



# State Machines with automatic code generation to JavaFrame

Version 050923 Revision 1





## 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

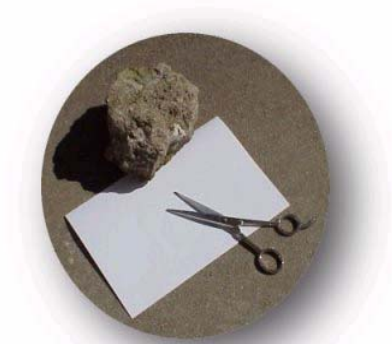
- Finite
  - a finite number of states
  - [here] a *small* number of *named* states
- State
  - a stable situation where the process awaits stimuli
  - a state in a state machine represents the history of the execution
- Machine
  - that only a stimulus (signal, message) triggers behavior
  - the behavior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



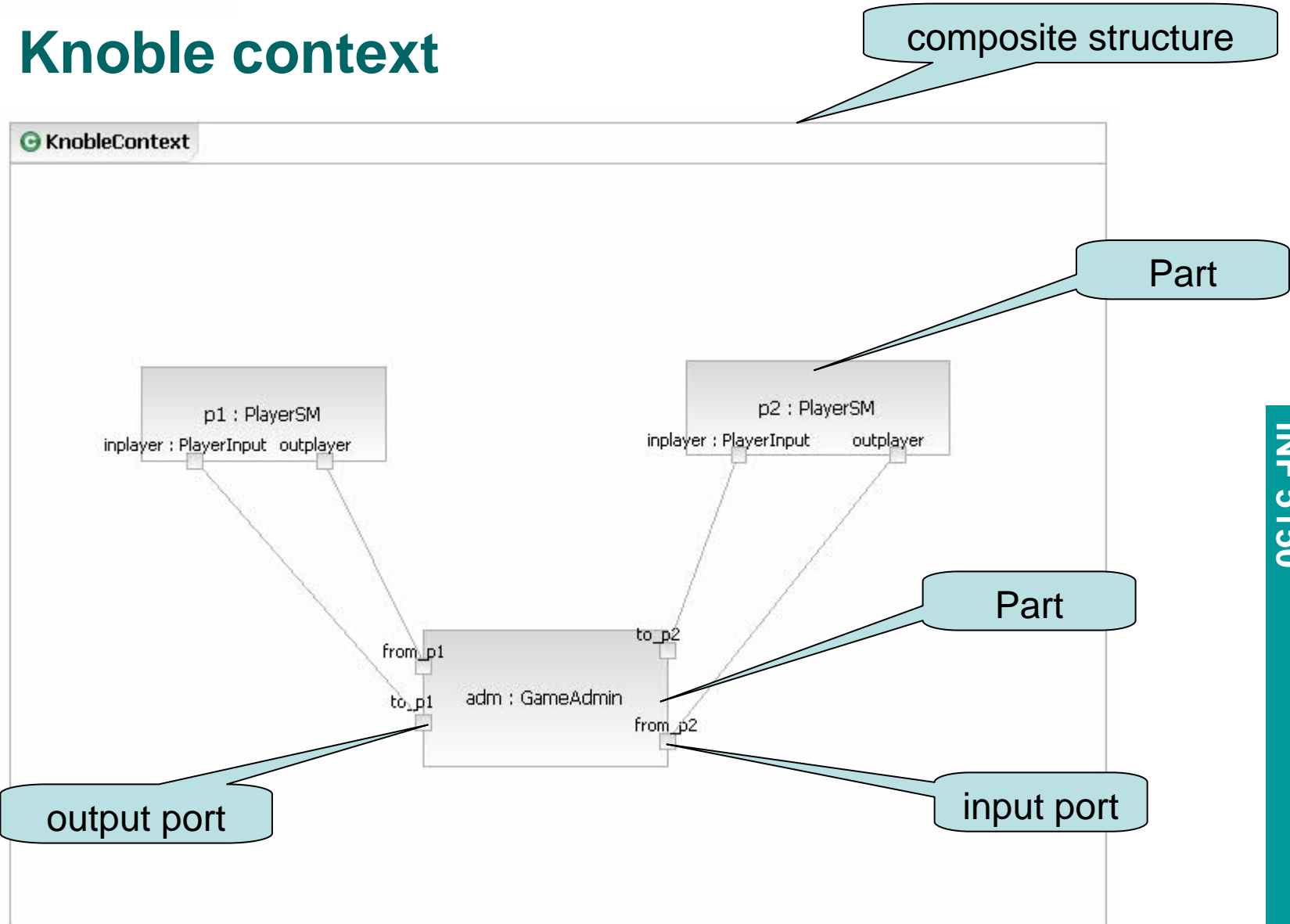
Rock, Scissors, Paper +

- The game administrator calculates the scores

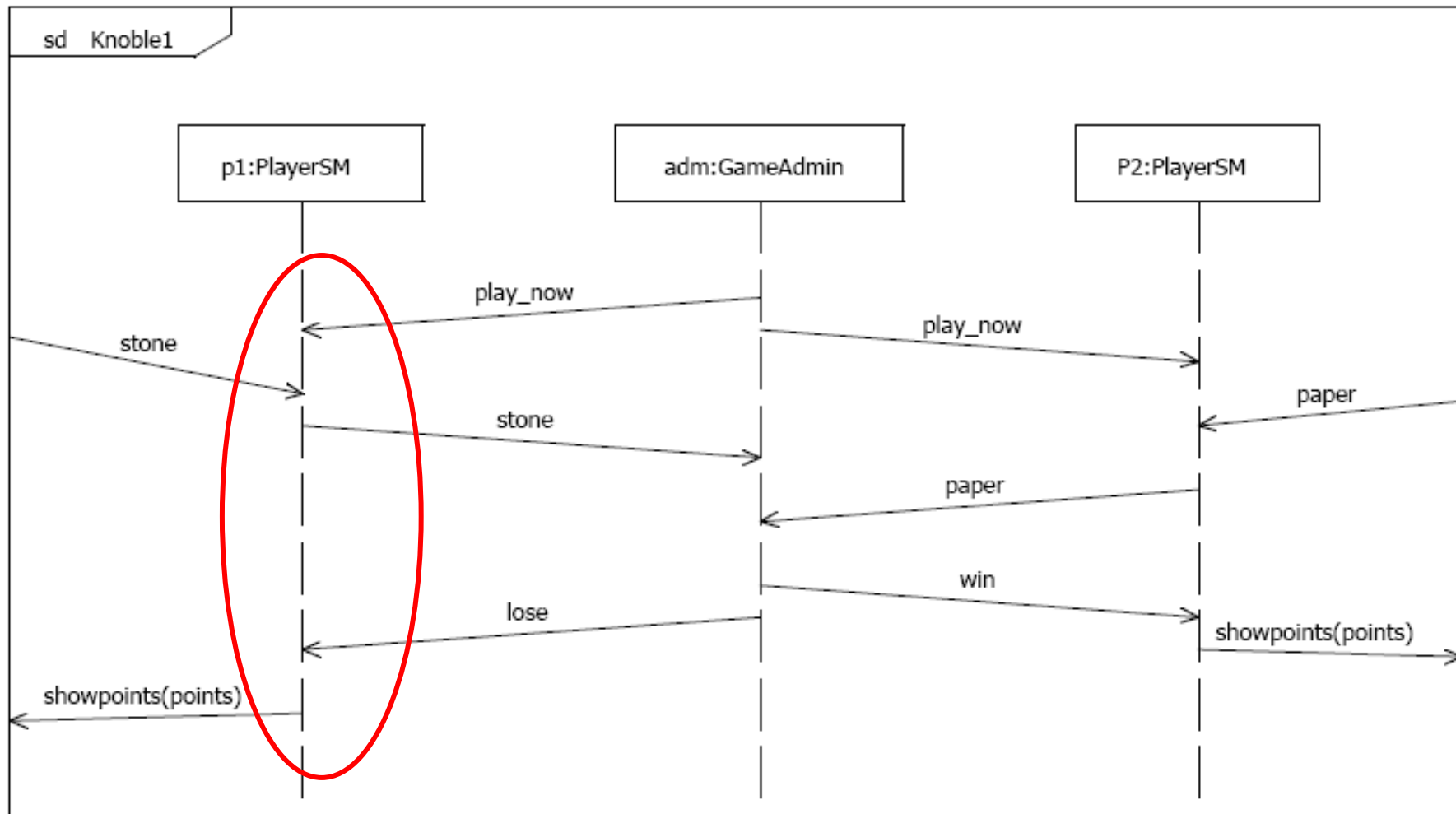




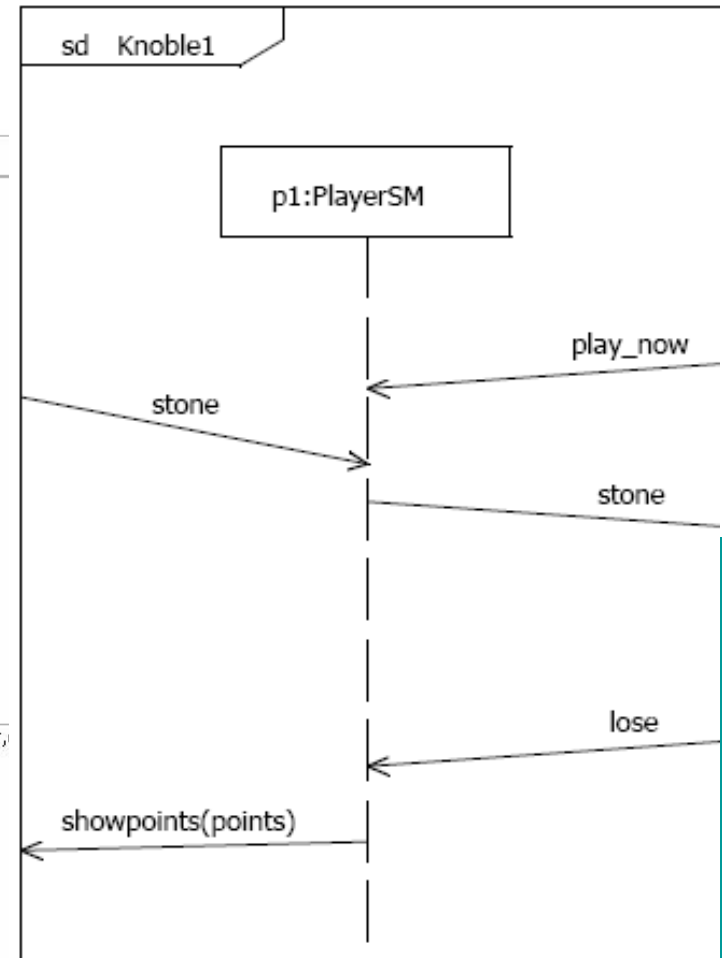
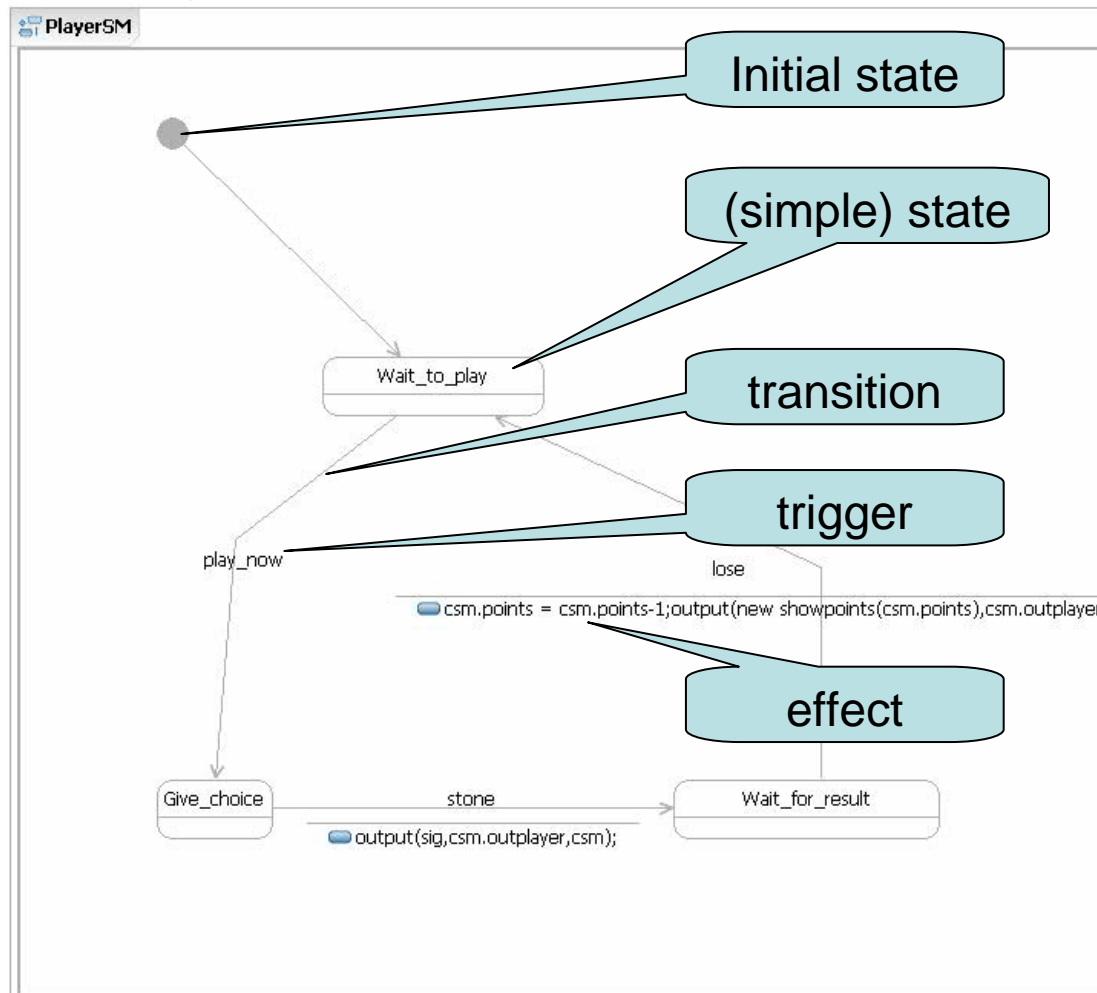
# The Knoble context



