Software Engineering
process report template

Andrea Torchi Francesco Giovanelli Giuseppe Tempesta

Alma Mater Studiorum – University of Bologna
via Venezia 52, 47023 Cesena, Italy
andrea.torchi@studio.unibo.it
francesco.giovanelli@studio.unibo.it
giuseppe.tempesta2@studio.unibo.it

1 Introduction

Questo modulo rappresenta una fotografia del processo di sviluppo e produttivo del software. Avere una documentazione completa circa tutte le fasi (Analisi, Progettazione, Implementazione, Testing) è importante per mantenere una traccia delle considerazioni emerse nelle fasi già citate e, soprattutto, il giusto flusso di sviluppo, che parte dai requisiti, fino ad arrivare ad un prodotto finale che è in linea con le attese del committente.
Possiamo approcciare i problemi che emergono in modo olistico (top-down), o riduzionistico (bottom-up). Cosa s'intende per top-down o bottom-up?
TOP-DOWN, cioè procedendo dall'alto verso il basso, quindi partendo da specifiche di alto livello, analizzando i sottosistemi ed arrivando ad una soluzione; questo approccio è in linea con una visione OLISTICA, ovvero che non si concentra sulle singole parti, ma ha una visione "più d'insieme" o di alto livello.
BOTTOM-UP, cioè procedendo dal basso verso l'alto, quindi partendo più da componenti, per poi passare alla loro sintesi. Questo approccio è in linea con una visione RIDUZIONISTA, opposta alla precedente, la quale mira a porre l'attenzione sulle singole parti di un certo sistema, piuttosto che sull'insieme intero.

2 Vision

La "Visione" definisce l'approccio ai problemi emersi, ovvero come affrontarli.
E' possibile affrontare un problema partendo da zero, senza alcuna ipotesi? Molto difficile, quasi impensabile!
Partire dal foglio bianco significa non avere alcuna ipotesi tecnologica (come se si partisse assolutamente privi di informazioni, non si ha nulla su cui orientarsi).
La questione che adesso ci si pone è l'affrontare un problema partendo da ipotesi tecnologiche e poi arrivando ad una soluzione utilizzando gli strumenti che si conoscono (approccio bottom-up). L'approccio giusto, in ogni caso, è decidere il più tardi possibile quale tecnologia utilizzare, poichè si vuole trovare quella

giusta(la tecnologia), al momento giusto. Non si costruisce in funzione della "scatola lego" (cioè dei pezzi che hai a disposizione), ma si analizza quello che si vuol fare e, solo dopo si cerca la "scatola lego" più opportuna a per quello che si vuol realizzare.
La visione adottata in questo caso è: "Dalle tecnologie alla analisi e al progetto logico e ritorno alle tecnologie." Cosa significa?
Dopo aver capito che comunque si deve partire dalle tecnologie (e quindi non si può partire da zero, senza alcuna contaminazione tecnologica), si fanno delle ipotesi su cosa fare. Poi ci si occupa di analisi dei requisiti. Successivamente si ritorna alle tecnologie per poter dire se si ha un "abstraction-gap". Se per ogni byte di codice di business se ne devono scrivere 100 per l'infrastruttura, significa che c'è un abstraction gap enorme. Quindi la tecnologia che sto utilizzando è insufficiente, o meglio, inappropriata per il mio problema, e, di conseguenza, il tempo richiesto per lo sviluppo e il mantenimento dell'infrastruttura è eccessivo.
Si parla di "tecnology-lock" se l'applicazione è strettamente contaminata dalla tecnologia, ovviamente questa caratteristica potrebbe rappresentare un problema. Questo accade quando la scelta della tecnologia viene fatta prima rispetto le scelte di analisi/progetto; si contamina/rende strettamente dipendente il sistema finale dalla tecnologia.
Altra cosa importante da tenere a mente è: "non c'è codice senza progetto, non c'è progetto senza analisi e non c'è analisi senza requisiti".

