State Machines with automatic code generation to JavaFrame

Version 060915

Our goals

- A good way of thinking for
  - modelers
  - programmers
- such that their programs will become:
  - rapidly made according to specification
  - have high quality
  - be efficient
  - maintainable by competent persons
  - be adaptive to a changing environment of requirements and third party software
- This should apply to large and small programs

Finite State Machines

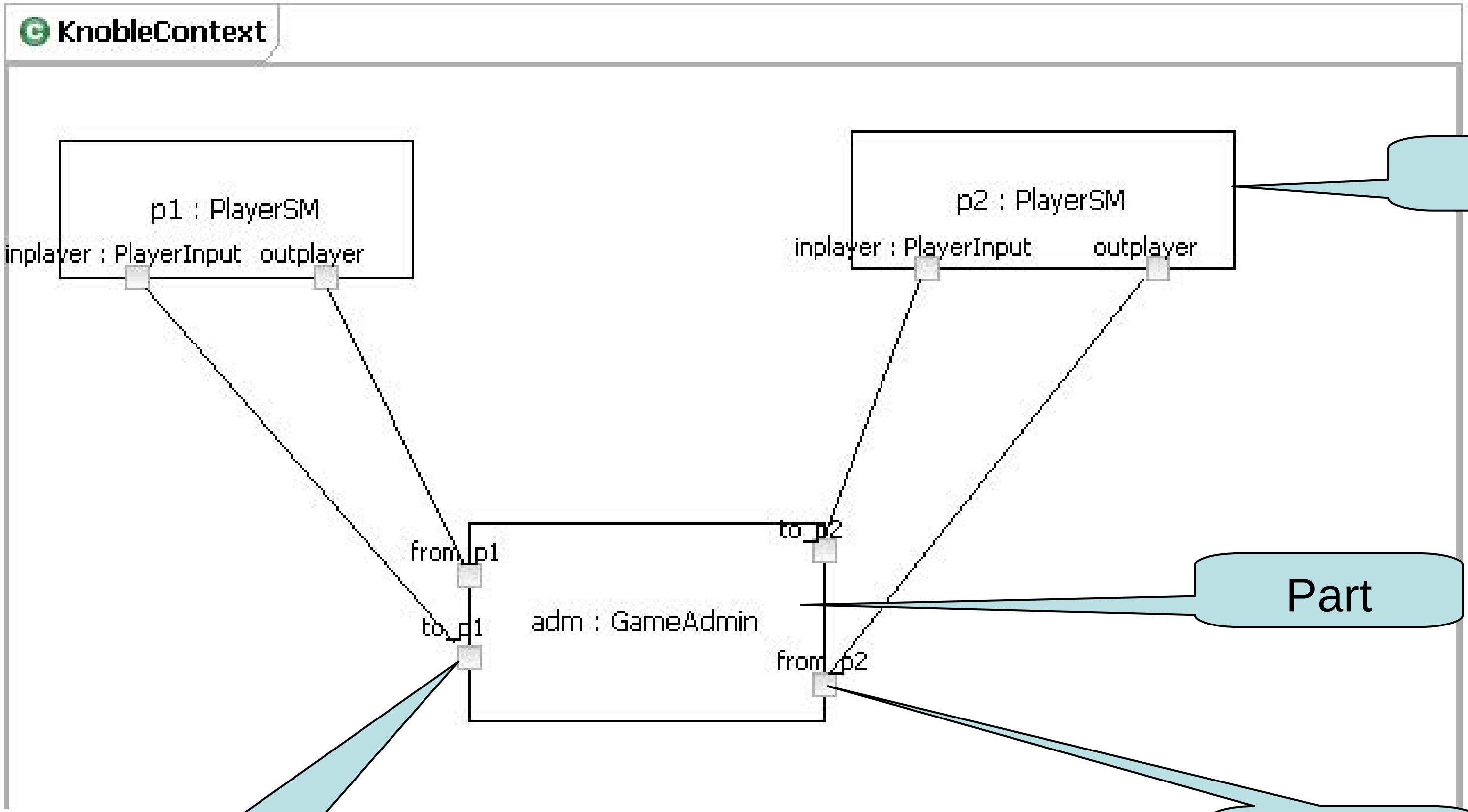
- Finite
  - a finite number of states
  - [here] a *small* number of *named* states
- State
  - a stable situation where the process awaits stimuli
  - a state in a state machine represents the history of the execution
- Machine
  - that only a stimulus (signal, message) triggers behavior
  - the behavior consists of executing transitions
  - may also have local data

The Knoble game

- A game administrator controls the game
- Invites the players
- The players make a draw like:

- The game administrator calculates the scores

The Knoble context



composite structure

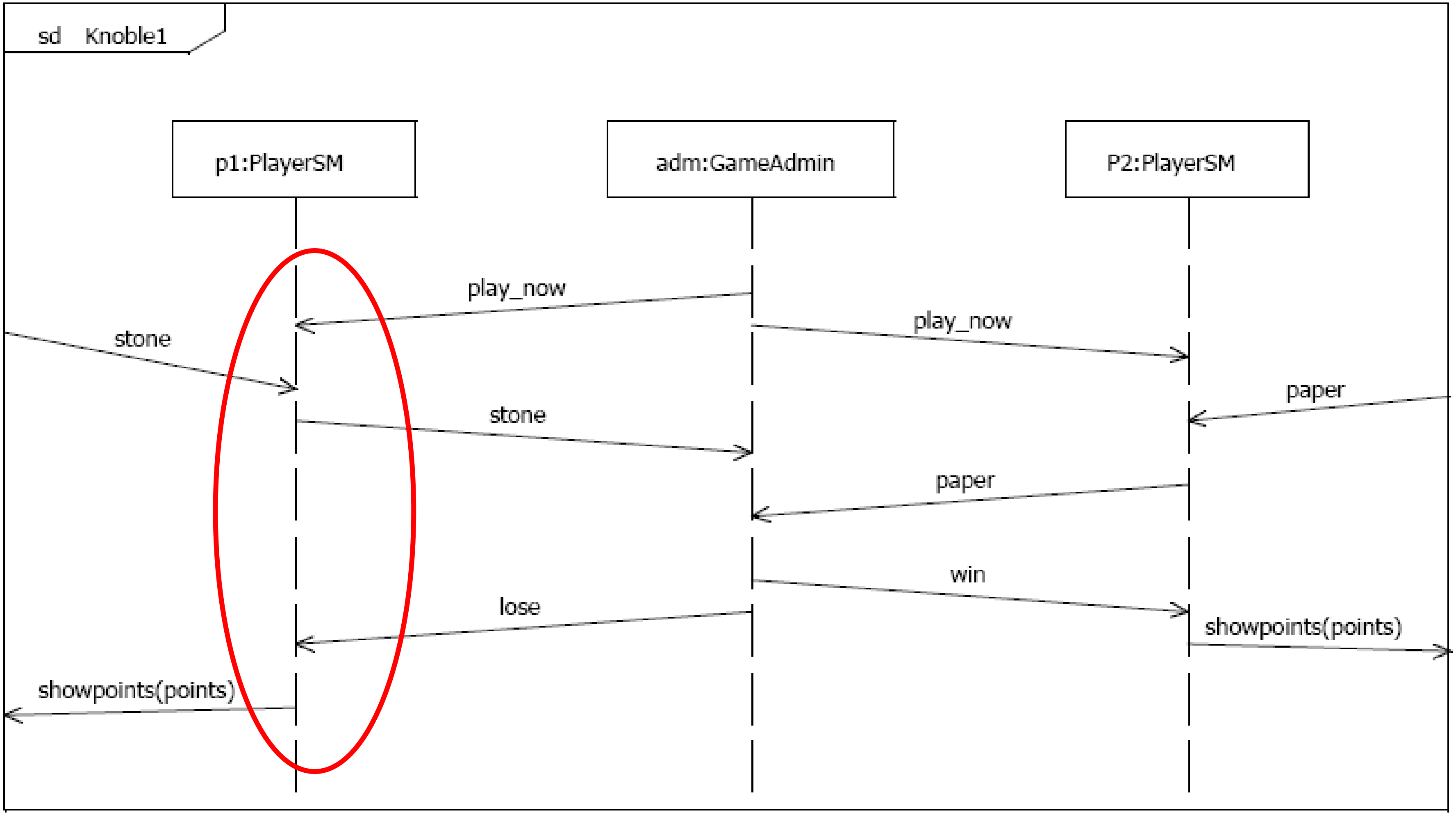
Part

Part

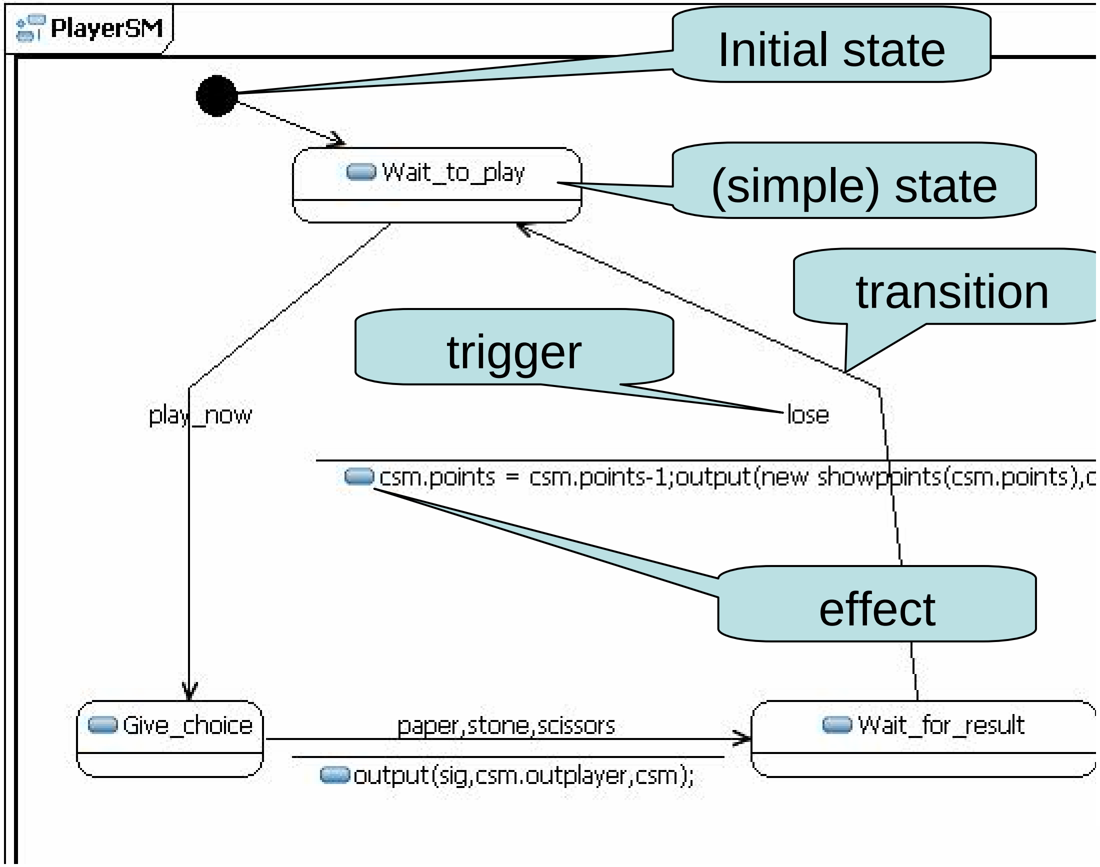
output port

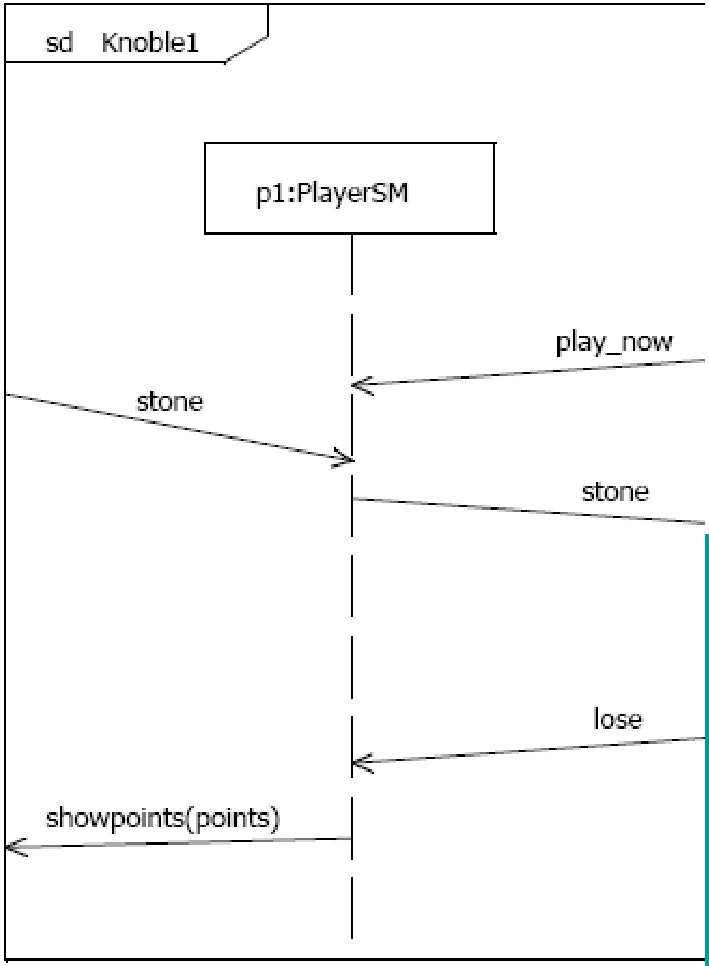
input port

What happens?

