GAME OF THE ROPE*

One of the most popular traditional games is the *Game of the Rope*. Two teams of contestants face each other on a playground trying to decide which one is the strongest by pulling at opposite ends of a rope. If both teams are equally strong, the rope does not move, a standstill occurs and there is a draw; if however one of the teams is stronger than the other, the rope moves in the direction of the stronger team so much faster as the strength difference is larger and this team wins.

A variation of this game will be assumed here. A match is composed of three games and each game may take up to six trials. A game win is declared by asserting the position of a mark placed at the middle of the rope after six trials. The game may end sooner if the produced shift is greater or equal to four length units. We say in this case that the victory was won by knock out, otherwise, it will be a victory by points.

A team has five elements, but only three compete at each trial. Member selection for the trial is carried out by the team's coach. He decides who will join for next trial according to some predefined strategy. Each contestant will loose one unit of strength when he is pulling the rope and will gain one unit when he is seating at the bench. Somehow the coach perceives the physical state of each team member and may use this information to substantiate his decision.

In order to ensure rules compliance, there is a referee. She has full control of the procedure and decides when to start a new game, or a trial within the game. She also decides when a game is over and declares who has won a game or the match.

Write a simulation of the life cycle of the referee, of the teams' coaches and of the contestants using one of the models for process (*thread*) communication and synchronization which have been studied: monitors or semaphores and shared memory.

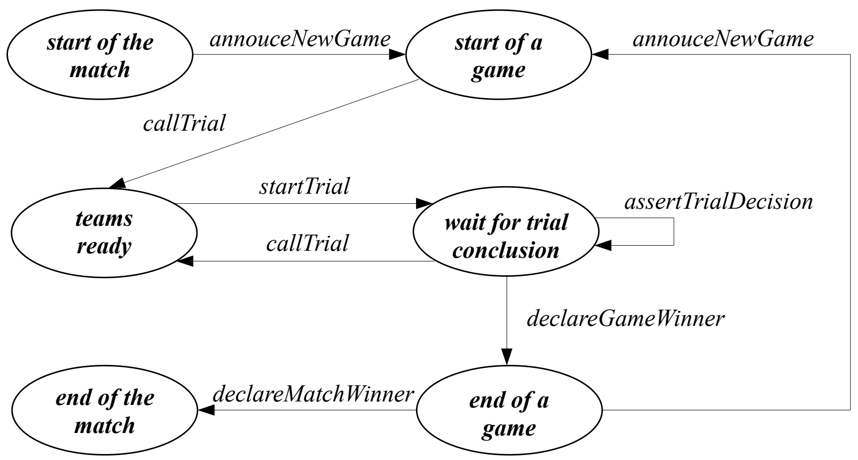
One aims for a distributed solution with multiple information sharing regions that has to be written in Java, run in Linux and terminate.

A *logging* file, which describes the evolution of the internal state of the problem in a clear and precise way, must be included.

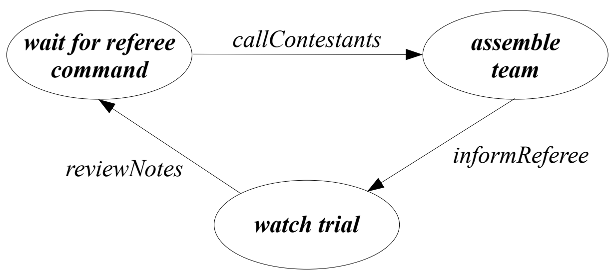
* Concept by Pedro Mariano

Suggestion to solution

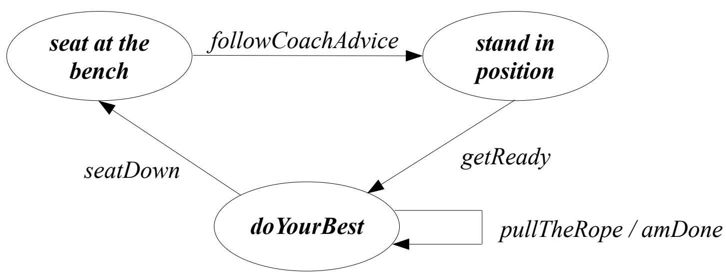
Referee life cycle



Coaches life cycle



Contestants life cycle



Characterization of the interaction

Referee

START_OF_THE_MATCH – initial state (transition)
START_OF_A_GAME – transition state
TEAMS_READY – blocking state
the referee is waken up by the last of the coaches in operation *informReferee* when the teams are ready to proceed
WAIT_FOR_TRIAL_CONCLUSION – blocking state
the referee is waken up by the last of the contestants in operation *amDone* when the trial has come to an end
END_OF_A_GAME – transition state
END_OF_THE_MATCH – final state