3 Goals

L'obbiettivo principale è produrre e sviluppare software seguendo criteri di qualità, in termini di prodotto e di processo.

4 Requirements

In a home of a given city (e.g. Bologna), a ddr robot is used to clean the foor of a room (R-FloorClean).
The floor in the room is a flat floor of solid material and is equipped with two sonars , named sonar1 and sonar2 as shown in the picture (sonar1 is that at the top). The initial position (start-point) of the robot is detected by sonar1 , while the final position (end-point) is detected by sonar2 .
The robot works under the following conditions:

1. R-Start : an authorized user has sent a START command by using a human GUI interface (console) running on a conventional PC or on a smart device (Android).
2. R-TempOk : the value temperature of the city is not higher than a prefixed value (e.g. 25 degrees Celsius).
3. R-TimeOk : the current clock time is within a given interval (e.g. between 7 a.m and 10 a.m)

While the robot is working:

- it must blink a Led put on it, if the robot is a real robot (R-BlinkLed).
- it must blink a Led Hue Lamp available in the house, if the robot is a virtual robot (R-BlinkHue).
- it must avoid fixed obstacles (e.g. furniture) present in the room (R-AvoidFix) and/or mobile obstacles like balls, cats, etc. (R-AvoidMobile).

Moreover, the robot must stop its activity when one of the following conditions apply:

1. R-Stop : an authorized user has sent a STOP command by using the console.
2. R-TempKo : the value temperature of the city becomes higher than the prefixed value.
3. R-TimeKo : the current clock time is beyond the given interval.
4. R-Obstacle : the robot has found an obstacle that it is unable to avoid.
5. R-End : the robot has finished its work.

During its work, the robot can optionally:

- R-Map : build a map of the room floor with the position of the fixed obstacles. Once built, this map can be used to define a plan for an (optimal) path form the start-point to the end-point .

Other requirements:

1. The work can be done by a team composed of NT people, with 1<=NT<=4 .
2. If NT>1 , the team must explicitly indicate the work done by each component.
3. If NT==4 , the requirement R-Map is mandatory.

5 Requirement analysis

Dati i requisiti forniti dal committente, ci poniamo alcune domande:

- cosa intende il committente per robot?
- cosa intende per robot virtuale?
- possiamo avere un robot/ambiente virtuale a nostra disposizione (ad esempio dal committente)?
- dove sono presenti i sonar? Solo nell'ambiente fisico, solo nel virtuale, in entrambi...?
- cosa si intende per realizzare una mappa del pavimento? Cosa significa realizzare un percorso ottimo da start ad end point?

In base ai requisiti dati emergono alcune considerazioni e osservazioni.

- R-FloorClean: sappiamo che è presente un'entità "robot" che opera in una stanza di una abitazione, in una determinata città. La stanza di un ambiente domestico è definita come un qualsiasi spazio chiuso con pavimento solido e piatto. Il requisito ci dice anche che il robot è tenuto a pulire il pavivento della stanza.
  Per "pulire il pavimento della stanza" s'intende che il robot deve coprire tutta l'area del pavimento.
  Apprendiamo inoltre che la stanza è dotata di due sonar che rappresentano la posizione iniziale e quella finale che il robot deve raggiungere. Non ci sono coordinate specifiche per nessuno dei due sonar.
  Il robot deve essere nella posizione iniziale per poter iniziare la propria attività di pulizia della stanza.
  In base al colloquio col committente abbiamo appreso che, in mancanza di sonar fisici, è possibile utilizzare i sonar presenti nell'ambiente virtuale per far lavorare il robot fisico (e quindi avere punti di inizio e fine).