# What happens?



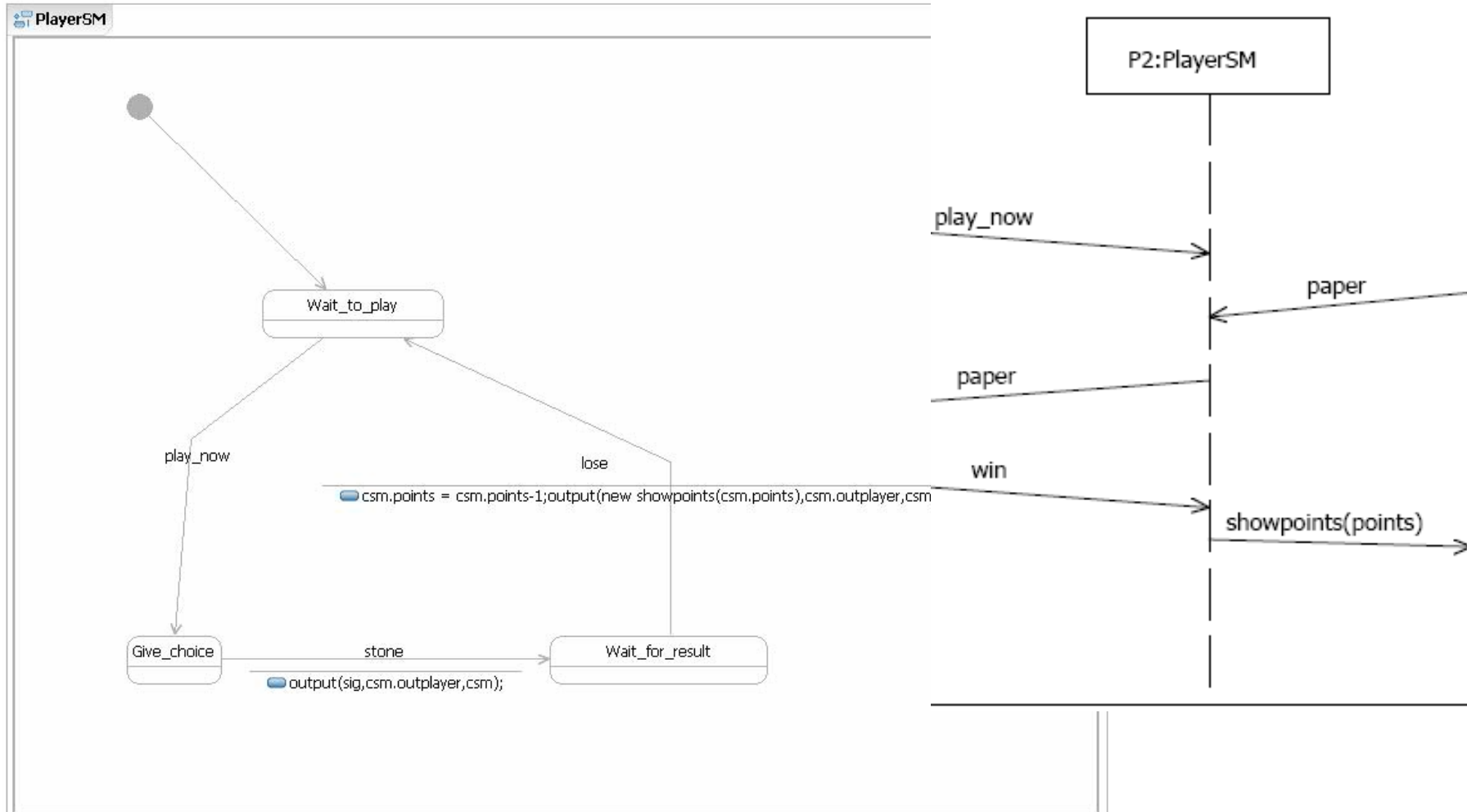
# Player: first attempt



INF 5150



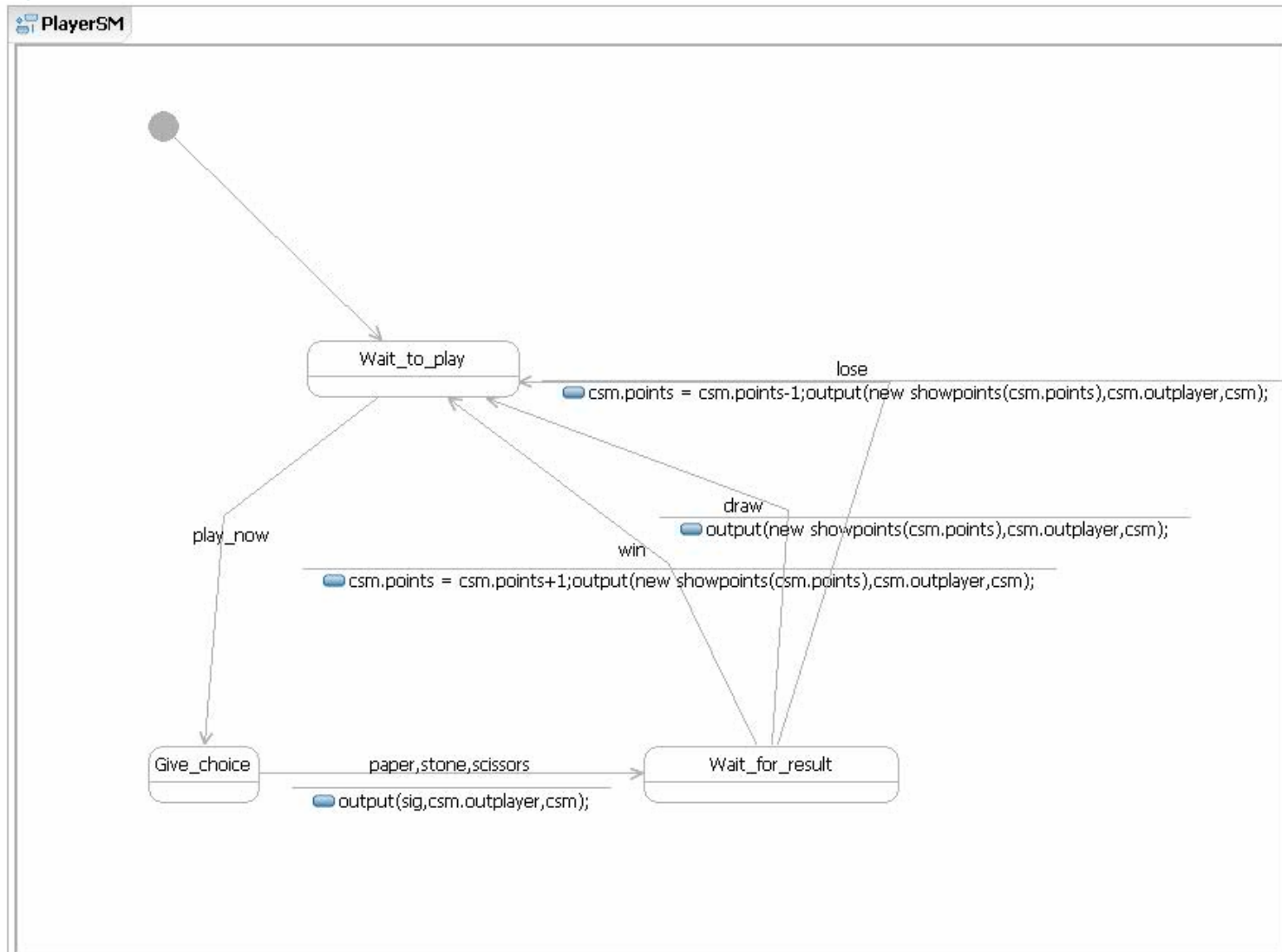
# Player: why it is not sufficient





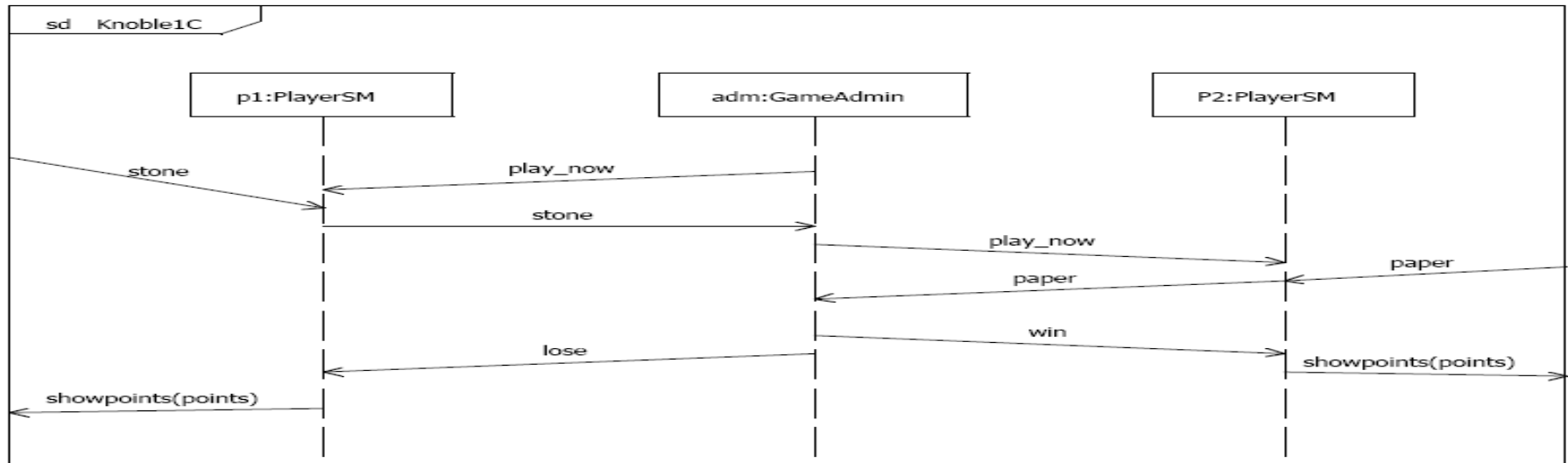
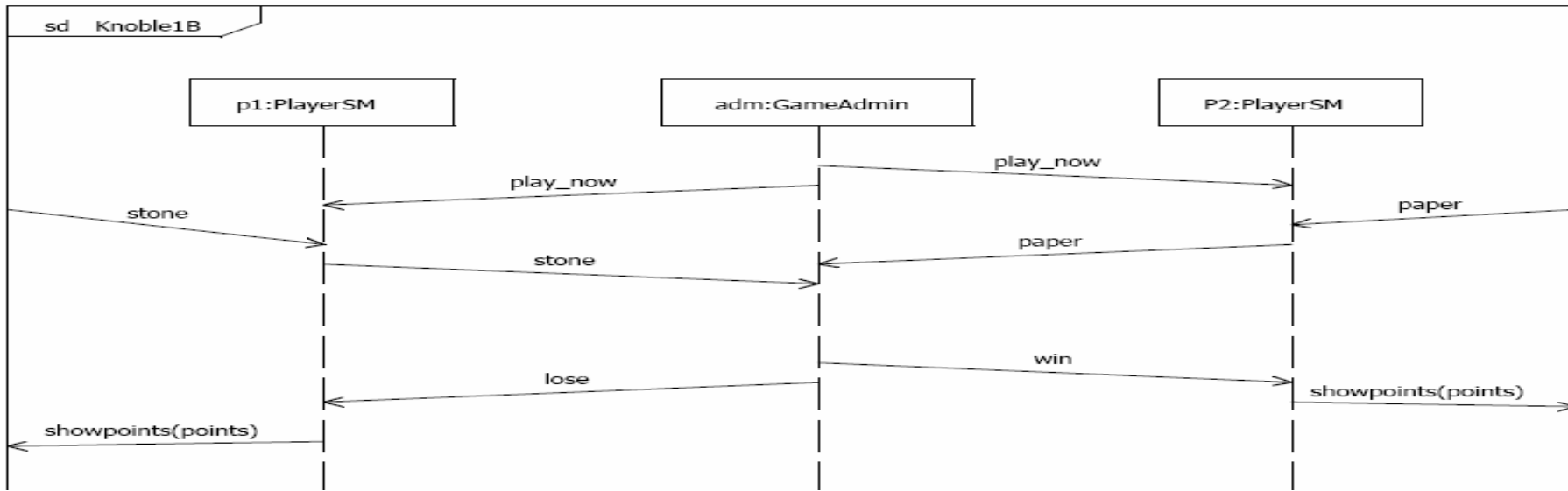