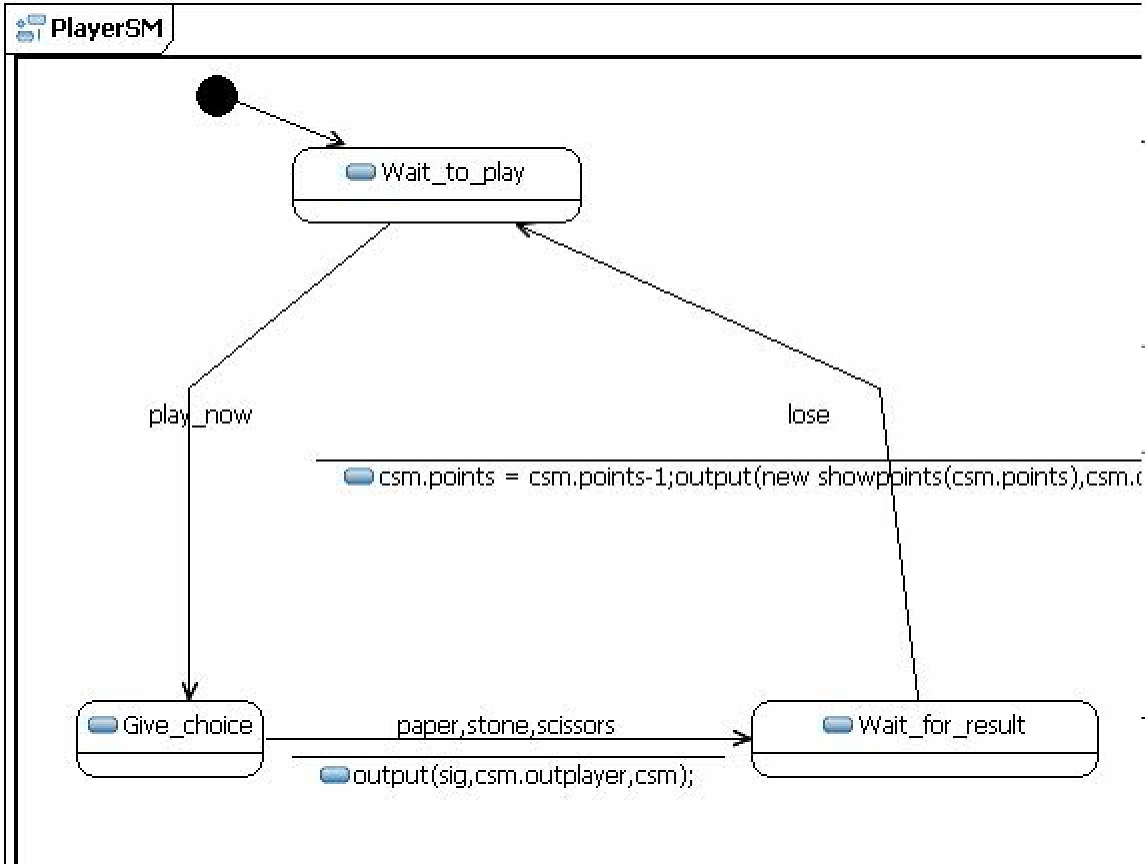


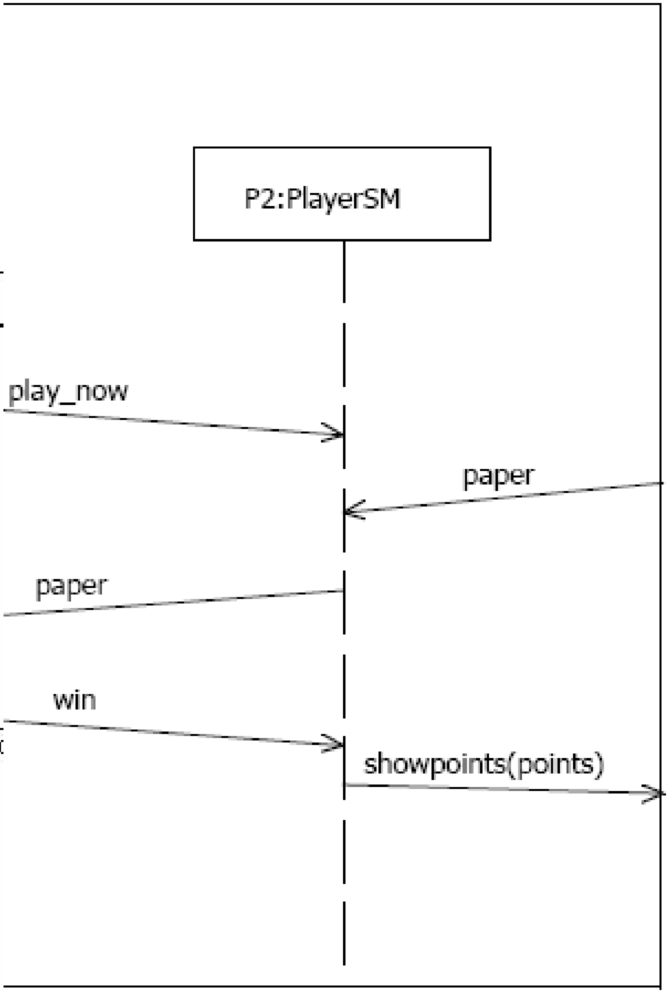
Player: first attempt



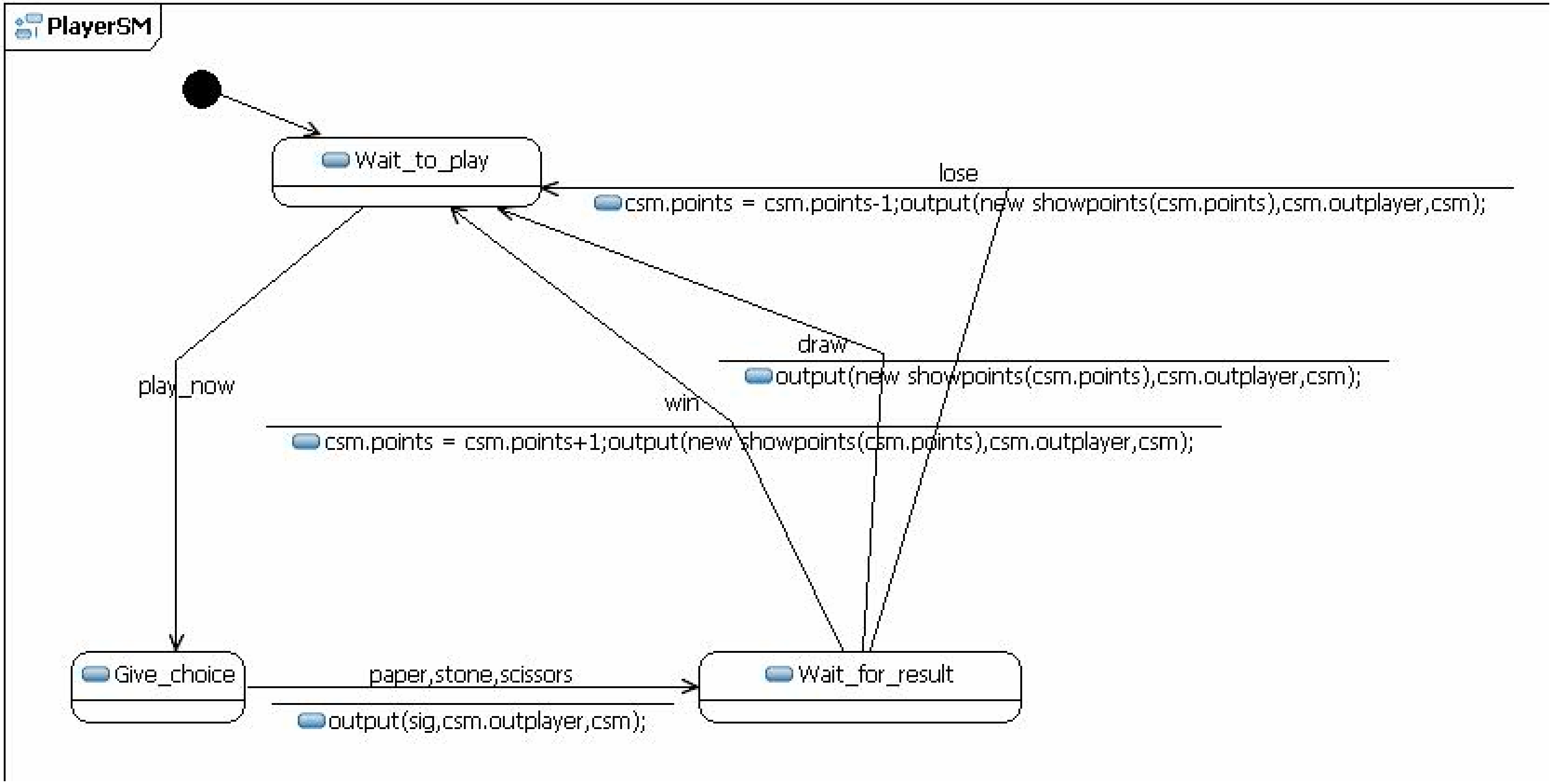


Player: why it is not sufficient

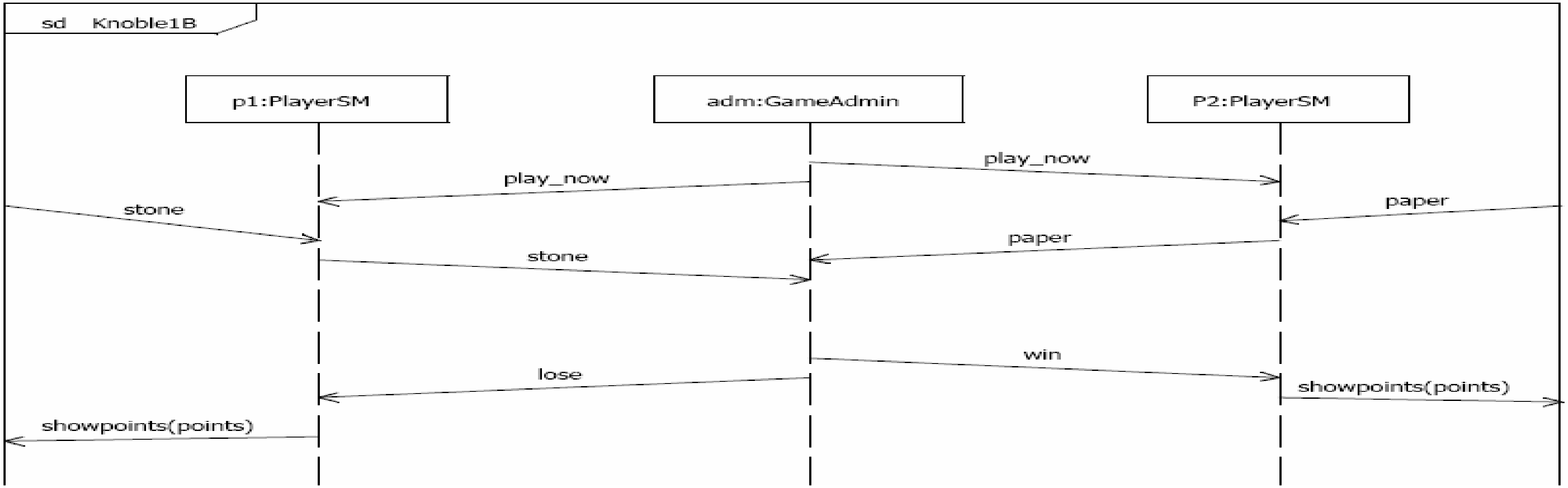

PlayerSM
Wait_to_play
play_now
lose
csm.points = csm.points-1;output(new showpoints(csm.points),csm.(
Give_choice
paper,stone,scissors
output(sig,csm.outplayer,csm);
Wait_for_result