- R-Start: questo requisito implica che nel sistema è presente una seconda entità, rappresentata dal PC/smartphone, che si affianca all'entità robot. Deduciamo anche che il robot possa assumere uno stato di "working" dove esso sta lavorando.
  Ci deve essere una GUI (console) e deve essere messa a disposizione dal PC/smartphone. Questa sarà utilizzata dall'utente autorizzato per inviare comandi di START.
  L'utente autorizzato è un utente il quale ha effettuato un accesso sul sistema utilizzando le credenziali a sua disposizione.
  Dal requisito sappiamo, inoltre, che ci deve essere una fase di autenticazione, e che questa deve precedere la fase di invio del comando START. Non ci sono indicazioni sul tipo di autenticazione richesta e non ci sono vincoli su come e dove vadano mantenute le credenziali, queste saranno valutazioni affrontate in analisi del problema.
  Il requisito ci dice anche che il comando inviato tramite GUI deve raggiungere il robot e cambiarne lo stato.

- R-TempOk: ci dà un vincolo sulla temperature massima della città tollerata dal robot, sotto la quale può lavorare. Il valore di soglia, basandoci anche sull'interrogazione del committente, è a nostra discrezione.
  Il requisito non specifica come il robot ottiene il valore di temperatura, in più sappiamo, anche dall'interrogazione del committente, che il robot non è dotato di un proprio sensore in grado di rilevarla. Quindi come ottenere la temperatura sarà una valutazione da affrontare nell'analisi del problema.
  Il vincolo sul valore della temperatura è relativo a quello della città in cui il robot opera, ciò significa che la temperatura media della città reperita da un servizio esterno o da un termometro soddisfa i requisiti, così come è valida la

temperatura di una qualsiasi stanza di un edificio presente nei confini della città stessa.

- R-TimeOk: il robot può lavorare soltanto all'interno di un intervallo temporale (nell'arco delle 24 ore di una giornata) prefissato. L'intervallo non è specificato nei requisiti ed è quindi a nostra discrezione.
  Non è specificato cosa si intenda per "current clock time", potrebbe essere il tempo interno al robot (il committente ha dichiarato che il robot possa avere un suo clock interno), o il tempo relativo al fuso orario della città in cui si trova il robot, oppure il tempo relativo alla locazione dell'utilizzatore del robot. Anche queste sono valutazioni da fare in fase di analisi del problema.

- R-BlinkLed: apprendiamo che possa essere presente un robot reale, e, se è così, che il robot reale sia dotato di un led. Non è specificato il modello del led. Il led deve lampeggiare qualora il robot si trovi in stato di "working". Non è specificato chi debba comandare il led.

- R-BlinkHue: sappiamo che la casa è dotata di una Led Hue Lamp. Apprendiamo anche che possa essere presente un robot virtuale, e, se è così, che durante la sua attività debba lampeggiare la Led Hue Lamp presente nella stanza dell'abitazione.
  Non è specificato chi comandi la lampada.

- R-AvoidFix: ci dice che nella stanza possono essere presenti ostacoli fissi, come ad esempio il mobilio. Non ci viene data una definizione esplicita o formale degli ostacoli fissi.
  Durante la sua attività il robot deve evitare questo tipo di ostacolo (se presente), ma cosa si intende per "evitare"? Non è espresso nei requisiti, sarà da valutare in fase di analisi del problema.

- R-AvoidMobile: ci dice che nella stanza possono essere presenti ostacoli mobili, come ad esempio gatti o palle. Non c'è una definizione esplicita di ostacolo mobile.
  Durante la sua attività il robot deve evitare questo tipo di ostacolo (se presente), ma cosa si intende per "evitare"? Non è espresso nei requisiti, sarà da valutare in fase di analisi del problema.

In base ai due requisiti appena analizzati deduciamo che il robot debba poter percepire ostacoli esterni, non è espresso come.

- R-Stop: apprendiamo che tramite la GUI (console) fornita, l'utente autorizzato possa inviare anche comandi di STOP, i quali dovranno avere come effetto l'interruzione dell'attività del robot.
  Il requisito ci dice anche che il comando inviato tramite GUI deve raggiun-

gere il robot e cambiarne lo stato.

