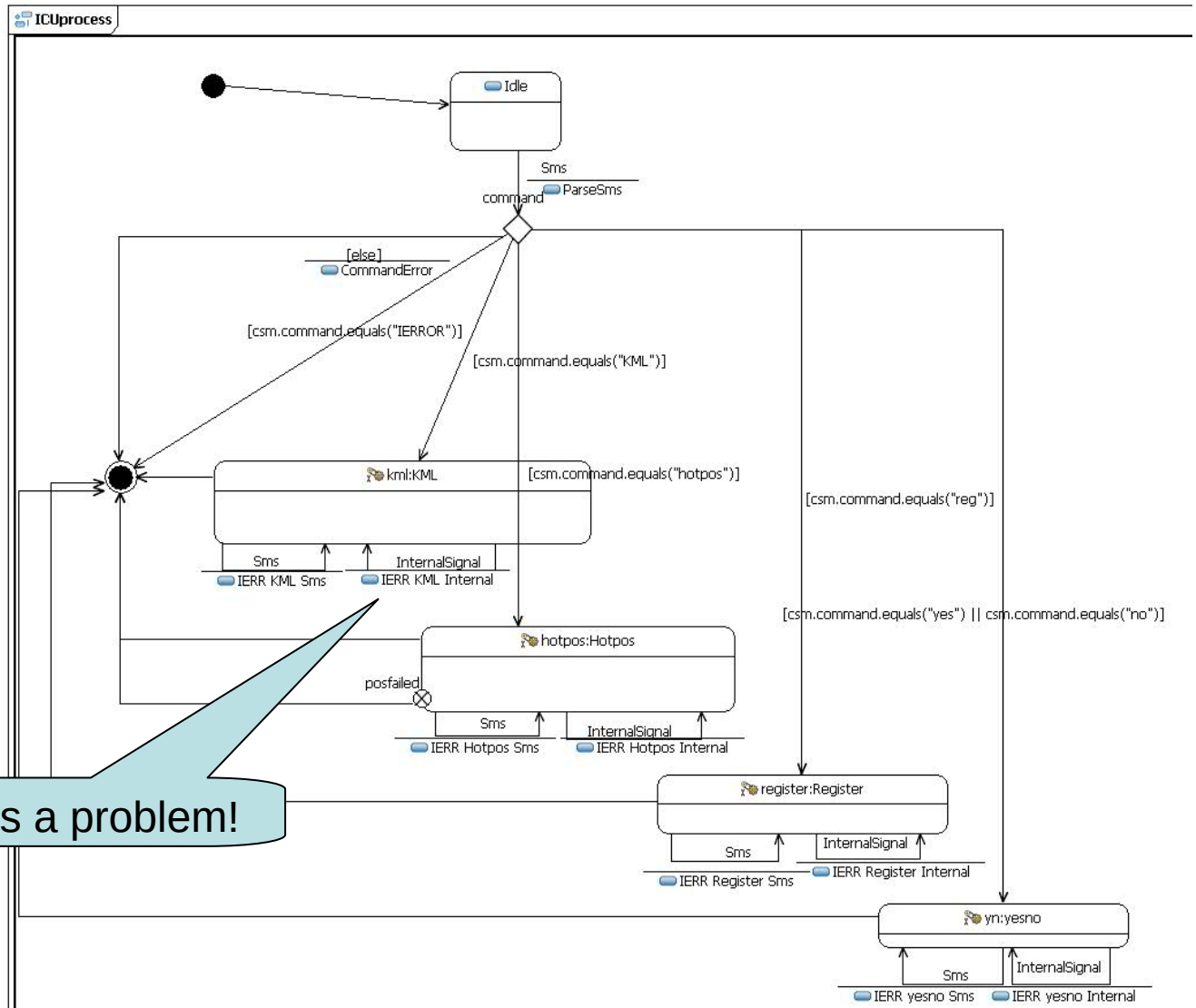
Implementing the new protocol (Hotpos)



Implementing the new protocol (yesno)



Implementing the new protocol (ICUprocess)



There is a problem!

