Basic robot 24

Ci sono diverse componenti che possono aiutare il robot a fare mosse e gestire le cose:

- Engager → manda una richiesta di engage/disangage, se il basicrobot libero la richiesta è accettata (engagedone), sennò rifiutata (engagefailed) [Nel nostro caso potrebbe essere fatto da cargorobot]
- Planexec → manda richiesta di eseguire una serie di mosse di fila al basicrobot con doplan, se il Basicrobot le fa tutte avremo doplandone, se si interrompe (trova ostacolo o evento alarm) allora blocca l'esecuzione [più avanti verrà detto che l'argomento di doplanfailed è la sequenza che ha già fatto, quindi IDEA potremmo salvare la sequenza chiesta in cargorobot e da ciò che dice dostepfailed capire cosa resta e usarla come nuovo piano]
- Robotpos → gestisce richieste di posizionamento e con planner determina il piano di movimento, questo viene eseguito con planexec, se va a buon fine moverobotdone sennò se l'esecuzione del piano fallisce o viene interrotta moverobotfailed [onestamente non ho ben capito che dovrebbe fare]
- Basicrobot → gestisce comandi elementari tramite cmd e richieste di movimento con step, se hanno successo stepdone sennò stepfailed

BR24 messaggi

```
System BR24

Dispatch cmd        : cmd(MOVE)      //MOVE=w|s|d|a|r|l|h
Dispatch end        : end(ARG)

Request step       : step(TIME)
Reply stepdone     : stepdone(V)
Reply stepfailed   : stepfailed(DURATION, CAUSE)

Event  sonardata   : sonar( DISTANCE )        //percepito da sonarobs/engager
Event obstacle     : obstacle(X)
Event info         : info(X)

Request  doplan    : doplan( PATH, STEPTIME )
Reply doplandone   : doplandone( ARG )
Reply doplanfailed  : doplanfailed( ARG )

Dispatch setrobotstate: setpos(X,Y,D)
Dispatch setdirection : dir( D )  //D =up|down!left|right

Request engage        : engage(CALLER)
Reply   engagedone    : engagedone(ARG)
Reply   engagerefused : engagerefused(ARG)

Dispatch disengage    : disengage(ARG)

Event alarm           : alarm(X)
Dispatch nextmove     : nextmove(M)
Dispatch nomoremove   : nomoremove(M)

//Inglobamento endosimbitico di robotpos
Request moverobot    :  moverobot(TARGETX, TARGETY)
Reply moverobotdone  :  moverobotok(ARG)
Reply moverobotfailed:  moverobotfailed(PLANDONE, PLANTODO)
```

N.B. se cargorobot si deve fermare per il guasto al sonar per fr fermare basicrobot bisogna usare l'evento alarm

Esecuzione di un piano

Planexec è l'attore adibito a far eseguire un piano ovvero una sequenza di mosse (in **forma verbosa** (ad esempio **[w, w, l, w, w]**) oppure in **forma compatta** (ad esempio **wwlww**))

Abbiamo una richiesta di fare fare un plan:
Request doplan:doplan(PLAN,OWNER,STEPTIME)

Dove:
- Plan → sequenza di mosse
- Owner → nome attore chiamante Basic robot [Nel nostro caso direi cargorobot]
- Steptime → tempo di esecuzione dello step di basicrobot

Se la request ha esito:
- Positivo → doplandone dove argomento non è significativo
- Negativo → doplanfailed dove argomento è la porzione di plan che è stata eseguita prima del fallimento

[Se facessimo il doplan e fallisse ci darebbe doplanfailed dove argomento è il pezzo che è stato fatto, se cargorobot conoscesse il plan completo poi saprebbe cosa manca e a quel punto quando le attività riprenderebbero si potrebbe dare il nuovo plane]

Linguaggio aril per le mosse del basicrobot

Poichè l'uso di un robot virtuale è **solo un passo intermedio** verso un robot fisico, che potrebbe avere un linguaggio di comando diverso, può essere opportuno introdurre un linguaggio di comando 'technology-independent', che denominiamo **linguaggio aril** (**Abstract Robot Interaction Lanaguage**) con cui esprimere i comandi-base di spostamento con la sintassi che segue:

```
MOVE = w | s | l | a | r | d| h | p

w : muove avanti per 2500 msec
s: muove indietro per 2500 msec
l,a:  gira a sinistra di 90 per 300 msec
r,d:  gira a destra di 90 per 300 msec
p: step asincrono per 370 msec
```