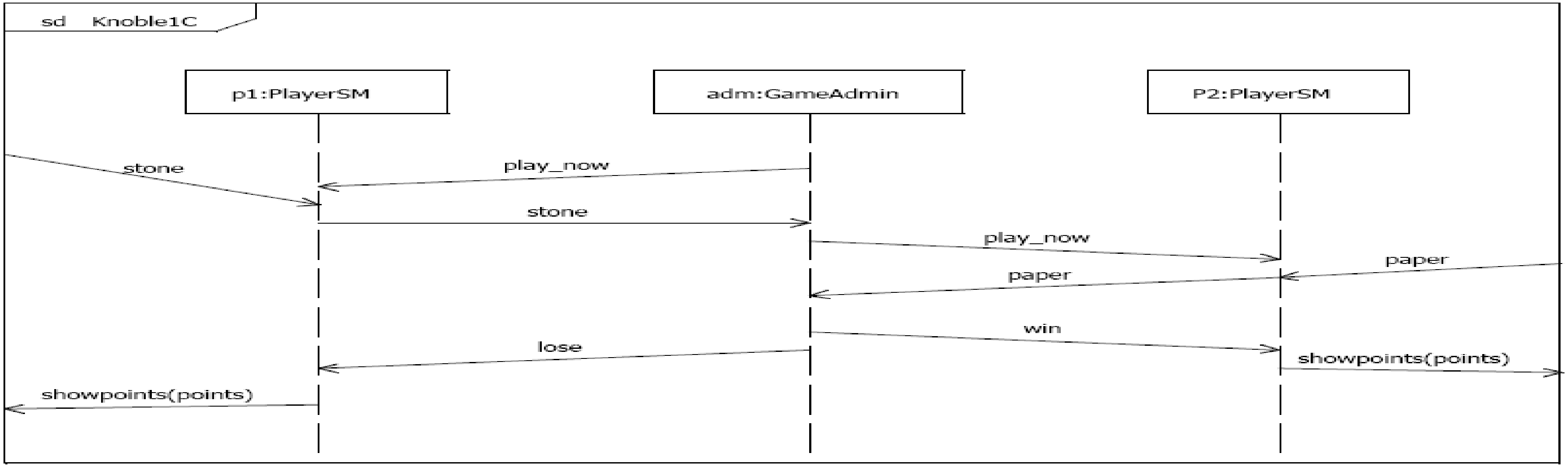


Player: second solution

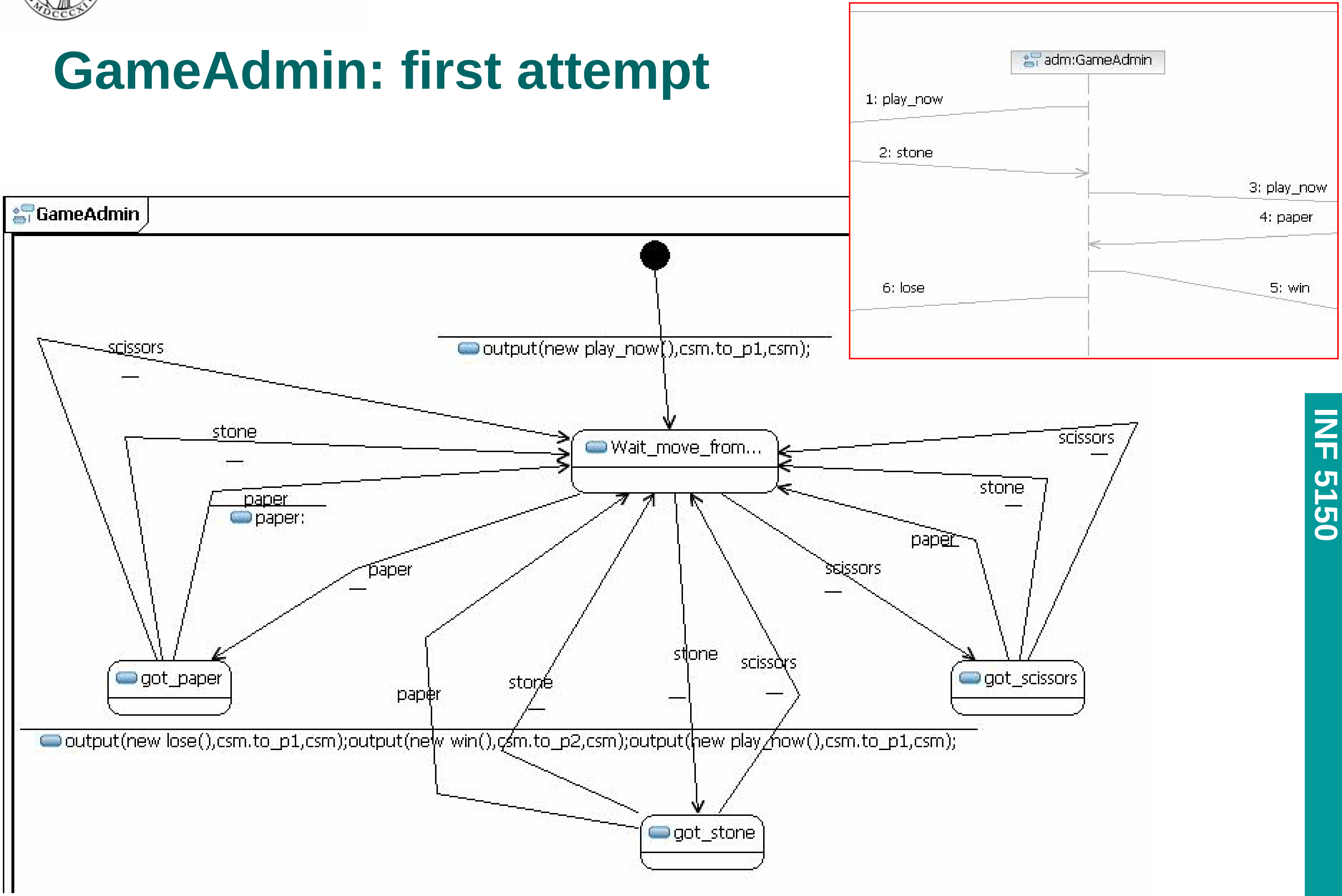

PlayerSM
Wait_to_play
lose
csm.points = csm.points-1;output(new showpoints(csm.points),csm.outplayer,csm);
draw
output(new showpoints(csm.points),csm.outplayer,csm);
play_now
win
csm.points = csm.points+1;output(new showpoints(csm.points),csm.outplayer,csm);
Give_choice
paper,stone,scissors
output(sig,csm.outplayer,csm);
Wait_for_result

GameAdmin: are these diagrams acceptable?

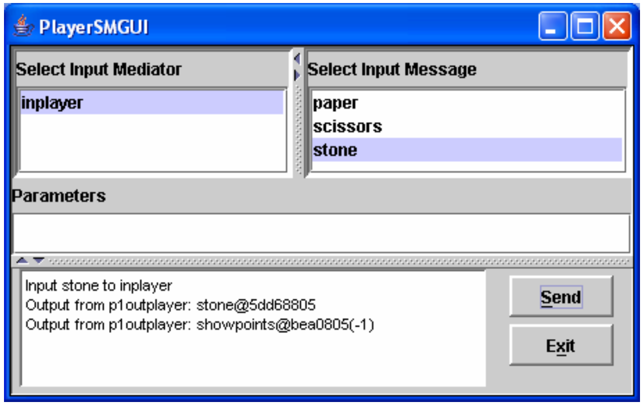




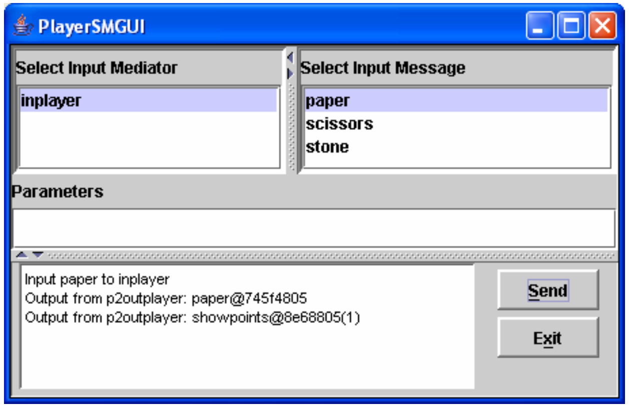
GameAdmin: first attempt



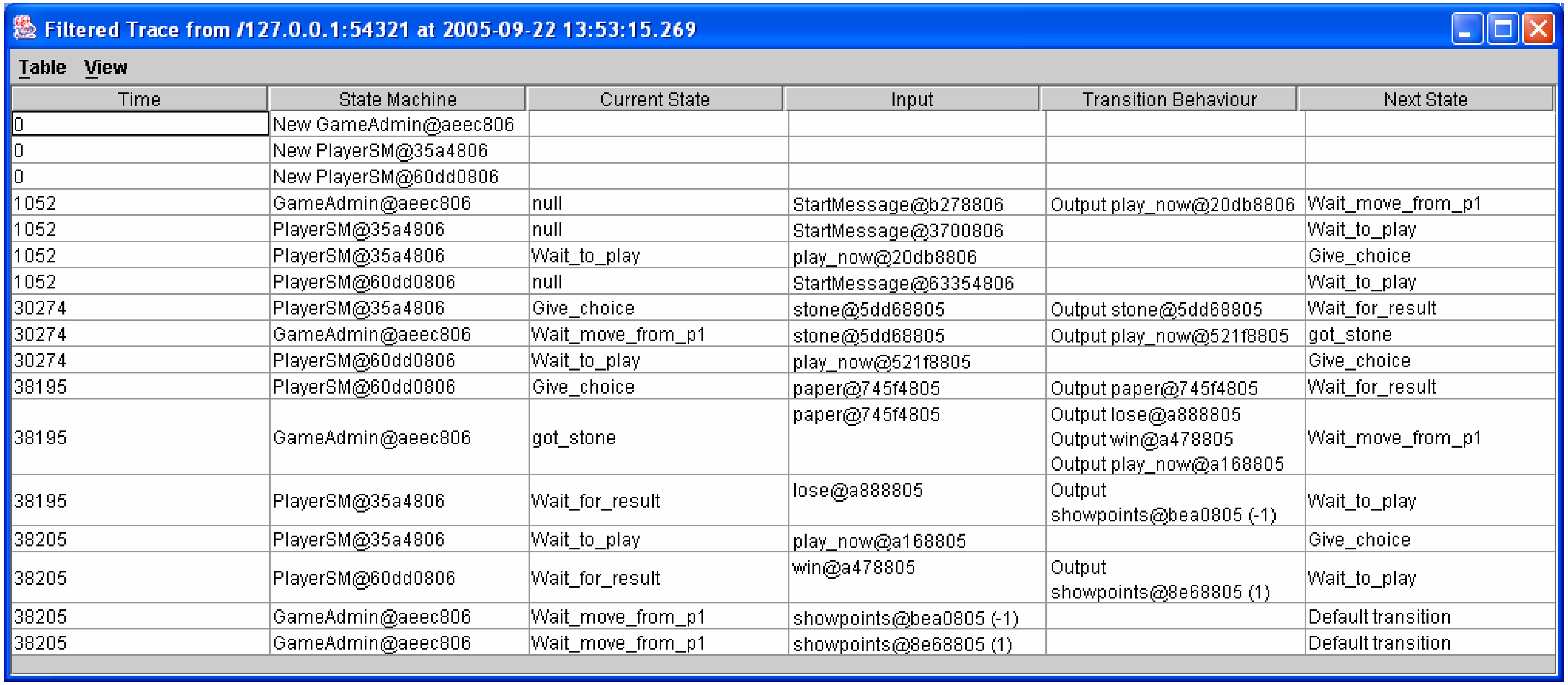
Demo Knoble2 (running JFTrace)



```
Input stone to inplayer
Output from p1outplayer: stone@5dd68805
Output from p1outplayer: showpoints@bea0805(-1)
```

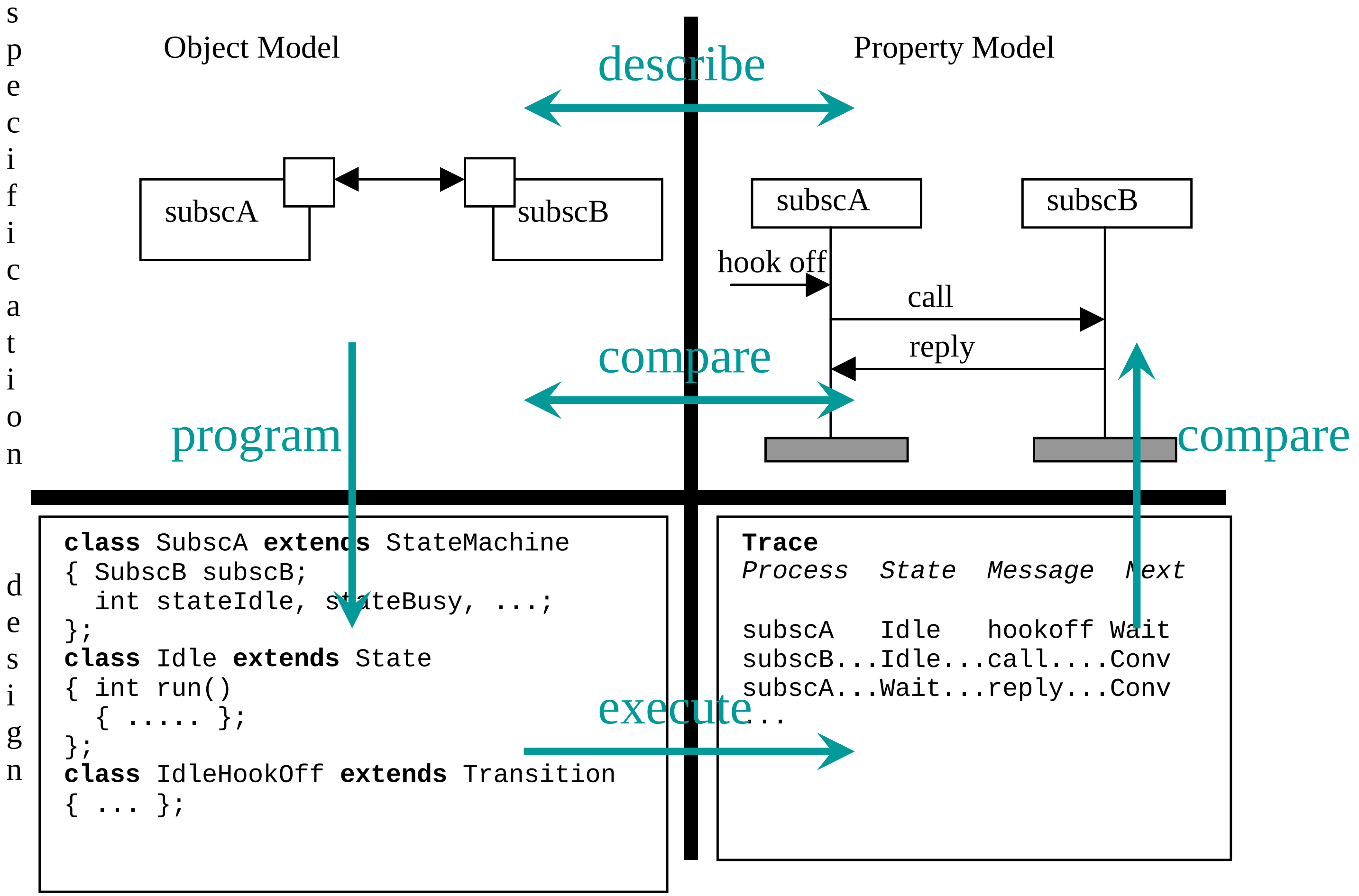


```
Input paper to inplayer
Output from p2outplayer: paper@745f4805
Output from p2outplayer: showpoints@8e68805(1)
```


