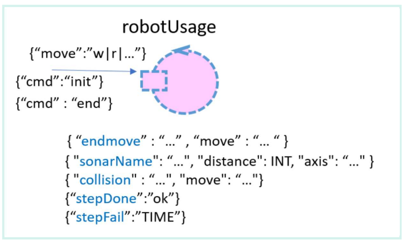
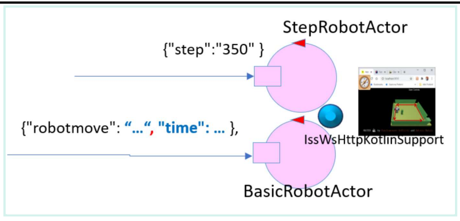
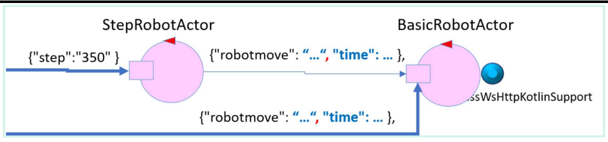
LabIss2021

Divide et impera?

As usully happens in oop, the Single Responsibility Principle could induce the software designer to distribute different behaviours in different actors.
An accurate analysis is required to consider the effects of different architectural configurations.
Let us consider here what could happen if we want to introduce a StepRobotActor, as discussed in the project phase of cautiousExplorerActors.html.

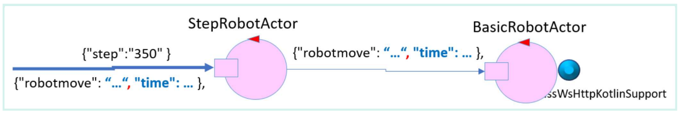
> The main feature of a StepRobotActor is related to its capability of executing a **step** in 'atomic' way (i.e. all or nothing).
> A step is done when distance traveled by the robot is equal to the diameter of the circle in which it is supposed to be inscribed.

Step+Basic: case 0
 robotUsage
{"move":"w|r|..."}
{"cmd":"init"}
{"cmd" : "end"}
{ "endmove" : "..." , "move" : "... " }
{ "sonarName": "...", "distance": INT, "axis": "..." }
{ "collision" : "...", "move": "..."}
{"stepDone":"ok"}
{"stepFail":"TIME"}
	 StepRobotActor
	{"step":"350" }
	{"robotmove": "...", "time": ... },
	IssWsHttpKotlinSupport
	BasicRobotActor
	• **pros:** greater atomicity
	• **cons:** each actor has a single responsibility, so you may need to have a large number of actors. Each actor has a specific input format, which you have to remember.
	• **caveat:** the wenv is controlled by two entities

Step+Basic: case 1 • **pros:** the support is controlled by a single actor, you avoid competition in sending or receiving messages. • **cons:** stepRobotActor must always pass through the BasicRobotActor,causing delay in sending to wenv. Each actor has a single responsibility, so many actors are required for several operations.	 StepRobotActor BasicRobotActor {"step":"350" } {"robotmove": "...", "time": ... }, {"robotmove": "...", "time": ... }, IssWsHttpKotlinSupport

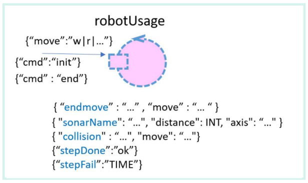
- **caveat:** The basicRobotActor receives messages from two distinct entities, this may cause confusion in the order of arrival

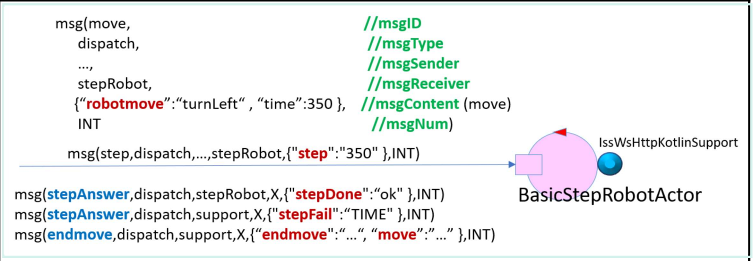
Step+Basic: case 2



- **pros:** the support is controlled by a single actor. Only the stepRobotActor sends messages to the BasicRobotActor.
- **cons:** To communicate with the BasicRobotActor you must always pass through the StepRobotActor, that has to translate all massagges. we have two different types of message.
- **caveat:**

Step+Basic: case 3


robotUsage
{"move":"w|r|…"}
{"cmd":"init"}
{"cmd" : "end"}
{ "endmove" : "…" , "move" : "… " }
{ "sonarName": "…", "distance": INT, "axis": "…" }
{ "collision" : "…", "move": "…"}
{"stepDone":"ok"}
{"stepFail":"TIME"}



```
msg(move,                                     //msgID
    dispatch,                                 //msgType
    ...,                                      //msgSender
    stepRobot,                                //msgReceiver
   {"robotmove":"turnLeft" , "time":350 },    //msgContent (move)
    INT                                       //msgNum)
  msg(step,dispatch,...,stepRobot,{"step":"350" },INT)

msg(stepAnswer,dispatch,stepRobot,X,{"stepDone":"ok" },INT)
msg(stepAnswer,dispatch,support,X,{"stepFail":"TIME" },INT)
msg(endmove,dispatch,support,X,{"endmove":"...", "move":"..." },INT)
```

- **pros:** we delete the StepRobotActor, eliminating the additional translation phase. Only one actor sends and receives different types of messages from support. Unique messagge structure.
- **cons:** you no longer have single-responsibility classes. BasicStepRobotActor contains all the features.
- **caveat:**

Problem analysis

Using BasicStepRobotActor, the following problems were detected:

- The construction of the stepper forces the robot to execute a "moveBackward" command when it encounters an obstacle.
- The BasicStepRobotActor does not keep track of the moveBackward in execution, the robot will then send new moves to the support receiving a "not allowed" as a response,

as in support it is still executing the "moveBackward" command.

Solutions:

- To work around the first problem, one could intercept the execution time of the "moveBackward" command and execute the "moveForward" command for the same time.

Deployment

A first prorotype with limited capabilities is available here. Run the BoundaryWalkRobotActorCaller.kt file.

By Lucrezia Corsaro lucrezia.corsaro@studio.unibo.it https://github.com/LucreziaCorsaro