- R-TempKo: da questo requisito deduciamo che sia necessario che il robot sia a conoscenza delle variazioni di temperatura, in modo che, superata una certa soglia, l'attività del robot si interrompa. Valgono le considerazioni fatte per l'R-TempOk sul modo in cui reperisce i dati di temperatura.

- R-TimeKo: apprendiamo che, se il valore temporale supera il valore massimo dell'intervallo di lavoro consentito, il robot deve cessare la sua attività. Valgono le considerazioni fatte per R-TimeOk sui valori dell'intervallo.

- R-Obstacle: ci dice che è possibile che sia presente un ostacolo inevitabile, ma non ci viene detto cosa s'intenda per "inevitabile". Questo verrà valutato in fase di analisi del problema. Ciò che sappiamo è che quando si verifica questa condizione il robot deve arrestarsi.

- R-End: ci viene detto che il robot deve fermarsi quando il suo lavoro è terminato, ma come si capisce che il robot ha completato il suo lavoro? In base R-FloorClean sappiamo che è il sonar finale a rilevare la presenza del robot quando questo è nella posizione finale, quindi una volta che il robot ha raggiunto il sonar finale deve arrestarsi.

- R-Map: dobbiamo realizzare la mappa del pavimento della stanza. La mappa deve mostrare la posizione degli ostacoli fissi. La mappa può essere utilizzata per trovare il percorso ottimo dal sonar iniziale a quello finale. Cosa si intende per percorso ottimo? Lo valuteremo in analisi del problema.

5.1 Use cases

I possibili casi d'uso sono 3:

1. Autenticazione: l'utente accede al sistema utilizzando un determinato metodo di autenticazione.
2. Avvio: l'utente, tramite l'interfaccia grafica (GUI) messa a disposizione da PC o Smartphone avvia l'attività del robot.
3. Arresto: l'utente, tramite l'interfaccia grafica (GUI) messa a disposizione da PC o Smartphone arresta l'attività del robot.

5.2 Scenarios

Un possibile scenario è dato dall'utente(e.g casalinga) che, tramite smartphone, è in grado di avviare/arrestare il robot collocato in una posizione all'interno della stanza, qualora le condizioni per l'avvio siano rispettate. Il robot pulirà l'area della stanza, facendo attenzione ad evitare ostacoli fissi e mobili, e si fermerà una volta raggiunta la posizione finale.

5.3 (Domain)model

La modellazione può essere eseguita in maniera formale o non formale. Nel primo caso si utilizza un linguaggio (UML, QActor, ecc.) tramite il quale è possibile esprimere modelli comprensibili non solo dall'uomo ma anche da una macchina. Aspetto chiave della modellazione è catturare gli aspetti essenziali delle entità considerate in maniera non ambigua, in più vogliamo che che possa essere eseguito su una macchina. Il modello deve catturare struttura, interazione e comportamento, permettendoci di mettere in primo piano (foreground) aspetti fondamentali, lasciando "sullo sfondo" (background) aspetti irrilevanti a questo livello.
Ci chiediamo quanti e quali linguaggi siano presenti, quali costi e quali pro/contro abbiano, magari evidenziando anche quanto costerebbe in termini di tempo l'adottare un nuovo linguaggio. Un esempio potrebbe essere UML, strumento per delineare entità coinvolte in un sistema. Ma non va bene per due motivi; il primo è che non ci troviamo in un sistema concentrato (dove il sw è in esecuzione su un solo nodo), il secondo è il non riuscire a cogliere aspetti cardine, come struttura, interazione e comportamento, oltre al fatto di non essere eseguibile. Scegliamo di utilizzare il linguaggio di modellazione QActor, messo a disposizione dalla nostra software house, in quanto riesce a cogliere tutti gli aspetti per noi rilevanti (sopra citati). Ci saranno sicuramente aspetti che tralasceremo a questo livello, perchè di più stretta competenza dei progettisti. A noi interessa modellare in maniera formale.