INF 5150

A story about history

- Unexpected signals can be caught on outer levels
  - but we want the net effect to be ignoring them after giving an error message
    - This can be done in UML with History states
- History states assures that when returning through a history point into a submachine state, execution will return where it left off in there
  - there are shallow and deep histories (one level, or all levels)
- UML has history states, but JavaFrame has not!
  - 1. implement history in JavaFrame? (not done yet)
  - 2. let transition return into the state anyway? (will restart the state)
  - 3. let transition end in a final state (terminating service which means that there is a way to perform denial of service)
  - 4. flatten the outer level error transitions into the inner levels

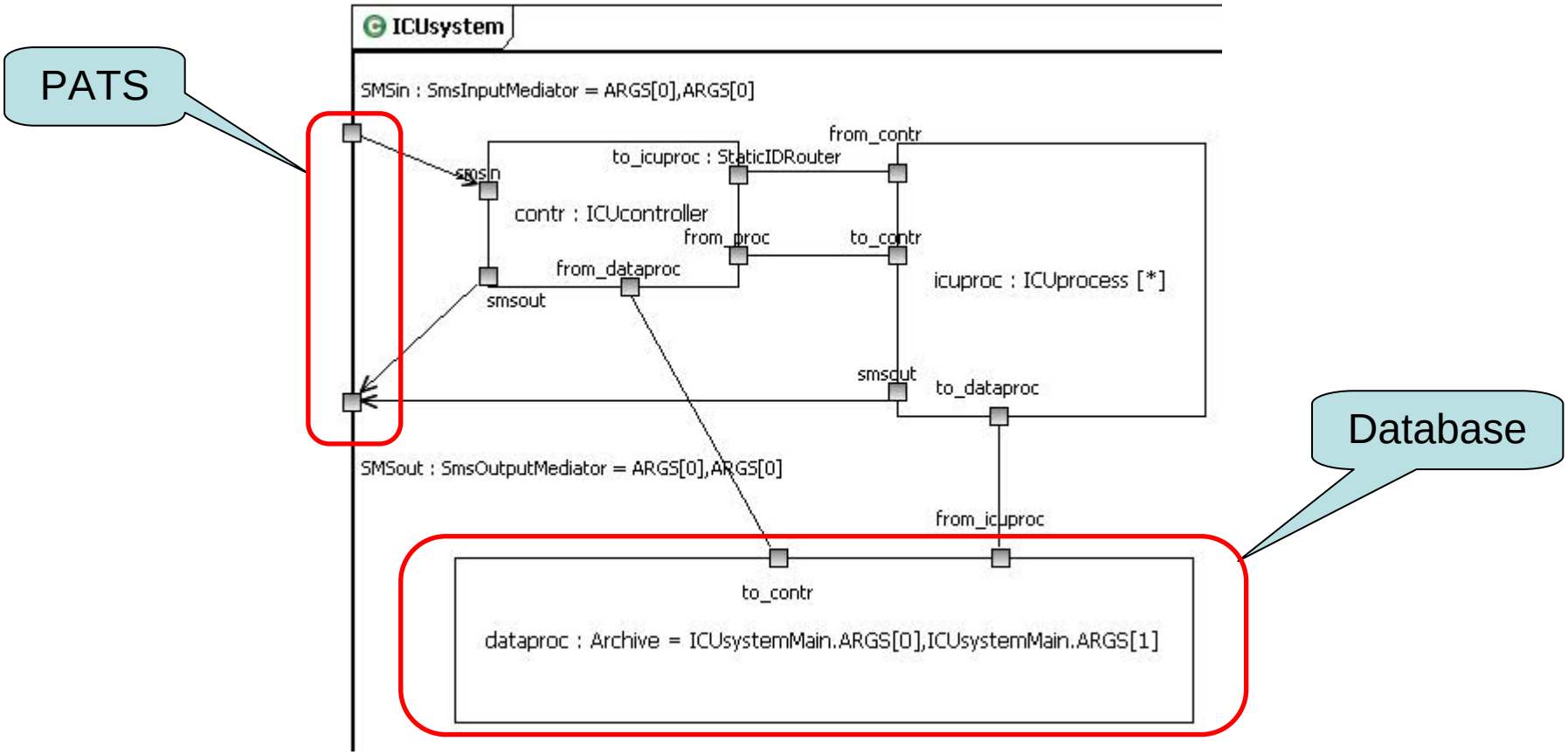
A lesson learned

- What you have not checked, may not work
  - We did not manage to check the Sms errors and therefore did not manage to discover the history problem
- What is defined in a standard, may not be implemented
  - History states are found in UML 2, but are not implemented in JavaFrame
- It is not always obvious what is the optimal solution
  - 1. Implementing History states in JavaFrame
    - good for the future, but time-consuming now
  - 2. Restart the state
    - will also restart the service and that is not in general attractive
  - 3. Terminate
    - simple solution that actually hurts an innocent user
  - 4. Flatten the transitions down
    - not very elegant, but requires only finite time to do
    - not very future-oriented

Availability

- Availability
  - That authorized users can get the services they want when they want them
- It may be too late to check the availability when the service is being asked for
  - It may be necessary to check regularly regardless of demand
- External resources upon which the service depends
  - should be checked regularly
- Internal resources
  - may be trusted as they may only be divisions of the program
  - may be checked if they involve external resource (like network)

