Ingegneria dei Sistemi Software
Approfondimento
Terza Parte

Beatrice Mezzapesa, Alessia Papini, Lorenzo Pontellini

Alma Mater Studiorum – University of Bologna
via Venezia 52, 47023 Cesena, Italy
{beatrice.mezzapesa, alessia.papini, lorenzo.pontellini}@studio.unibo.it

1 Problem Analisys and Abstraction Gap

Riproponiamo alcuni concetti relativi all'analisi del problema e alla definizione e identificazione dell'abstraction gap presentati nella precedente relazione utili contestualizzare questo lavoro:

> *"Visto il contesto nel quale ci si pone, le problematiche identificate hanno portato, come citato all'interno dell'analisi dei requisiti, alla definizione di una serie di concetti per le varie modalità di esecuzione delle azioni. Occorre identificare una modalità che ci permetta di astrarre dallo specifico problema in gioco (ovvero quello dei robot) e che ci consenta di definire una soluzione generale alla problematica, sfruttando il robot come approccio, definendo i concetti validi anche per future fasi evolutive del progetto. I requisiti ci portano a riconoscere una serie di problematiche delle quali ci andremo ad occupare: definizioni di azioni sincrone/asicrone ed azioni interrompibili.*
> *Ponendoci nel contesto di esempio, il robot deve essere sensibile ad alcuni cambiamenti che potranno avvenire nell'ambiente circostante nel quale è situato. Occorre, quindi, definire la gestione di questi cambiamenti in modo che durante l'esecuzione delle azioni previste, nel caso di rilevamento di un cambiamento, il robot possa reagire di conseguenza. In particolare si vuole fare in modo che l'azione intrapresa dal robot si blocchi e si possano eseguire azioni alternative, al termine di queste sarà necessario valutare se continuare o meno con l'esecuzione precedente che include le azioni già pianificate. Da qui in avanti viene definito* ***piano*** *come sequenza di azioni.*
> *Così facendo occorre definire, dato un robot, tutti i cambiamenti ai quali dev'essere in grado di reagire, descrivendo i comportamenti da avere in ogni situazione e le modalità di controllo che permettano di fare una valutazione sullo stato di avanzamento dell'azione corrente intrapresa dal robot. Inoltre è necessario riconoscere quando un'azione è giunta al termine in modo da controllare lo stato dell'azione ed eventualmente proseguire con la successiva.*
> *La base di partenza per noi è l'astrazione di classe* ***BaseRobot*** *il quale*

tramite apposito metodo ha la possibilità di eseguire azioni tramite l'uso di appositi attuatori, all'interno del mondo reale. Legata alla problematica identificata, sorge inoltre il problema di specificare la tipologia di azione da utilizzare all'interno del contesto in esame dato che questa, avrà effetti sul controllo del robot stesso e sul comportamento dell'architettura logica creata a valle.
Si vogliono fissare le definizioni relative alle possibili azioni utilizzate all'interno del contesto del problema appena definito:

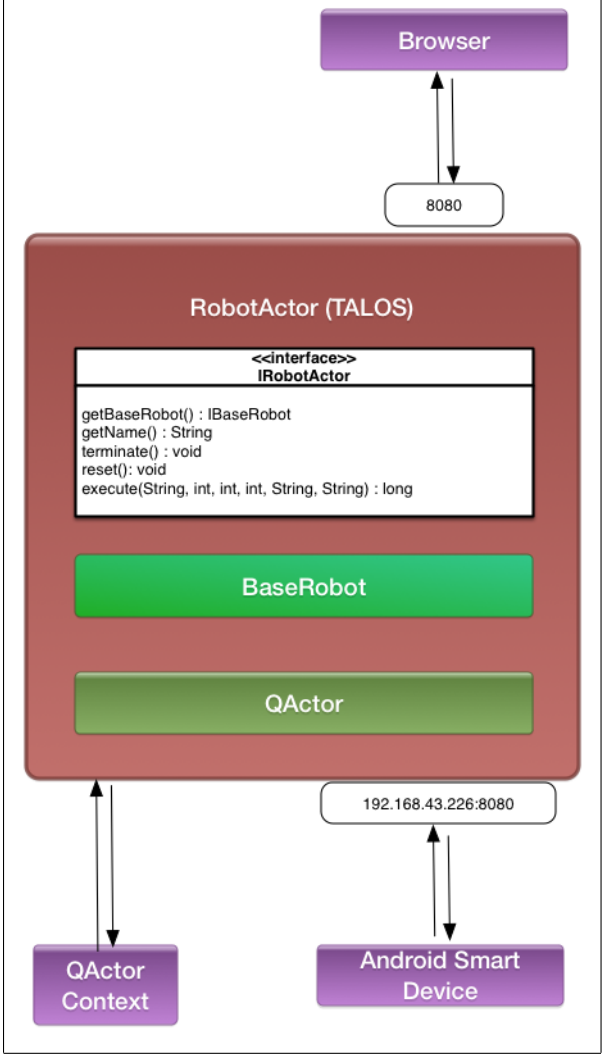
- ***Azione sincrona**: si considera un'esecuzione sincrona quando questa viene concretizzata e occorre attendere la terminazione della stessa per poter restituire il controllo al chiamante.*
- ***Azione asincrona**: si definisce un'esecuzione asincrona quando questa può essere eseguita senza che il chiamante debba attendere la fine dell'esecuzione dato che il controllo viene immediatamente restituito al chiamante.*