# Player: second solution



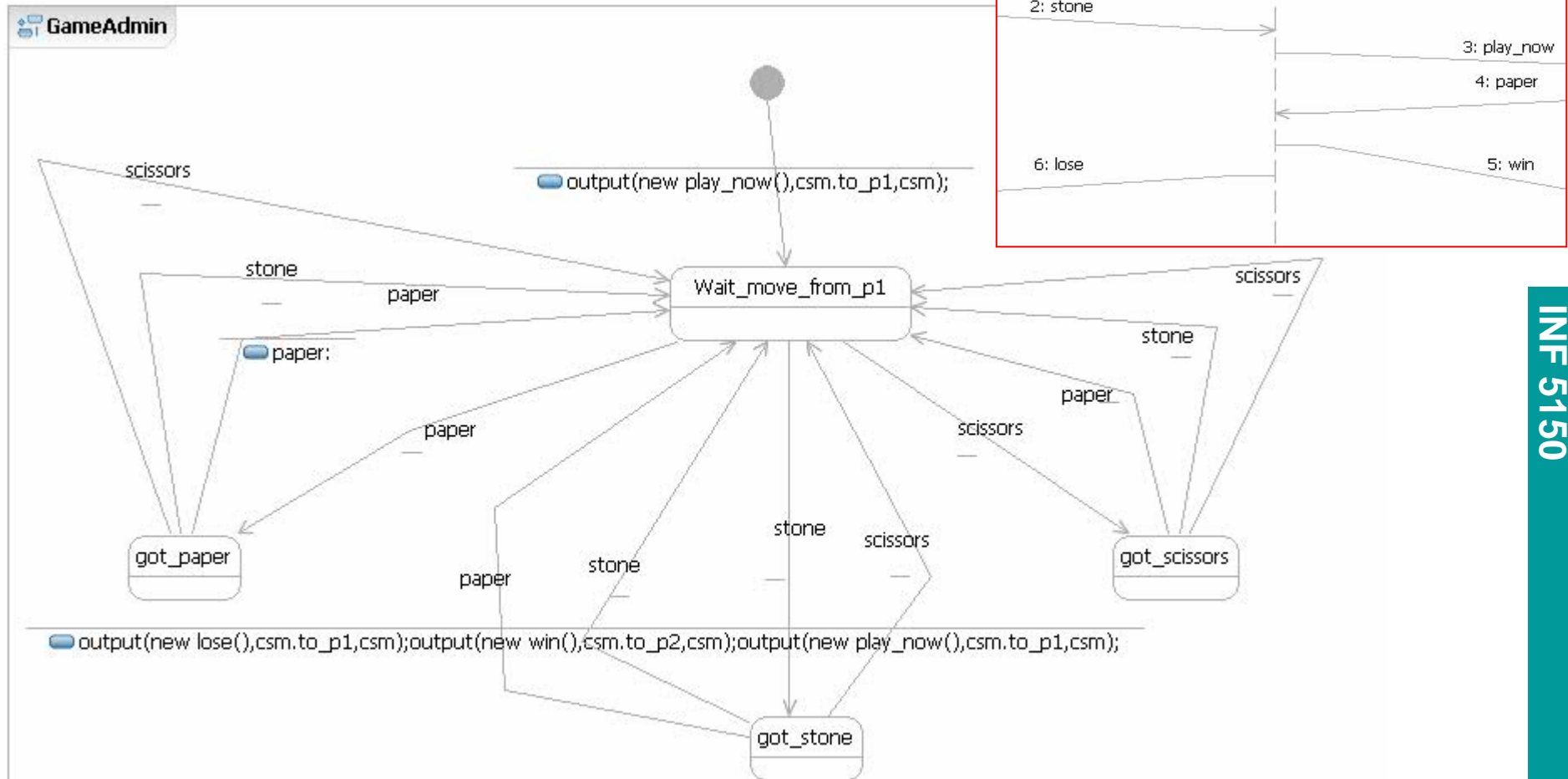


# GameAdmin: are these diagrams acceptable?





# GameAdmin: first attempt





# Demo Knoble2 (running JFTrace)

**PlayerSMGUI**

Select Input Mediator: inplayer

Select Input Message: paper, scissors, stone

Parameters:

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

Send  
Exit

**PlayerSMGUI**

Select Input Mediator: inplayer

Select Input Message: paper, scissors, stone

Parameters:

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

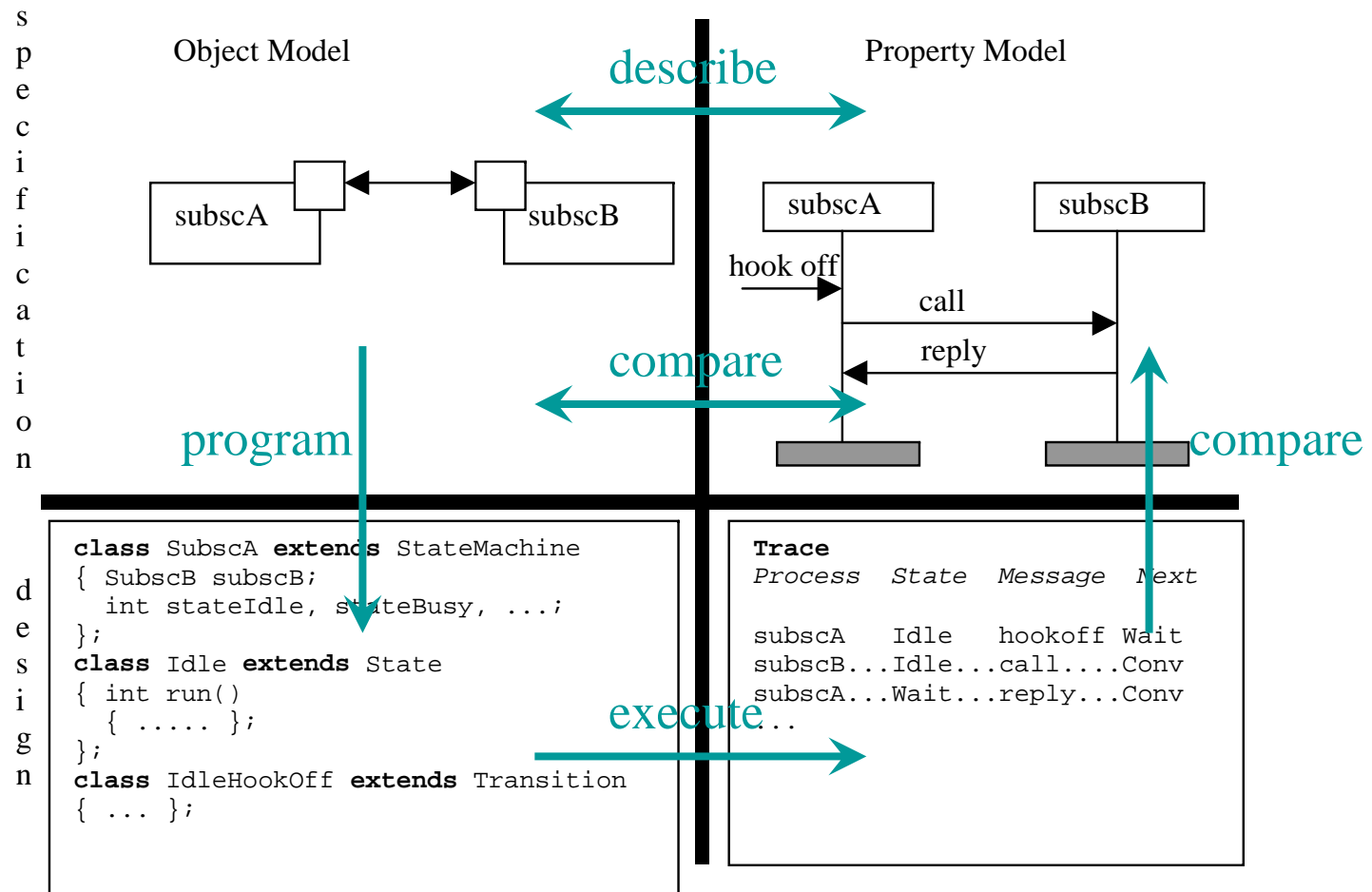
Send  
Exit

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

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





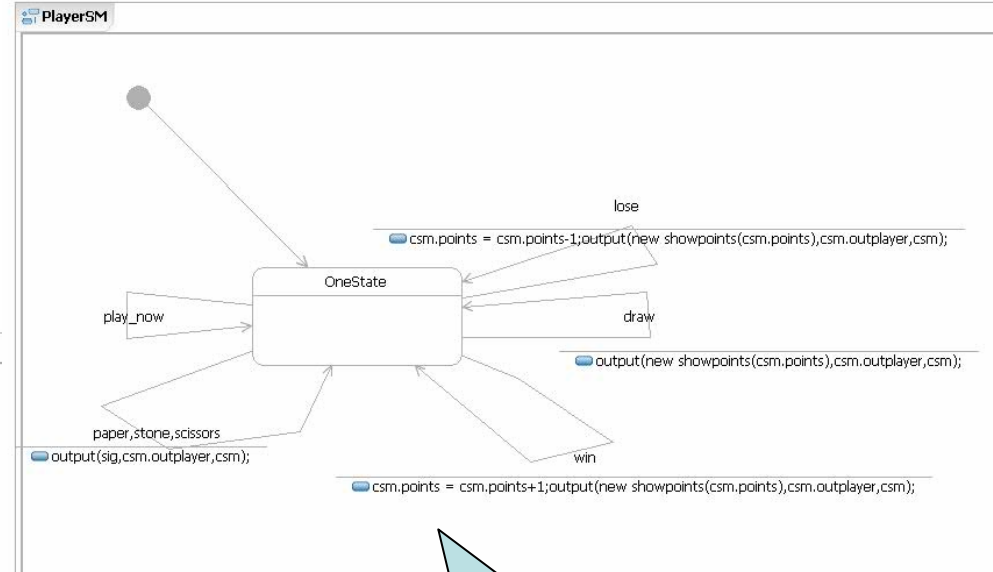
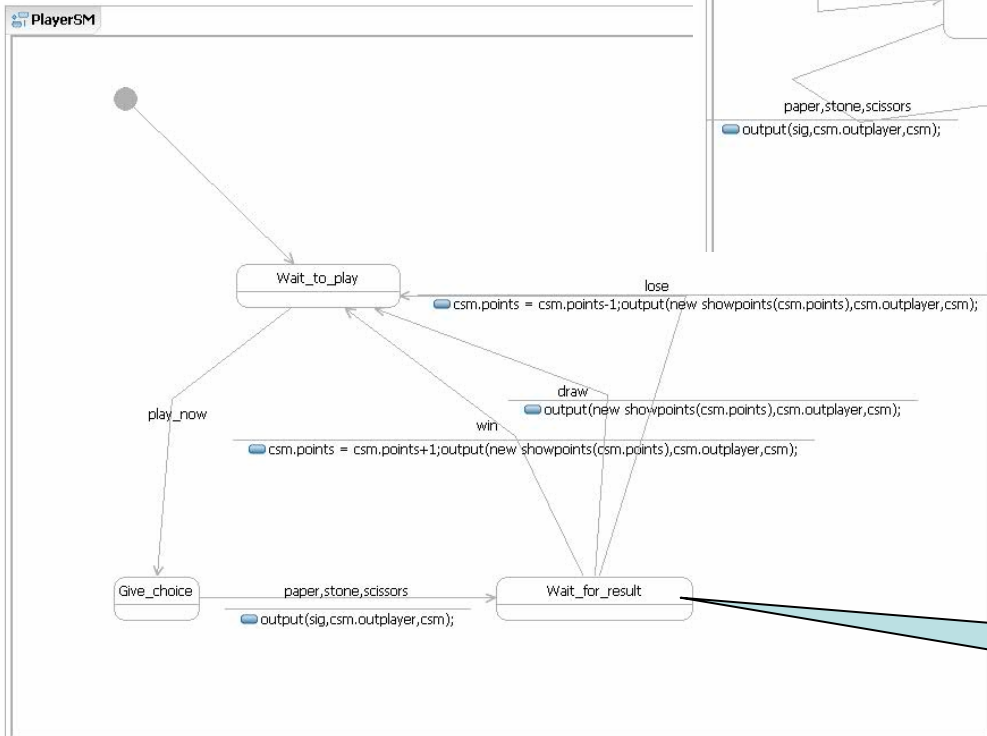
## State Machines: unassailability?

- 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!



this state machine does not detect sequence errors!

what if 'play\_now' is received here?





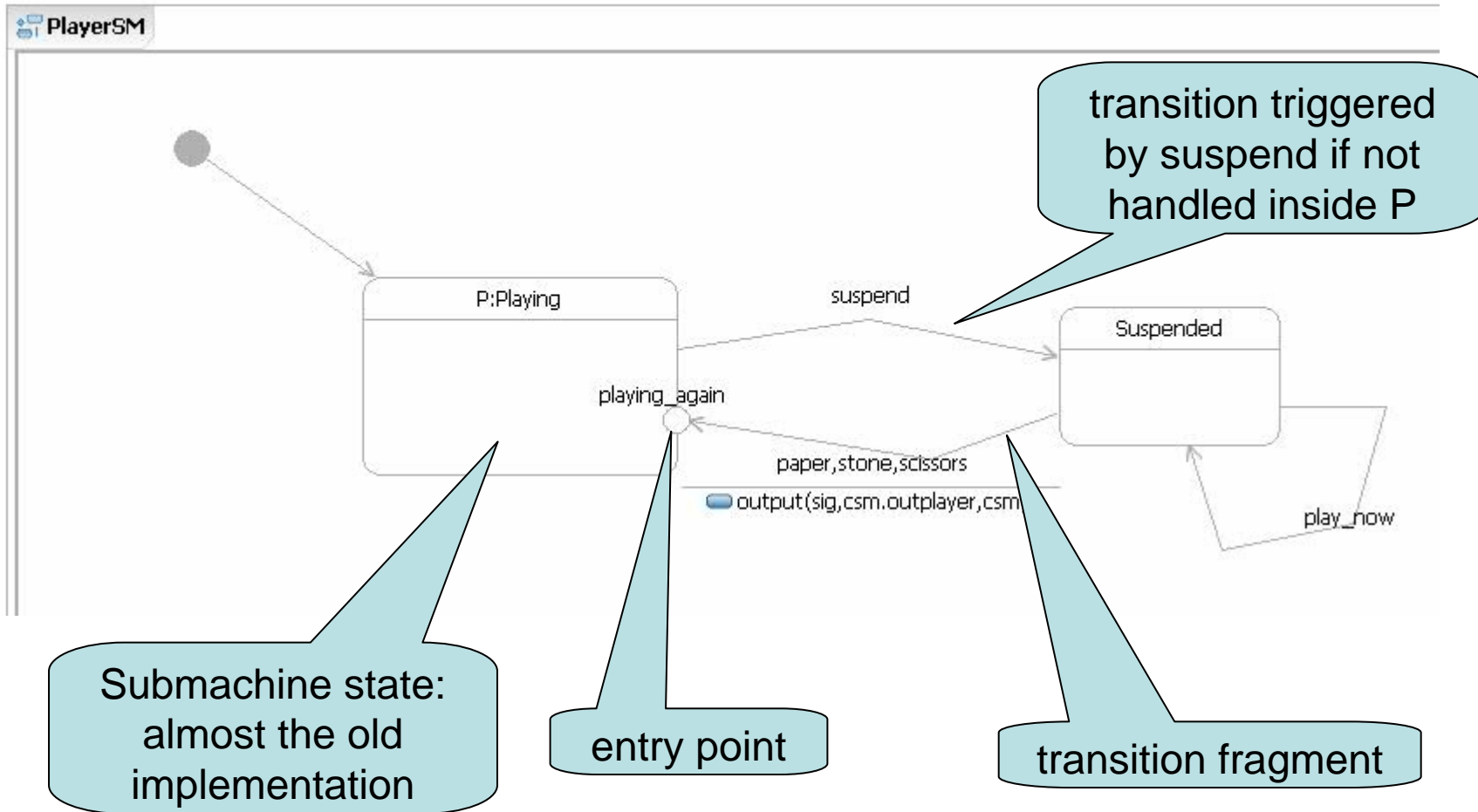
## Knoble: Now we add another requirement