Filtered Trace from /127.0.0.1:54321 at 2005-09-22 13:53:15.269
Table
View

Time	State Machine	Current State	Input	Transition Behaviour	Next State
0	New GameAdmin@aeec806				
0	New PlayerSM@35a4806				
0	New PlayerSM@60dd0806				
1052	GameAdmin@aeec806	null	StartMessage@b278806	Output play_now@20db8806	Wait_move_from_p1
1052	PlayerSM@35a4806	null	StartMessage@3700806		Wait_to_play
1052	PlayerSM@35a4806	Wait_to_play	play_now@20db8806		Give_choice
1052	PlayerSM@60dd0806	null	StartMessage@63354806		Wait_to_play
30274	PlayerSM@35a4806	Give_choice	stone@5dd68805	Output stone@5dd68805	Wait_for_result
30274	GameAdmin@aeec806	Wait_move_from_p1	stone@5dd68805	Output play_now@521f8805	got_stone
30274	PlayerSM@60dd0806	Wait_to_play	play_now@521f8805		Give_choice
38195	PlayerSM@60dd0806	Give_choice	paper@745f4805	Output paper@745f4805	Wait_for_result
38195	GameAdmin@aeec806	got_stone	paper@745f4805	Output lose@a888805 Output win@a478805 Output play_now@a168805	Wait_move_from_p1
38195	PlayerSM@35a4806	Wait_for_result	lose@a888805	Output showpoints@bea0805 (-1)	Wait_to_play
38205	PlayerSM@35a4806	Wait_to_play	play_now@a168805		Give_choice
38205	PlayerSM@60dd0806	Wait_for_result	win@a478805	Output showpoints@8e68805 (1)	Wait_to_play
38205	GameAdmin@aeec806	Wait_move_from_p1	showpoints@bea0805 (-1)		Default transition
38205	GameAdmin@aeec806	Wait_move_from_p1	showpoints@8e68805 (1)		Default transition

UML JavaFrame Profile Model analysis



```
class SubscA extends StateMachine
{ SubscB subscB;
  int stateIdle, stateBusy, ...;
};
class Idle extends State
{ int run()
  { ..... };
};
class IdleHookOff extends Transition
{ ... };
```

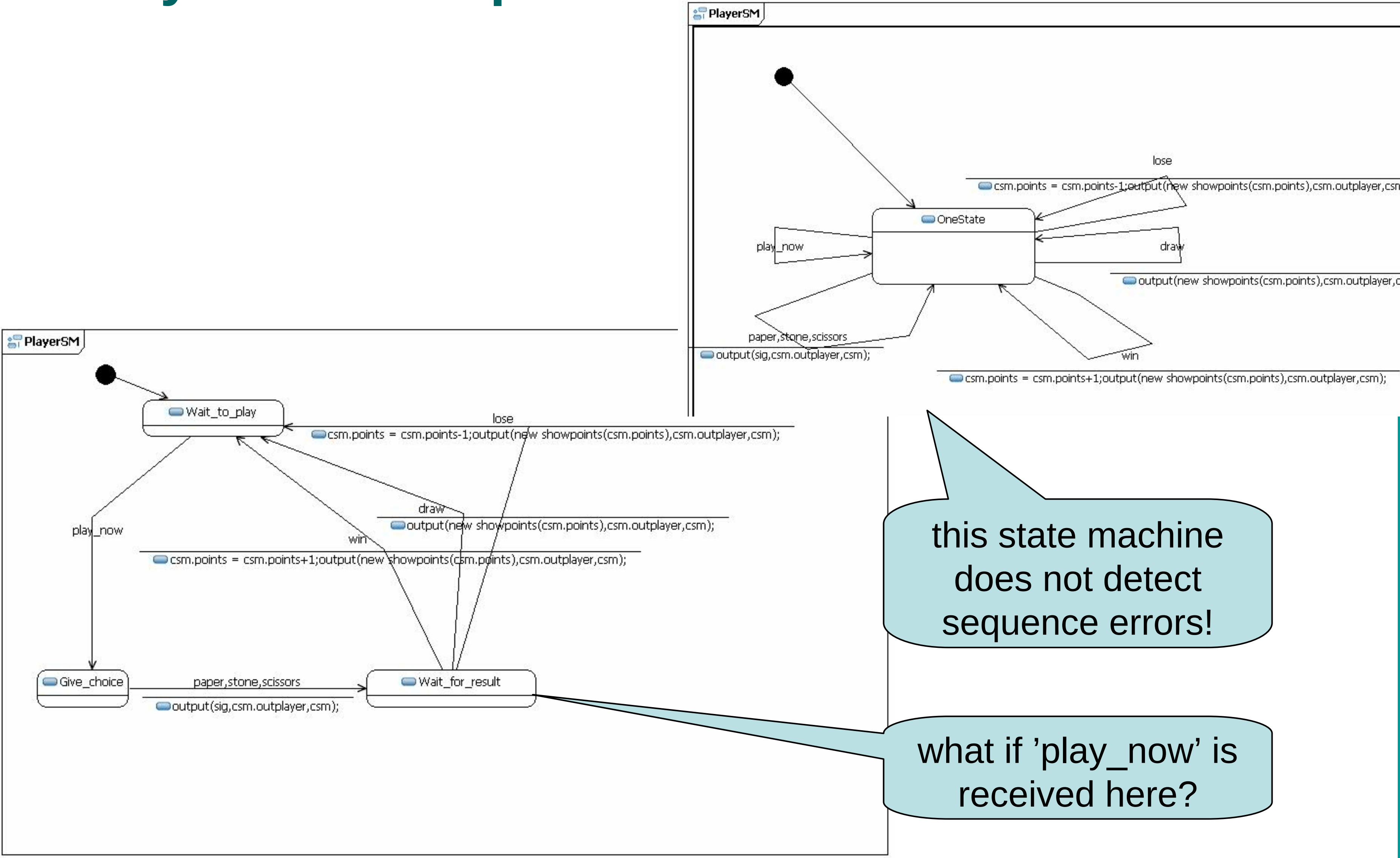
```
Trace
Process   State   Message   Next

subscA    Idle    hookoff Wait
subscB...Idle...call....Conv
subscA...Wait...reply...Conv
```

State Machines: unassailability?

- Understandable
  - think locally, act globally
  - states represent compressed representation of execution history
- Robust
  - detect errors through discovering undefined transitions
- Maintainable
  - make additions and alterations with a minimum of ripple effects
- Analyzable
  - systems of state machines can be handled by model checkers
  - compare sequence diagrams with state machine(s)

PlayerSM: Compare these versions!


PlayerSM
OneState
lose
csm.points = csm.points-1;output(new showpoints(csm.points),csm.outplayer,csm);
play_now
draw
output(new showpoints(csm.points),csm.outplayer,csm);
paper,stone,scissors
output(sig,csm.outplayer,csm);
win
csm.points = csm.points+1;output(new showpoints(csm.points),csm.outplayer,csm);
PlayerSM
Wait_to_play
lose
csm.points = csm.points-1;output(new showpoints(csm.points),csm.outplayer,csm);
draw
output(new showpoints(csm.points),csm.outplayer,csm);
play_now
win
csm.points = csm.points+1;output(new showpoints(csm.points),csm.outplayer,csm);
Give_choice
paper,stone,scissors
output(sig,csm.outplayer,csm);
Wait_for_result

this state machine does not detect sequence errors!

what if ’play_now’ is received here?

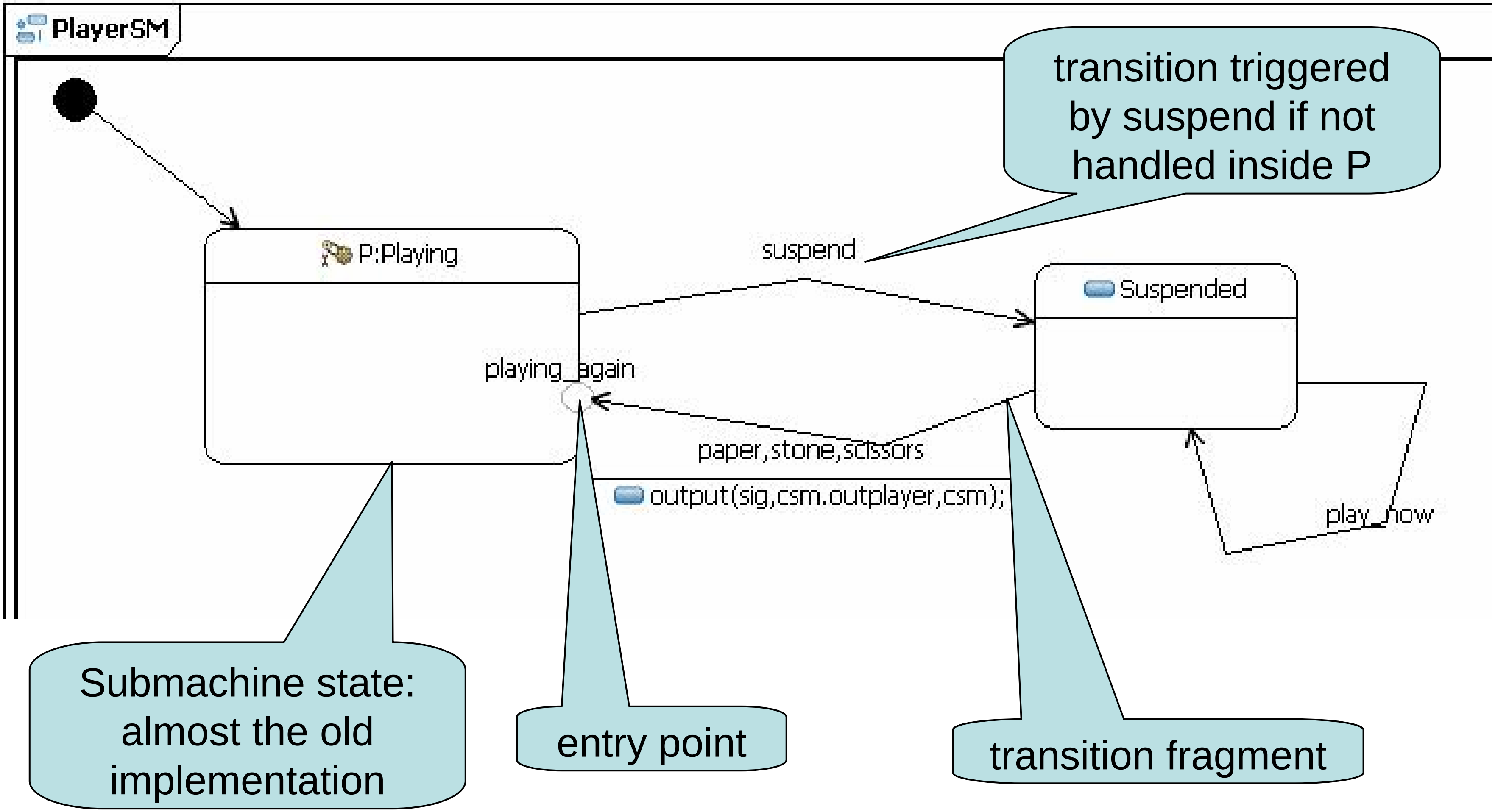
And now adding a new feature ...