Rispetto alla fase precedente si deve considerare un sistema non più concentrato ma distribuito, infatti l'utilizzo della console remota rende necessaria la definizione della comunicazione tra quest'ultima e il robot. Dev'essere definito un comando di "halt" che la console sia in grado di inviare al robot e quest'ultimo dovrà essere in grado di interpretarlo.

Avendo appena definito i concetti principali nei capitoli precedenti, ci si rende conto che la base di partenza considerata, ovvero il linguaggio OO Java, non permette di esprimere ad un sufficiente livello di astrazione le tematiche sopra citate e il dislivello dal punto di vista tecnologico sarebbe troppo ampio da essere colmato in un unico step computazionale. Si procede quindi alla definizione di una serie di livelli di astrazione che permettano di arrivare gradualmente a definire le modalità di approccio al problema.
Il primo che occorre è la possibilità di una gestione tramite gli attori in quanto quest'ultimo modello computazionale, che si rifà ad uno stile proattivo/reattivo, riesce a modellare perfettamente il comportamento di un robot. Questo modello in realtà risiede già all'interno del framework computazionale che utilizzeremo (qActor) e potrà quindi essere considerato come assodato per le future implementazioni del problema. La base di partenza, risulta essere quindi il RobotActor, come già definito nella precedente relazione[1]*, esteso con la possibilità di ricevere comandi inviati dalla console remota oppure dal browser.*

[1] Ingegneria dei Sistemi Software A differential drive Robot Prima Parte

Robot Actor Model



E' stato detto in precedenza che per modellare correttamente il comportamento del robot occorre introdurre il concetto di automa a stati finiti, questo ci induce a passare dalla programmazione message passing quale Java a quella event based. Questo ci permette di definire che il cambiamento che permette di eseguire la transizione da uno stato all'altro dell'automa può essere identificato come un ***evento****. In questo modo si vuole ottenere una entità reattiva che nel momento in cui esegue un'azione possa comunque essere sensibile a determinati "cambiamenti". Infatti, supponendo di essere in uno stato in cui viene eseguita una determinata azione, il robot si mantiene reattivo e nel momento in cui riceve*

un particolare evento è in grado transitare in un determinato stato che permetta la gestione dell'evento stesso.
In particolare, con questa modellazione, si è in grado di definire un evento che viene lanciato al termine dell'esecuzione di un azione sincrona/asincrona. Ovviamente per un particolare stato possono essere definiti più eventi ai quali il robot dev'essere sensibile ed altrettante transizioni di stato che permettano di gestirli. In un contesto generale, quando viene ricevuto un evento di "halt", che provoca una transizione di stato, esso viene gestito eseguendo un piano alternativo e fermando l'azione eseguita in precedenza. Nel caso del contesto specifico questo non è necessario in quanto, l'esecuzione di una nuova azione da parte del robot, interrompe/termina automaticamente quella precedente.

2 Future Work

3 Information about the author

Alessia Papini	Beatrice Mezzapesa	Lorenzo Pontellini