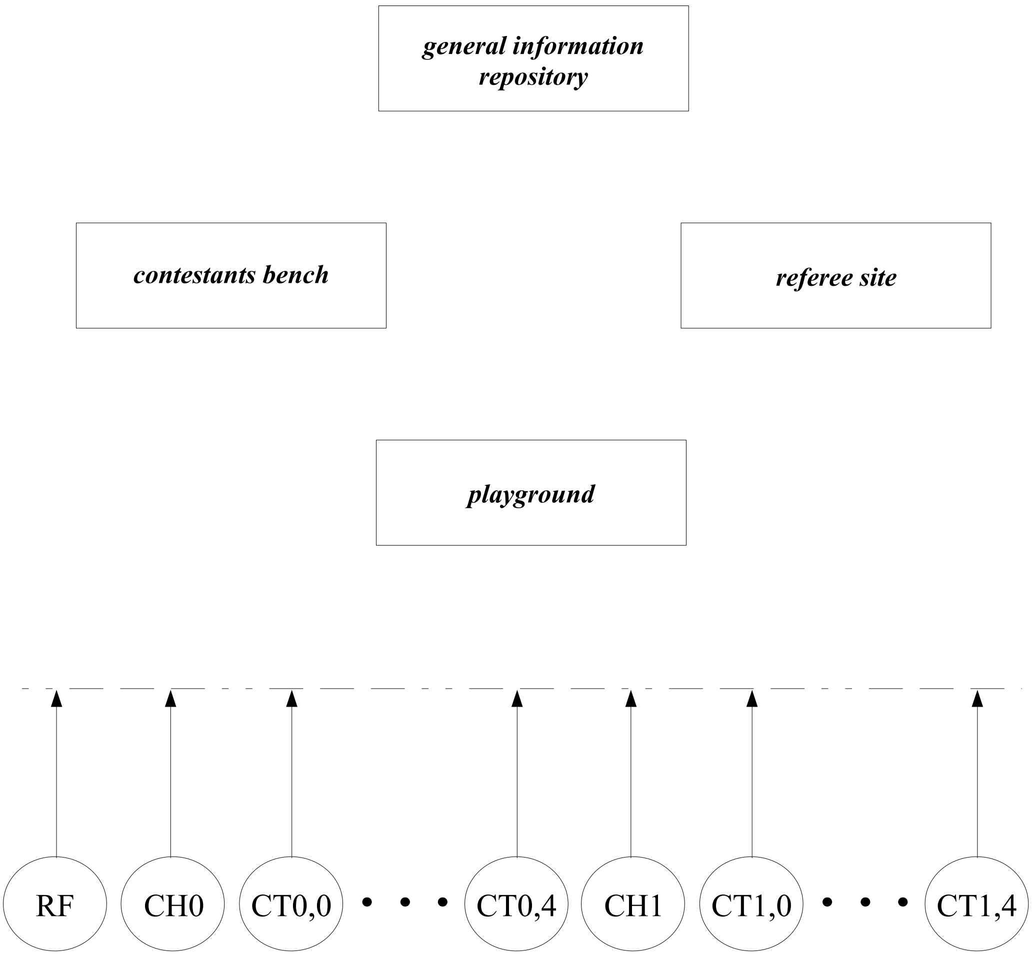
Coaches

WAIT_FOR_REFEREE_COMMAND – blocking state
the coaches are waken up by the referee in operation *callTrial* to start selecting next team
ASSEMBLE_TEAM – blocking state
the coaches are waken up in operation *followCoachAdvice* by the last of their selected contestants to stand in position
WATCH_TRIAL – blocking state
the coaches are waken up in operation *assertTrialDecision* by the referee

Contestants

SEAT_AT_THE_BENCH – blocking state
the contestants are waken up in operation *callContestants* by their coaches if they are selected to join the next trial
STAND_IN_POSITION – blocking state
the contestants are waken up in operation *startTrial* by the referee
DO_YOUR_BEST – independent state with blocking
the contestants are made to sleep for a random time interval in the simulation
they block next and are waken up in operation *assertTrialDecision* by the referee

Information sharing regions



Guidelines for solution implementation

1. Characterize interaction at the state level.

2. Specify the life cycle and internal properties of each of the *intervening entities*.

3. Specify for each *information sharing region* the internal data structure, the operations which will be invoked, identifying their signature, functionality and who is the calling entity, and the synchronization points.

4. Sketch the *interaction diagram* which describes in a compact, but precise, way the dynamics of your solution. Go back to steps 1 and 2 until you are satisfied the description is correct.

5. Proceed to its coding in Java as specific reference data types.

6. Write the application main program which should instantiate the different *information sharing regions* and the different *intervening entities*, then start the different entities and finally wait for their termination.

7. Validate your solution by taking several runs and checking for each, through the detailed inspection of the logging file, that the output data is indeed correct.