and by adding a feature to the model,
needing another feature from the language

Knoble: Now we add another requirement

- Assume that the Player may at any time receive a ’suspend’ message from the GUI
- This should have the effect that
  - the player will not play
  - until he/she receives a paper/stone/scissors message from GUI
    - then such a message is directly a move
- We would like to make this change
  - as compact as possible
  - without changing much of what is already made functioning

PlayerSM: Introducing Submachine states



transition triggered by suspend if not handled inside P

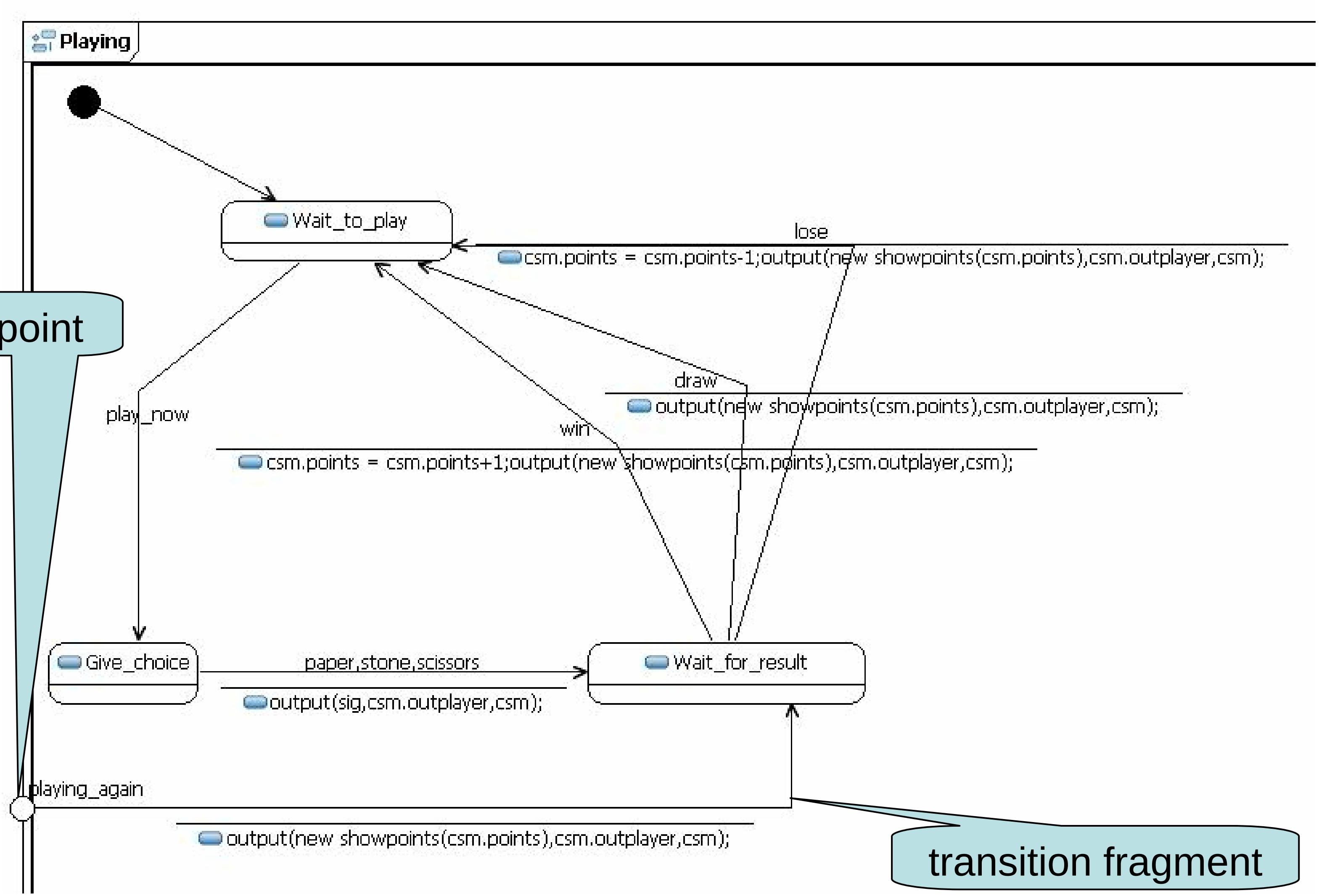
Submachine state: almost the old implementation

entry point

transition fragment

INF 5150

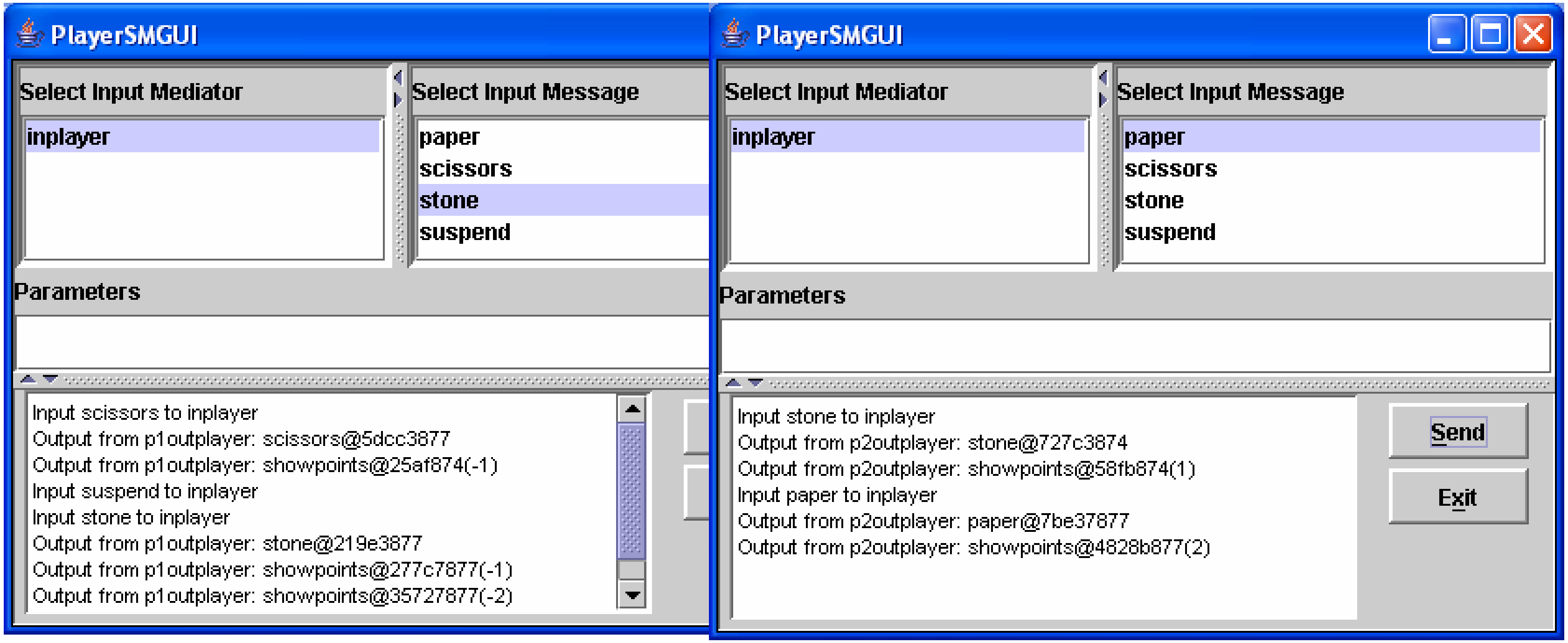
Playing: almost like the old Player with entry