External resources of the ICU system



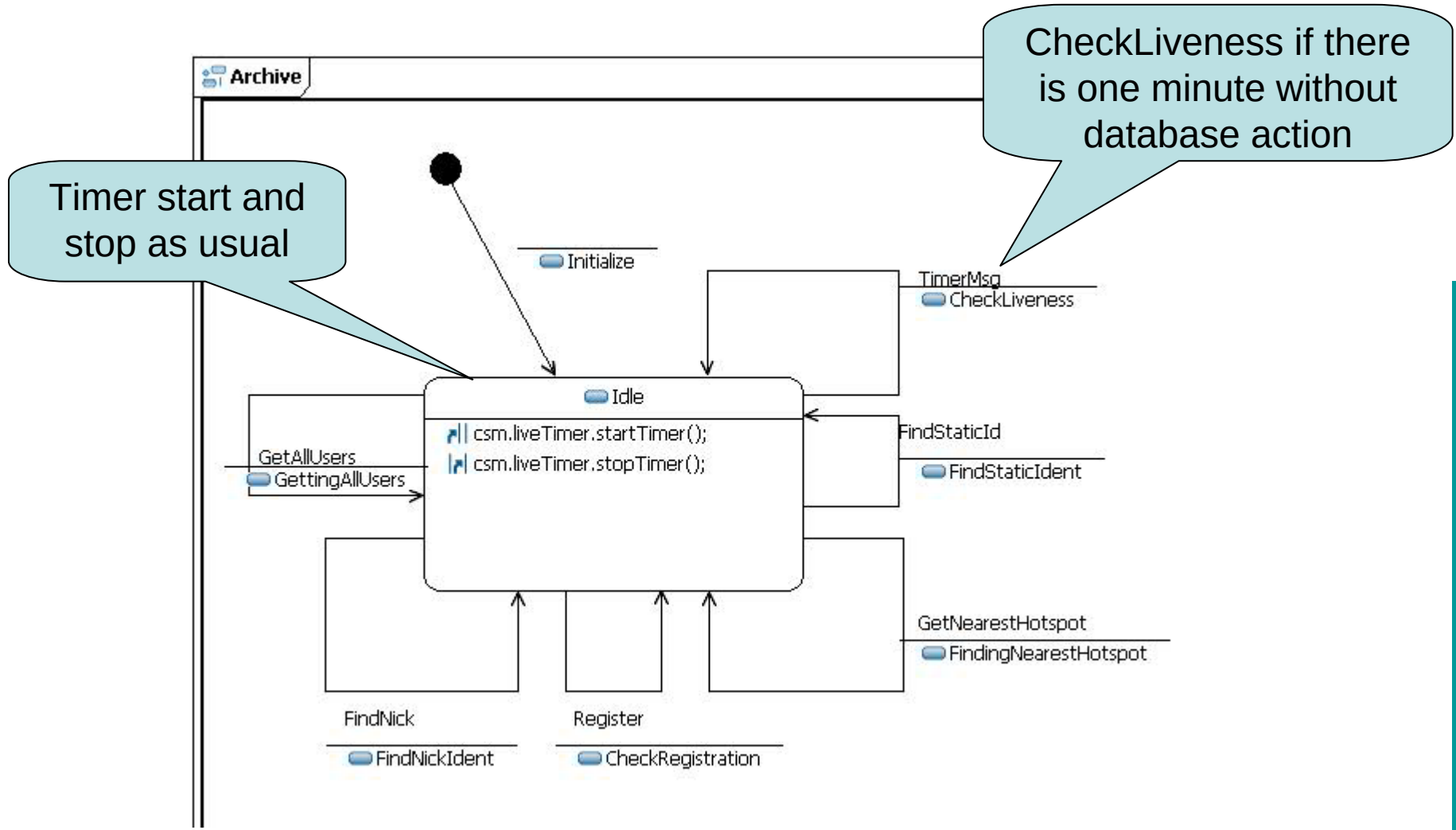
PATS

- The connection to PATS is controlled by the IFI lower level software
  - This is not always enough to make sure that PATS really works the way it is expected for our purposes
- In a normal situation there will be frequent requests to PATS and malfunction would be reported through the robustness means that we have already applied
  - If PATS connection is dead, nothing would reach our program
- Extra liveness checks would actually cost money (for commercial utilization of PATS)
- For ICU we decide not to introduce extra liveness checks against PATS

IFIORA – the IFI Oracle database server

- IFIORA will also be invoked frequently and failure reported through the exception handling
  - which should be improved from stack dump!
- For the sake of demonstration we also include a liveness check for IFIORA
  - We assume that the exception handling implicit in jdbc will always capture availability exceptions
  - An extra liveness check will be implemented through a regular timer-driven transition that performs a simple SQL-command
  - An availability exception will be reported back to the calling service through a special internal error signal (DataError)
    - on which the service may react by issuing a message to the user
- Many small cascading effects around in the model

Archive – with added liveness timer



CheckLiveness

```
/* Liveness check by performing the simplest kind of SQL command */
try {
        Statement stmt = csm.con.createStatement();
        String theQuery = "SELECT COUNT(*) FROM gsmuser";
        ResultSet r = stmt.executeQuery(theQuery);
} catch (Exception e) {
    System.err.println("ICU'Archive: Liveness check fails! Reconnecting!");
    try {
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Properties props = new Properties();

        props.put("user", csm.oracleAccount);
        props.put("password", csm.oraclePasswd);

        String url = "jdbc.:oracle:thin:@delphinium.ifi.uio.no:1521:IFIORA";
        csm.con = DriverManager.getConnection(url, props);
     } catch (SQLException ee) {
        System.err.println("ICU'Archive: Error when reconnecting!");
    }
}
```