- 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

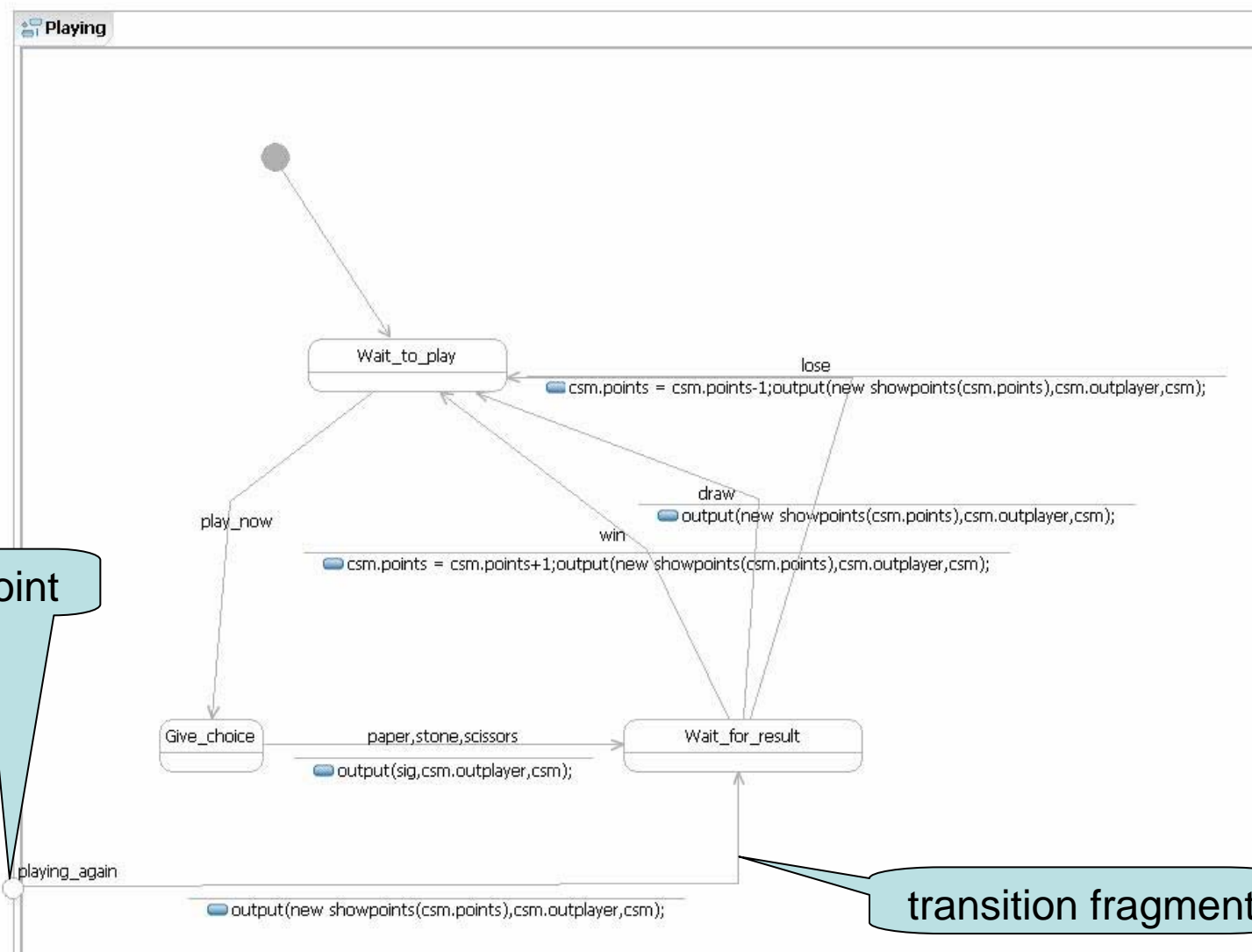




# PlayerSM: Introducing Submachine states



# Playing: almost like the old Player with entry



entry point

transition fragment



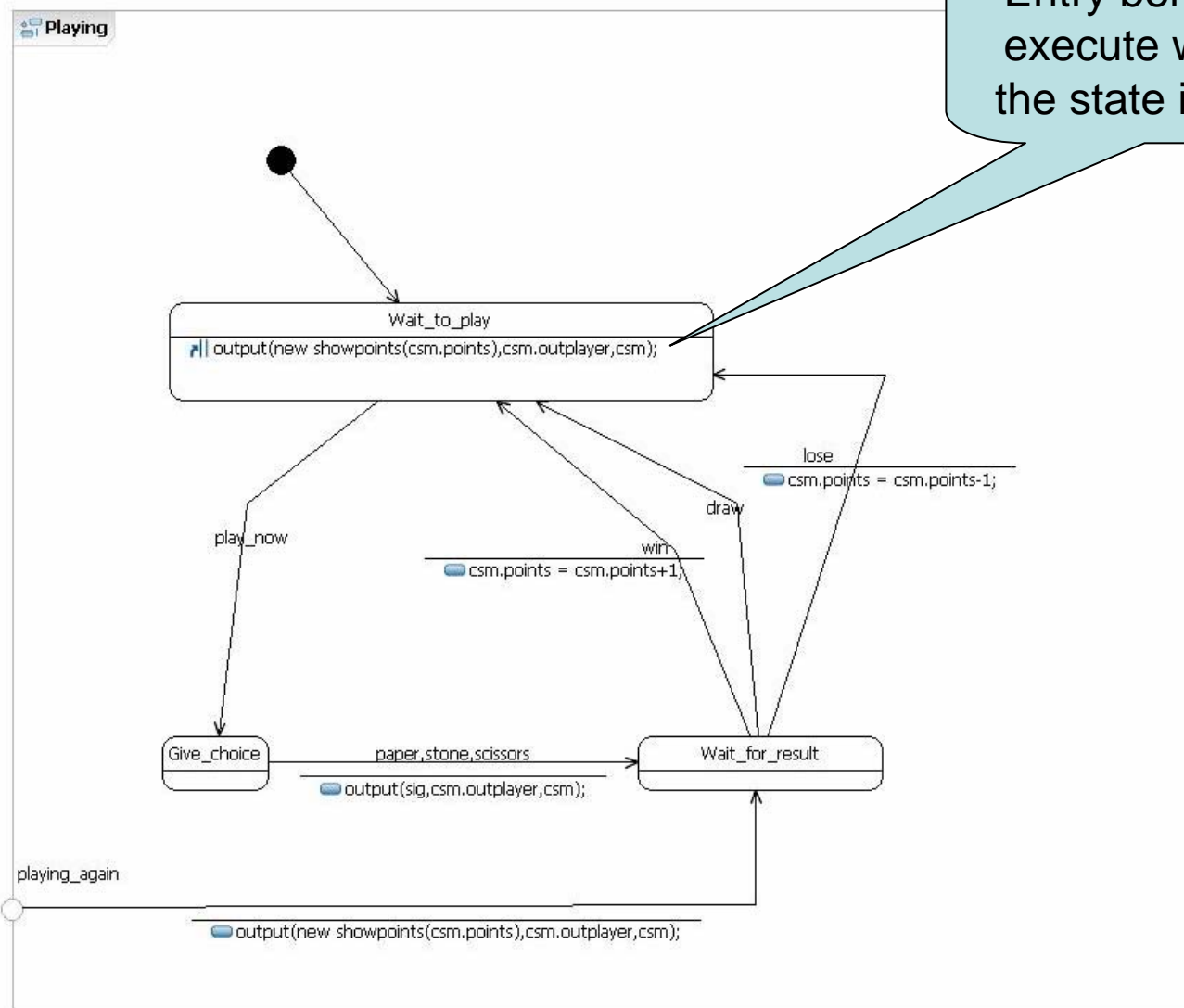


# Demo Knoble4 (not running JFTrace)

The image displays two side-by-side screenshots of the PlayerSMGUI application. Both windows have a blue title bar with the text 'PlayerSMGUI'. Each window is divided into three main sections: 'Select Input Mediator', 'Select Input Message', and 'Parameters'. The 'Select Input Mediator' section in both windows has 'inplayer' selected. The 'Select Input Message' section in the left window has 'stone' selected, while in the right window, 'paper' is selected. Below these sections is a 'Parameters' section, which