Playing
Wait_to_play
lose
csm.points = csm.points-1;output(new showpoints(csm.points),csm.outplayer,csm);
draw
output(new showpoints(csm.points),csm.outplayer,csm);
play_now
win
csm.points = csm.points+1;output(new showpoints(csm.points),csm.outplayer,csm);
Give_choice
paper,stone,scissors
output(sig,csm.outplayer,csm);
Wait_for_result
playing_again
output(new showpoints(csm.points),csm.outplayer,csm);

entry point

transition fragment

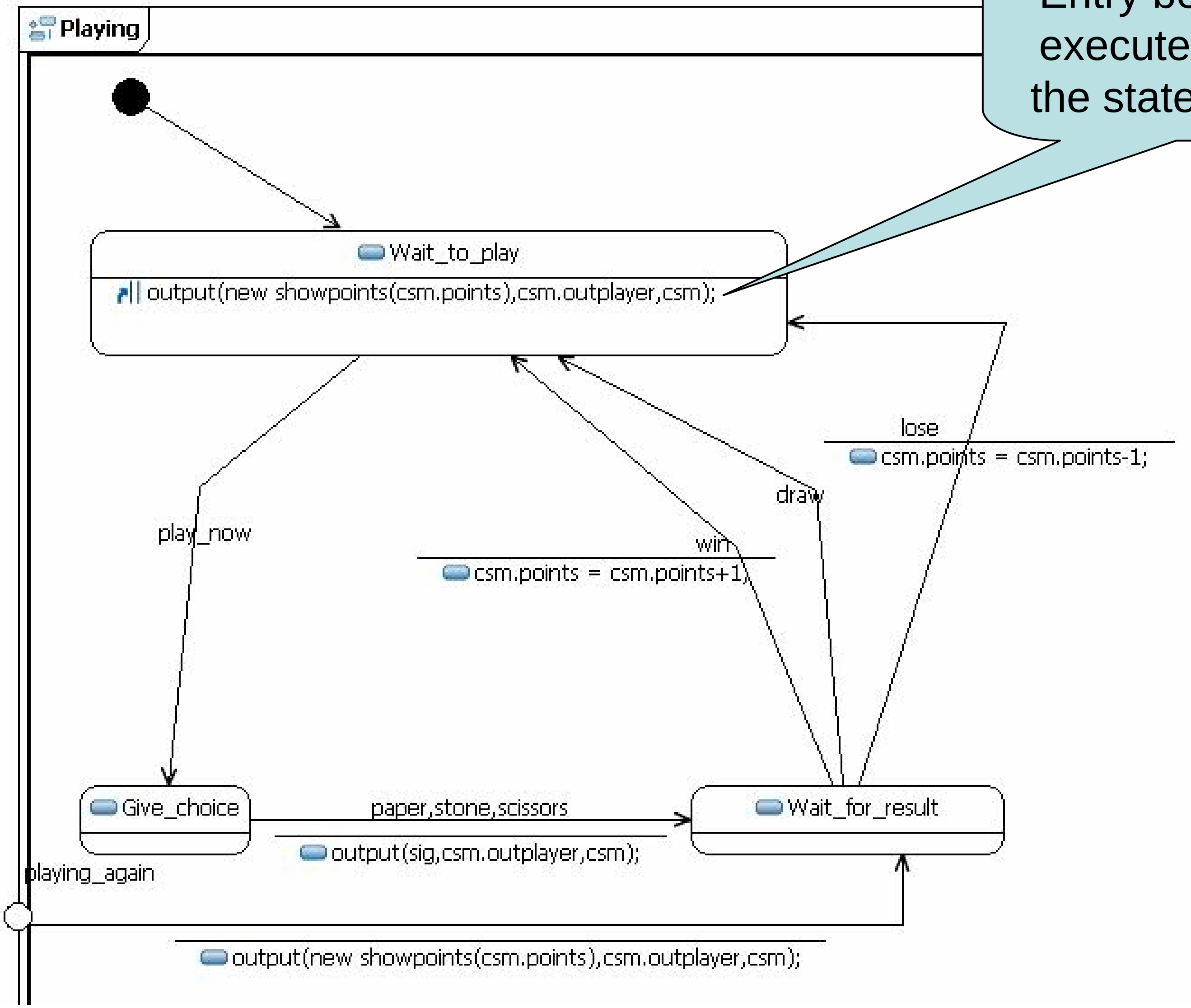
Demo Knoble4 (not running JFTrace)



```
Input scissors to inplayer
Output from p1outplayer: scissors@5dcc3877
Output from p1outplayer: showpoints@25af874(-1)
Input suspend to inplayer
Input stone to inplayer
Output from p1outplayer: stone@219e3877
Output from p1outplayer: showpoints@277c7877(-1)
Output from p1outplayer: showpoints@35727877(-2)
```

```
Input stone to inplayer
Output from p2outplayer: stone@727c3874
Output from p2outplayer: showpoints@58fb874(1)
Input paper to inplayer
Output from p2outplayer: paper@7be37877
Output from p2outplayer: showpoints@4828b877(2)
```

Entry and Exit behaviors


Playing
Wait_to_play
output(new showpoints(csm.points),csm.outplayer,csm);
lose
csm.points = csm.points-1;
draw
play_now
win
csm.points = csm.points+1;
Give_choice
paper,stone,scissors
output(sig,csm.outplayer,csm);
Wait_for_result
playing_again
output(new showpoints(csm.points),csm.outplayer,csm);

Entry behavior will execute whenever the state is entered

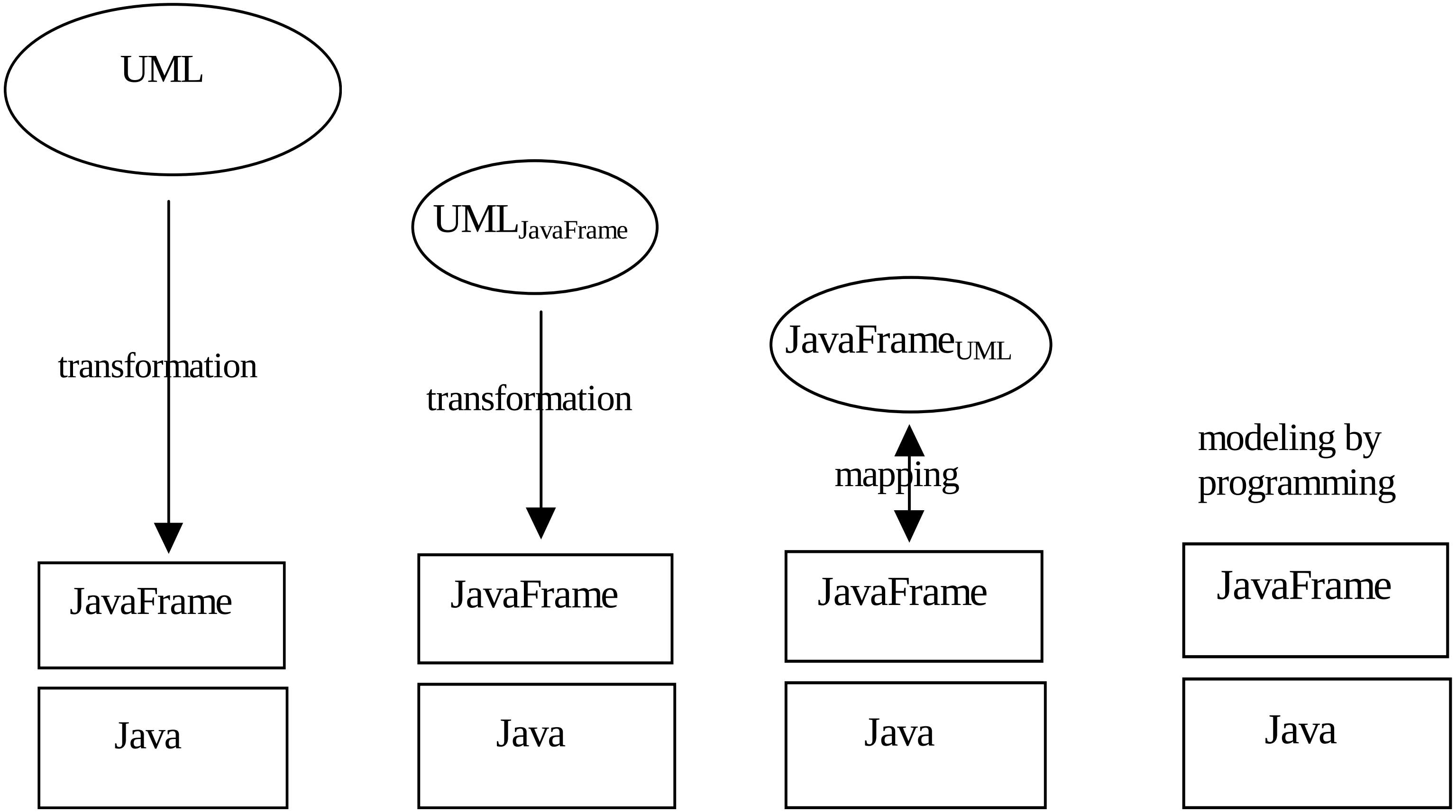
Summary State Machines

- State
  - finite number
  - simple or composite (submachine states)
- Transition
  - trigger
  - effect
- Exit and Entry Points
  - interface points within a runtime transition
- Exit and Entry Behaviors
  - behavior to be executed every time the machine exits or enters the state
- State machines may have variables (and parameters)

JavaFrame – the target framework

which can in principle be used all by itself

UML and Java: JavaFrame - the solution



UML

transformation

JavaFrame

Java

$\text{UML}_{\text{JavaFrame}}$

transformation

JavaFrame

Java

$\text{JavaFrame}_{\text{UML}}$

mapping

JavaFrame

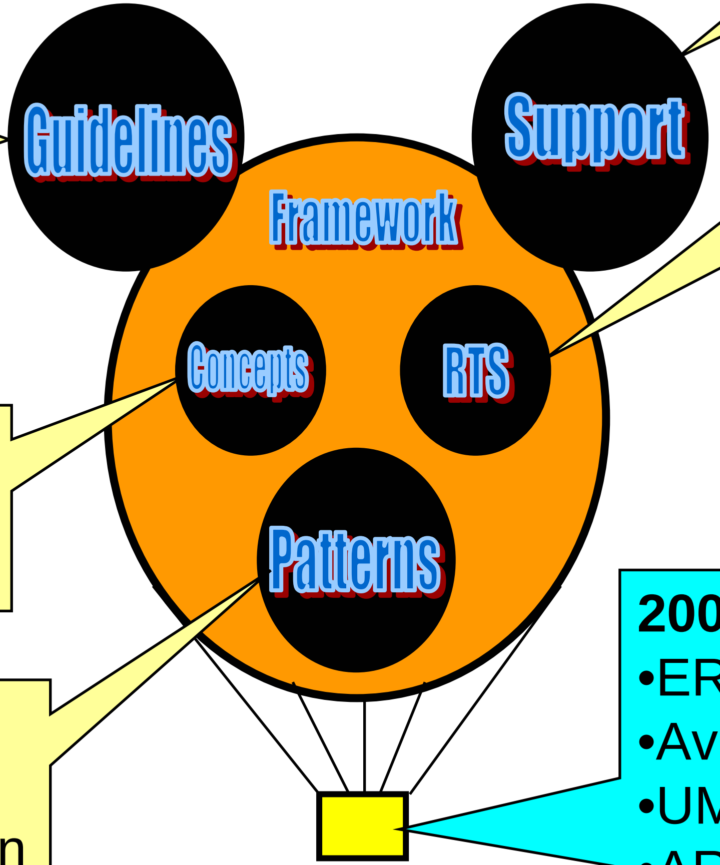
Java

modeling by
programming

JavaFrame

Java

JavaFrame – the object framework



Observation tool, input dialog

INF 5150 / INF 2120 .