cheap SQL statement

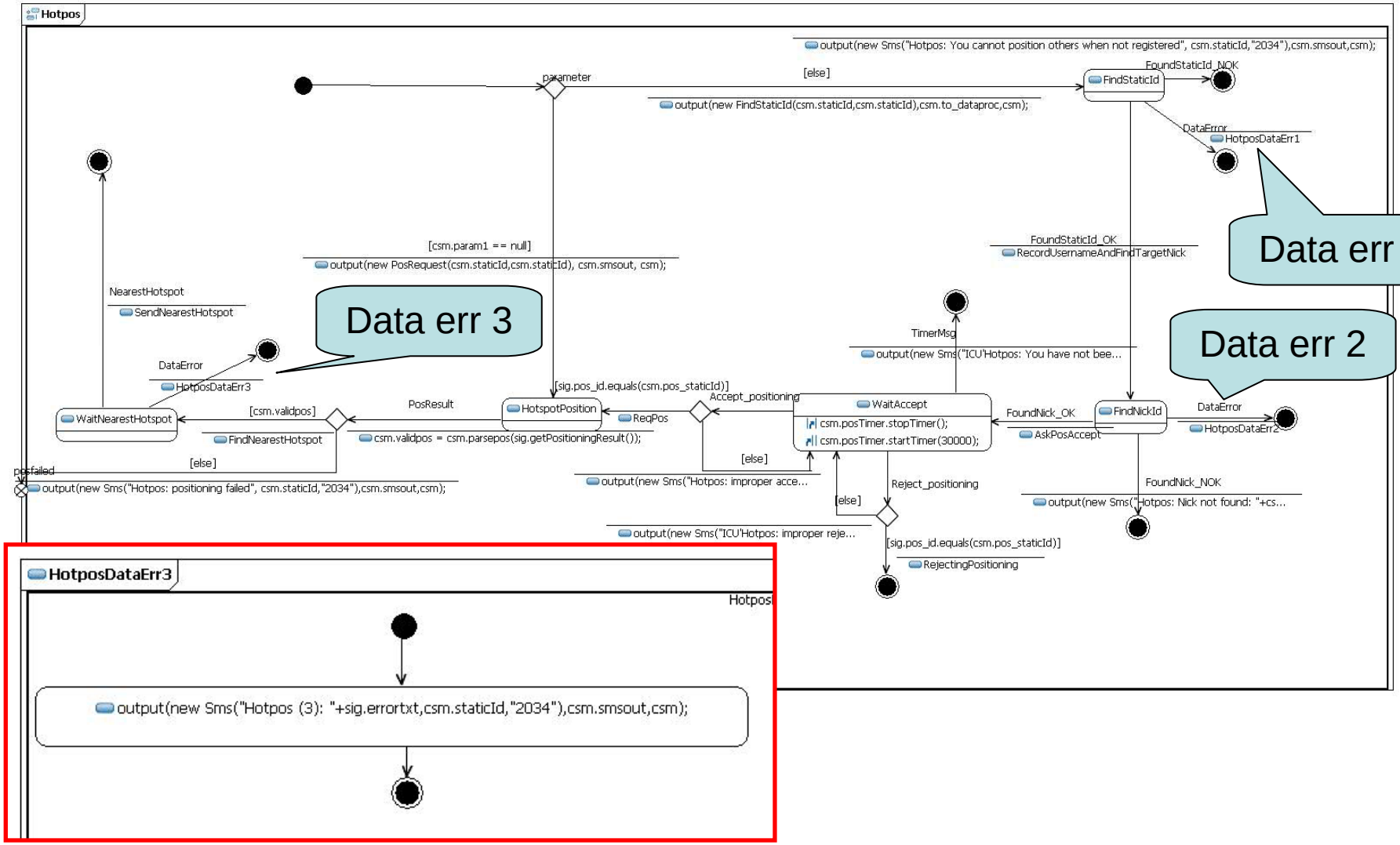
must reconnect to work again

FindStaticIdent

```
/* look in gsmusers for static id */
try {
    Statement stmt = csm.con.createStatement();
    String theQuery = "SELECT * FROM gsmuser WHERE staticid = '"+ sig.find_id + "'";
    ResultSet r = stmt.executeQuery(theQuery);
    if (r.next())
    { /* Static id found*/
        output(new FoundStaticId_OK(r.getString("nickname"),sig.find_id,sig.static_id),
        csm.to_contr,csm);
    }
    else
    { /* Static id not found */
        output(new FoundStaticId_NOK(sig.find_id,sig.static_id),csm.to_contr,csm);
    }
} catch (Exception e) {
  System.err.println("ICU'Archive: Error when Selecting staticid from gsmuser");
  output(
    new DataError("ICU'Archive: Error when Selecting staticid from gsmuser",
    sig.static_id), csm.to_contr, csm);
}
```

Double error messages: to the console and the calling service

Catching the DataError message in Hotpos



The robustification summarized

- Data may have strange syntax or values
  - We have looked at data checks for ICUcontroller
- An unexpected signal arrives
  - we explicitly describe every conceivable transition
  - We will look at this again for ”n+1” situation
- No signal arrives
  - we guard our protocols/services with timers
- Security issues
  - authentication + logging + statistics
  - Check for registration in ICUprocess’Hotpos
- Availability issues
  - liveness tests (Archive)

What more robustification could we have done?

- *KML* and *yesno* are still without authentication
  - in practice we would need a ”buddy” concept
- PATS is not checked
  - we could have covered sending Sms/PosRequest
  - probably best on lower level, but would cause some problems
- We have not tested every peculiar (but imagined) situation
  - because it is difficult/tedious to do
  - will require a very precise testing environment
- Probably should have had one more iteration of cleaning up the diagrams
  - aesthetics is important for understanding