is currently empty in both. At the bottom of each window is a text area for logs. The left window's log shows a sequence of messages: '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)', and 'Output from p1outplayer: showpoints@35727877(-2)'. The right window's log shows: 'Input stone to inplayer', 'Output from p2outplayer: stone@727c3874', 'Output from p2outplayer: showpoints@58fb874(1)', 'Input paper to inplayer', 'Output from p2outplayer: paper@7be37877', and 'Output from p2outplayer: showpoints@4828b877(2)'. The right window also features 'Send' and 'Exit' buttons on the right side.



# Entry and Exit behaviors



Entry behavior will execute whenever the state is entered





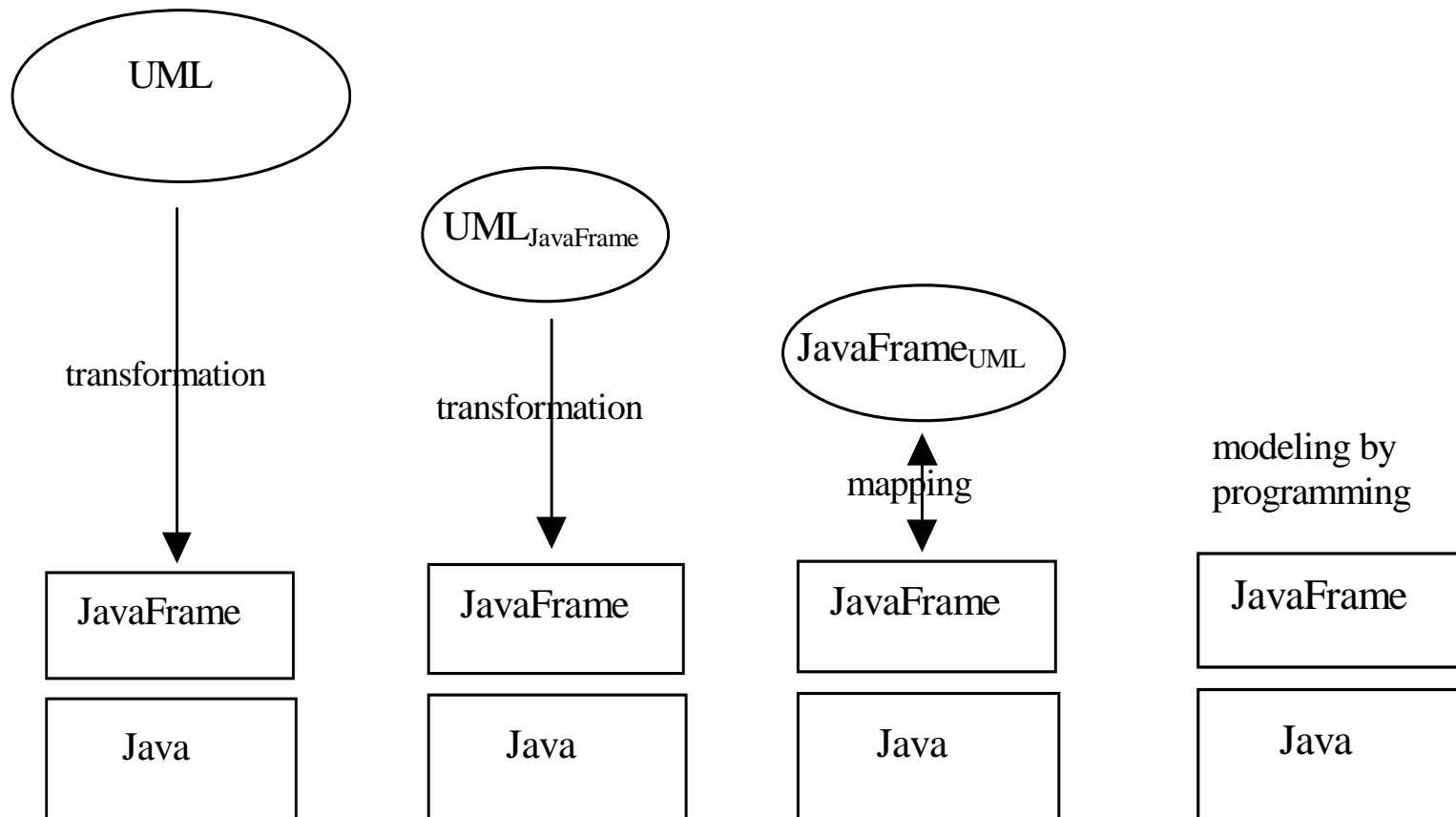
# Summary State Machines

- 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)