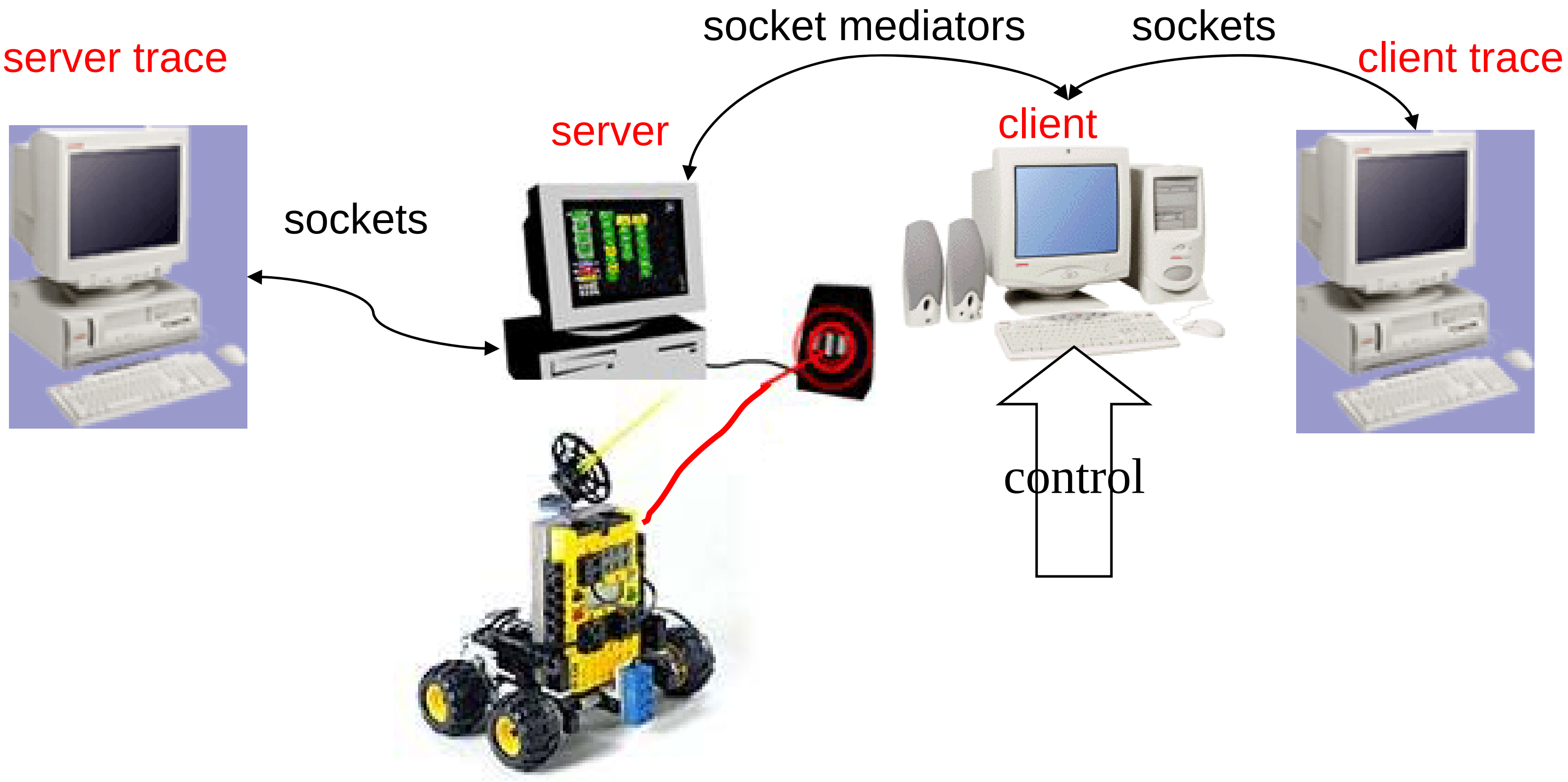
Thread-safe, runtime effective, reentrant composite states

Asynch. Interacting Active Objects

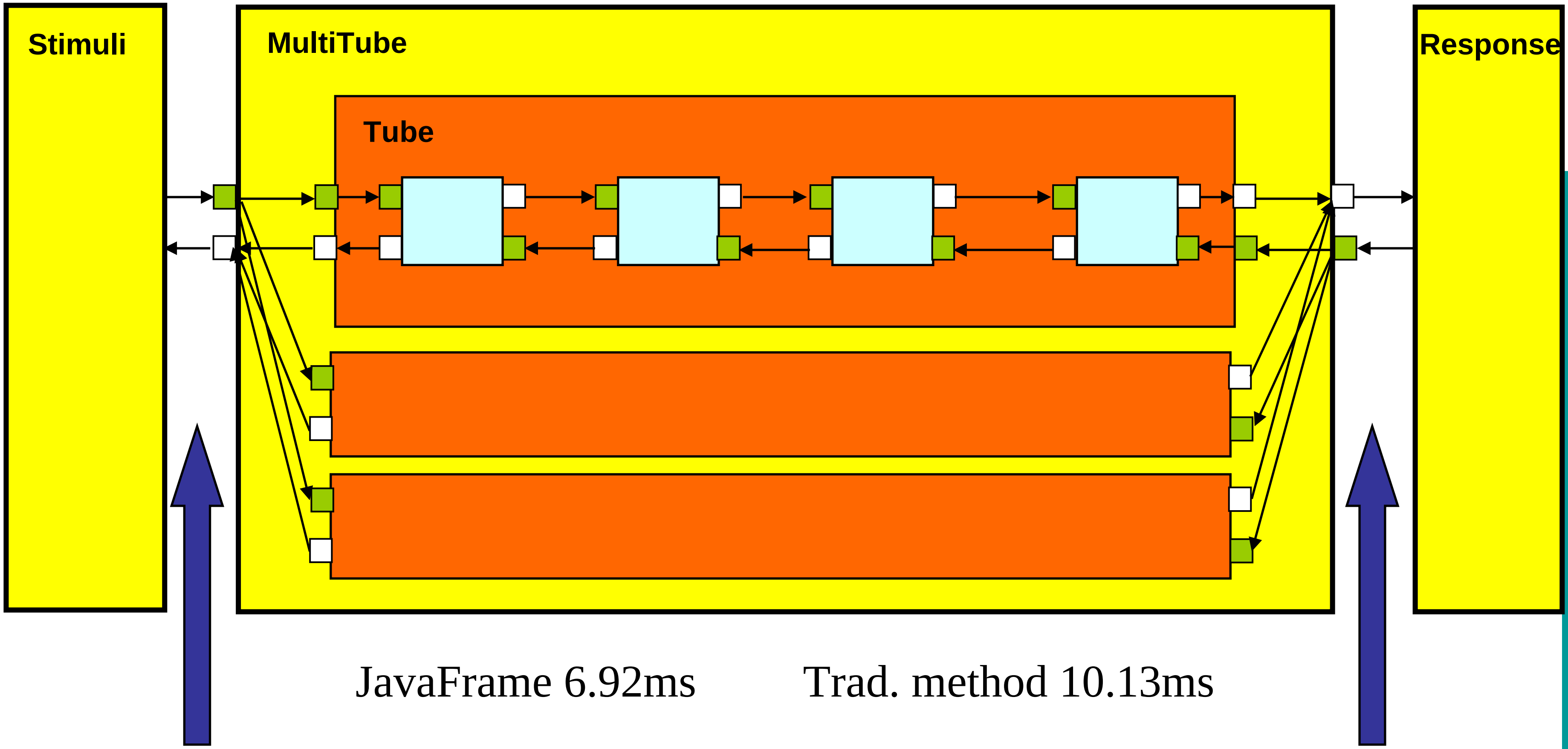
UML to JavaFrame transformation

2002 – 2004:
- •ERICSSON:
- •Avantel: Amigos
- •UML 2.0 laboratory
- •ARTS

Experiences - The Lego Mindstorm experiment



Experiences - The Performance Model



JavaFrame 6.92ms Trad. method 10.13ms

JavaFrame transition vocabulary

- sending asynchronous signals
  - output*(<the signal>*,*<the port>*,*<current state machine>*)
- <the signal>
  - new *SignalType(parameters)*
  - sig
    - meaning the signal just consumed as trigger
- <current state machine>
  - csm
- <the port>
  - csm.*portname*
- State machine variables
  - csm.*variablename*

RSM coding rules for state machines (1)

- Trigger of transitions
  - Name of the transition
  - or Generate a SignalTrigger by rightclicking on transition
- Effect of transition
  - Name of effect
  - or Use one Action within an Activity diagram (forget flow lines etc.) created when doubleclicking the effect icon.
- Inside the effect
  - JavaFrame statements
  - or Branch by using Choice points
    - outgoing transitions from a choice point should have a guard (predicate condition for this piece of the transition)

RSM coding rules for state machines (2)

- output (Signal, Port, csm)
  - sends a signal through the local port.
  - typically the signal is like ”new S(parm1, parm2)”
  - typically the port is like ”csm.toSomewhere”
  - ”csm” is like a keyword meaning ”current state machine”
- To read from the consumed signal, use ”sig”
  - sig has been cast to the right type (normally)
  - Example: ”sig.parm1” when sig is consumed as object of class S
- UML defer
  - to add a **deferrable trigger**, make sure the trigger to be deferred has a signaltrigger element in the state machine
  - right click the state > Properties > DeferrableTrigger and add the appropriate signaltrigger.
  - But you will not see the **defer** in the diagram – only in the model