Requirements

Design and build a software system (***basicrobot***) that is able to receive via Internet commands (represented in textual form), so that:

- the commands **w | s | h | a | d** move a *differential drive robot* (***robot***) respectively **forward | backward | stop | left | right** (**req-cmd**).
- the command **step** moves the ***robot*** forward for a prefixed time (e.g. 2 secs) (**req-step**).

Moreover, the ***robot*** must be always able to react 'immediately' to the **stop** command, by stopping any ongoing movement (**req-stop**) .

Requirement analysis

The robot must be able to listen for stop command while the it is executing the step routine.

```
System steprobot

Dispatch cmd    : cmd(X)   // req-cmd
Dispatch step   : step(T)  // req-step
Dispatch stop   : stop(X)  // req-stop

Context ctxMind   ip [ host= "localhost"   port= 8023 ]
Context ctxBasicRobot  ip [ host= "localhost"   port= 8020 ]
ExternalQActor robot context ctxRealRobot

QActor stepper context ctxMind {
    State s0 initial {
        println("init")
    }
    Goto idle

    State idle {
        println("idle")
    }
    Transition tWork
        whenMsg step -> sStep
        whenMsg stop -> sStopStep
        whenMsg cmd -> sCmd

    ... // req-cmd

    // REQUIREMENT: req-step
    // req-step problem complexity demands further analysis
    // not suitable to the Requirement Analysis
    State sStep {
        println("sStep")
        forward robot -m step : step(X)
    }
    Goto idle

    // REQUIREMENT: req-stop
    State sStopStep {
        println("sStopStep")
        forward robot -m cmd : cmd(h)
    }
    Goto idle
}
```

Problem analysis

The req-step problematic brings an intrinsic proactive behavior. The robot must be able to react to events by means of a ***timer***; and use the event to stop the step routine.

```
System stepper

Dispatch cmd    : cmd(X)   // req-cmd
```

```
Dispatch step   : step(T)  // req-step
Dispatch stop   : stop(X)  // req-stop

Context ctxMind ip [ host= "localhost"   port= 8023 ]
Context ctxBasicRobot  ip [ host= "localhost"   port= 8020 ]
ExternalQActor basicrobot context ctxBasicRobot

QActor robotmind context stepper {
    ["var StepTime = 0L;"]

    State s0 initial {
        println("init")
    }
    Goto idle

    State idle {
        println("idle")
    }
    Transition tWork
        whenMsg step -> sStep
        whenMsg cmd -> sCmd

    ... // req-cmd

    // REQUIREMENT: req-step
    State sStep {
        println("sStep")
        onMsg (step : step(T)){
            ["StepTime = payloadArg(0).toLong()"]
            forward basicrobot -m cmd : cmd(w)
        }
    }
    Transition tStop
        whenTimeVar StepTime -> sEndStep
        whenMsg stop -> sEndStep

    // REQUIREMENT: req-stop
    State sEndStep {
        println("sEndStep")
        forward basicrobot -m cmd : cmd(h)
    }
    Goto idle
}
```

Deployment

Build both BasicRobot and RobotMind into deployable zip files with the commands: gradle -b build_ctxMind.gradle distZip gradle -b build_ctxBasicRobot.gradle distZip Copy any *.pl file into the bin sub directory and execute the executable scripts.

By Michele Proverbio email: michele.proverbio@studio.unibo.it