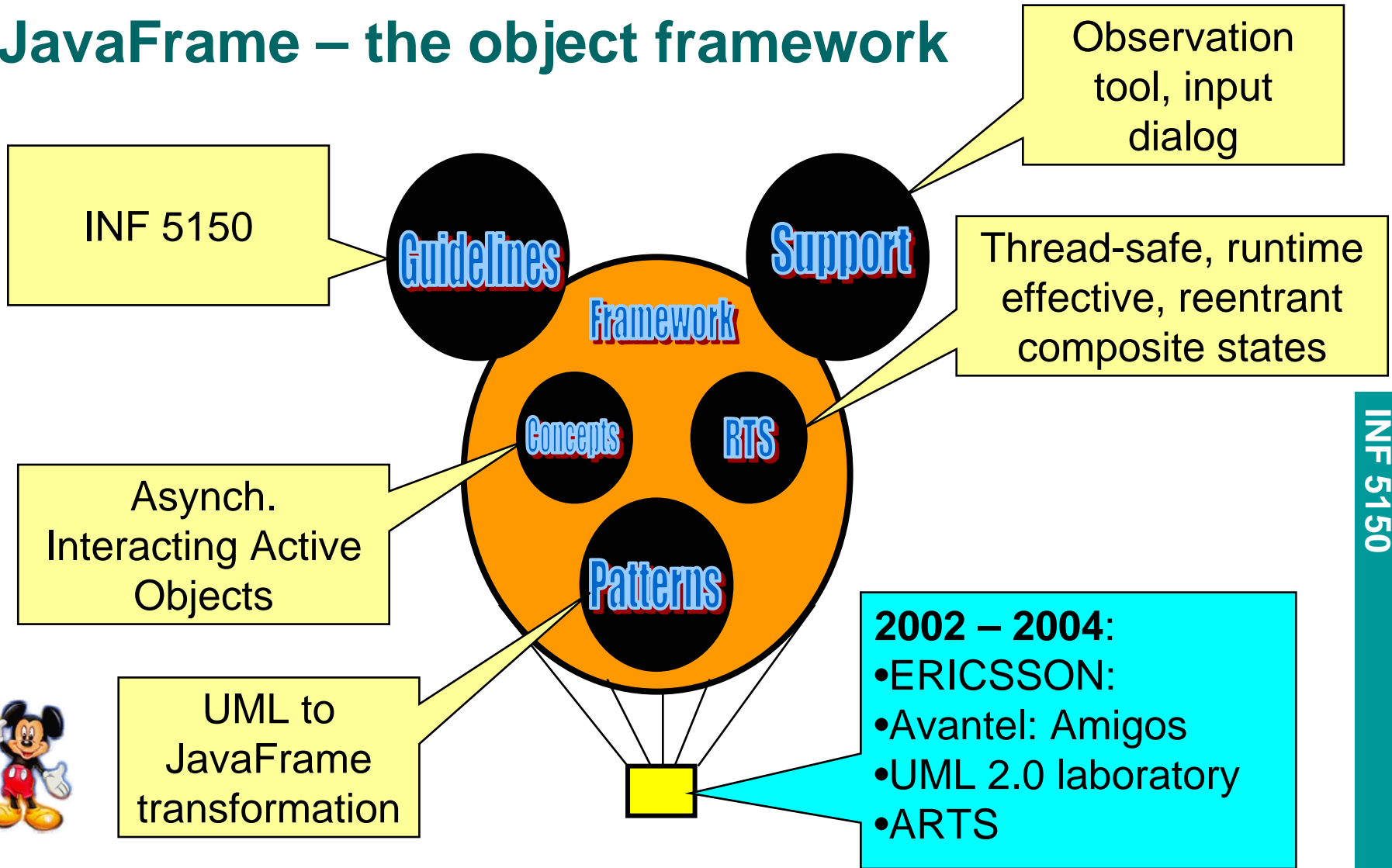


# UML and Java: JavaFrame - the solution





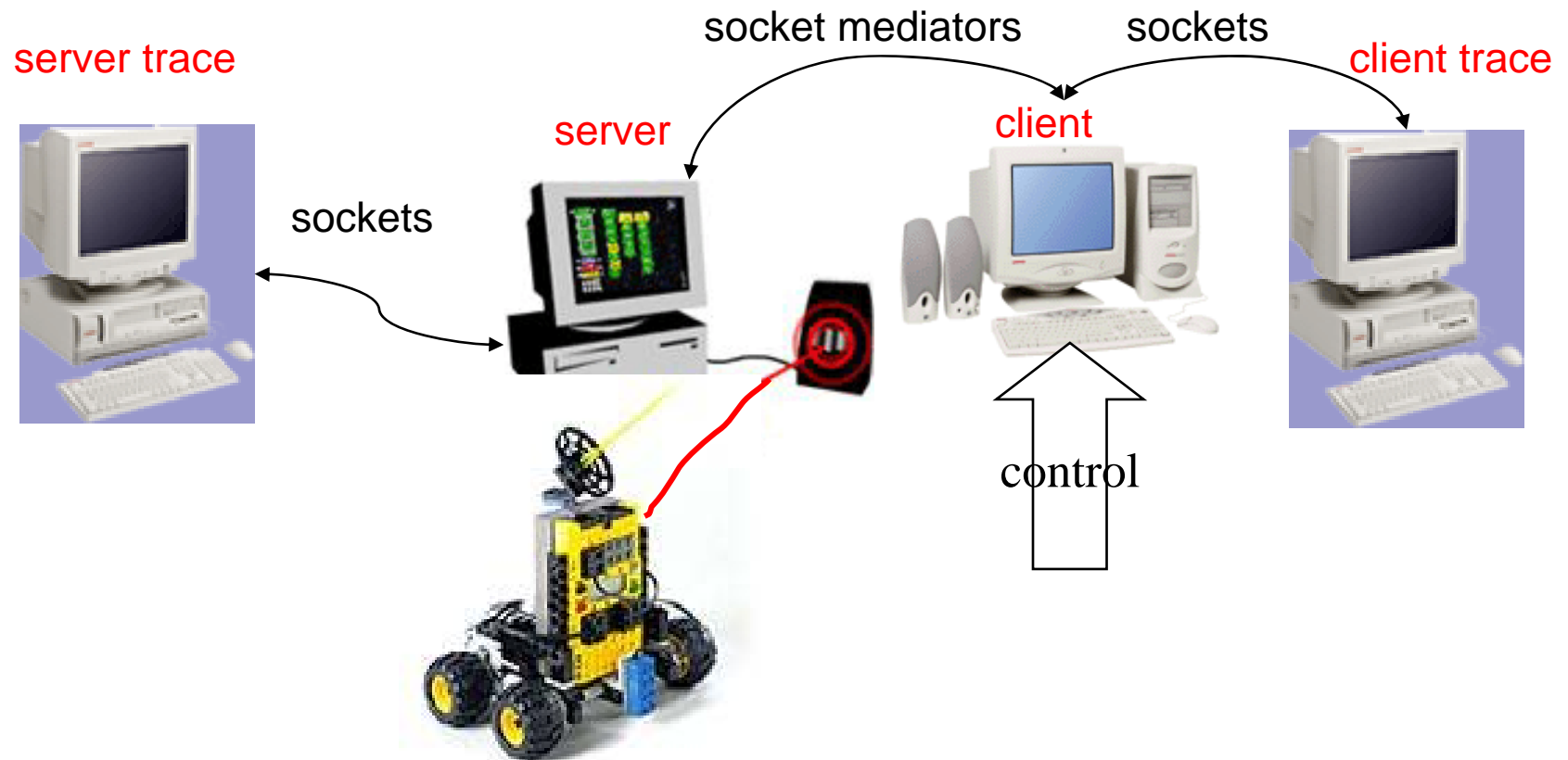
# JavaFrame – the object framework



INF 5150



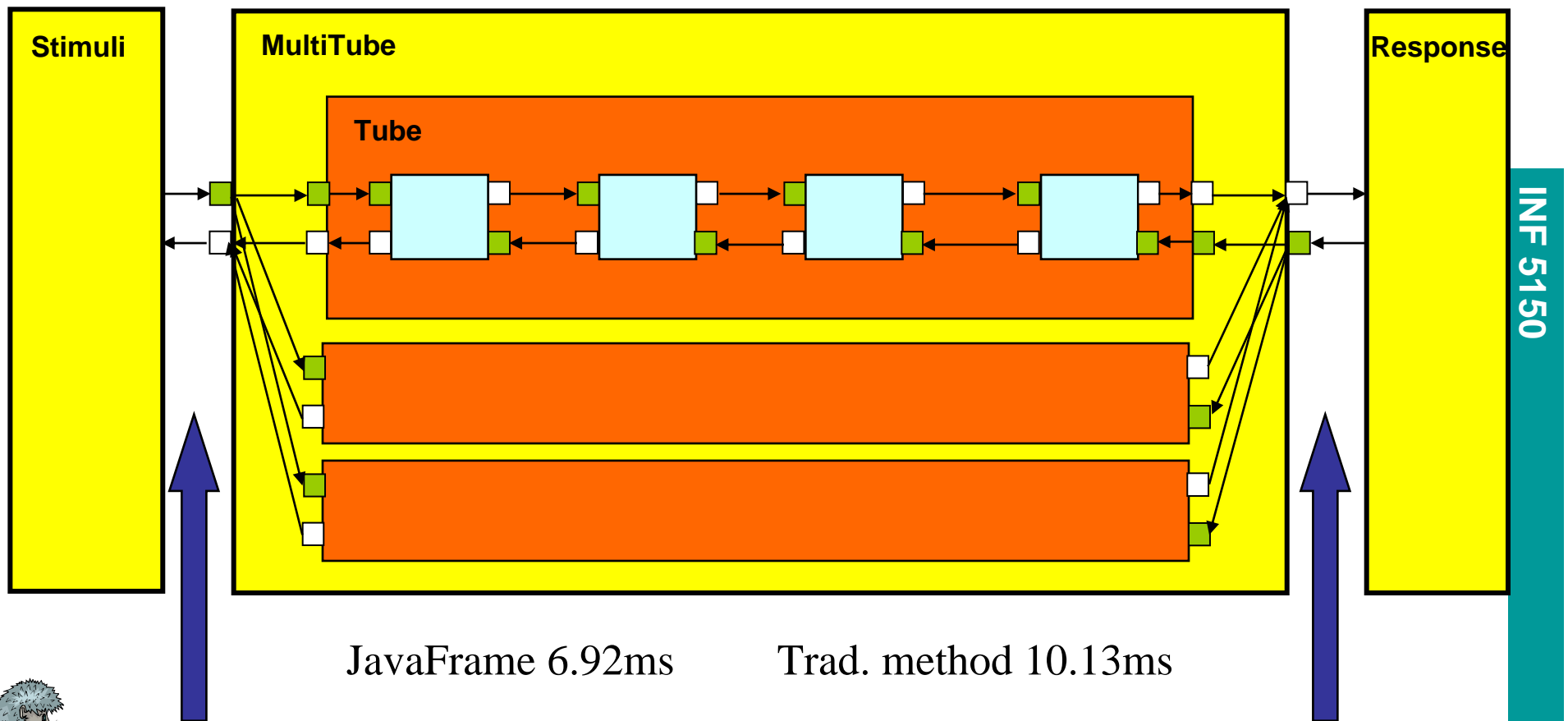
# Experiences - The Lego Mindstorm experiment







# Experiences - The Performance Model



INF 5150