Formalizziamo le entità espresse nei requisiti:

- Robot fisico: robot di tipo Mbot, dotato di ruote per il movimento
  (informazioni complete: https://www.makeblock.com/steam-kits/mbot)
- Robot virtuale: robot scritto in linguaggio Javascript da P.Soffritti
  (informazioni complete: https://github.com/PierfrancescoSoffritti/ConfigurableThreejsApp)
- Led: led di marca ELEGOO, dove led è un dispositivo elettronico a semiconduttore che al passaggio di corrente elettrica emette luce.
  (informazioni complete: https://www.elegoo.com/product/elegoo-electronic-fun-kit-bundle/)
- Led Hue Lamp: lampada Philips Led Hue Lamp, che è una lampada a led collegabile a rete internet tramite WiFi e, di conseguenza, controllabile da remoto tramite dipositivi elettronici (smartphone).
  (informazioni complete: https://www.philips.co.uk/c-m-li/hue)
- Sonar: sonar di tipo Me Ultrasonic Sensor V3.0.
  (informazioni complete: www.makeblock.cc/me-ultrasonic-sensor/)

La formalizzazione delle entità in gioco è la seguente: (vedi file).

5.4 Test plan

La fase di test plan è fondamentale per garantire solidità e affidabilità al nostro software, oltre ad assicurare un corretto metodo di lavoro in team in caso di

parallelizzazione dei compiti.
La fase di test plan può essere affrontata in maniera formale o informale.
Per formale si intende che i test vengono fatti in maniera automatica (JUnit) o manuale (personale che esegue test manuali). In questo caso i test sono eseguibili su una macchina.
Informale significa valutare empiricamente e fare un confronto tra risultato reale e risultato atteso. I test non sono strutturati per essere eseguiti da una macchina.
In fase di analisi sono stati pensati e eseguiti test formali, ma di tipologia manuale. Sfruttando lo stesso linguaggio di modellazione abbiamo potuto verificare che il nostro sistema si comportasse in maniera corretta secondo i requisiti e le richieste del committente (ad esempio test comandi start/stop utente, test valori temperatura, ...).
In una fase successiva (progettazione) verranno integrati test formali automatici che ci permetteranno di ridurre il tempo dedicato ai test manuali, oltre che a validare automamente il codice che viene prodotto (Junit).

6 Problem analysis

Nell'analisi del problema si valutano i problemi emersi nella fase precedente e si prospettano pro e contro delle possibili alternative. L'analisi è anche la fase che pianifica attività, e prospetta costi e rischi.

Elenco dei problemi emersi/punti aperti di interesse per il problema:

1. Autenticazione: da chiarire quale tipo di autenticazione usare, dove mantenere le credenziali e chi implementa il servizio di autenticazione (pc, servizio esterno, ...?)
2. Interfaccia GUI: client e server sulla stessa macchina (pc/smartphone) o separati? Quale metodo di comunicazione tra pc e robot?
3. Temperatura: chiarire in che modo il robot si rende conto della temperatura attuale, o della soglia di temperatura superata (fornita dal pc al robot? reperita dal robot tramite un'entità/servizio esterno/a?)
4. Orario: chiarire cosa si intende per "current clock time". Potrebbe essere il tempo interno al robot (il committente ha dichiarato che il robot possa avere un suo clock interno), o il tempo relativo al fuso orario della città in cui si trova il robot, oppure il tempo relativo alla locazione dell'utilizzatore del robot...
5. Led: chi comanda il led (robot, altro...)?
6. Led Hue Lamp: chi comanda la lampada? In che modo?
7. Ostacoli: discutere cosa si intende per ostacolo fisso e mobile, capire come si indentificano gli ostacoli, capire anche cosa si intende per "evitare" un ostacolo e capire cosa si intende per ostacolo inevitabile.
8. Mappa: cosa si intende per "mappa" e per "percorso ottimo"?

Analisi autenticazione:
Per ragioni di costi e semplicita' conviene modellare il metodo di autenticazione come autenticazione tramite username e password. Questo perche' l'utilizzo di

sistemi con smart-card o lettori biometrici (retina, impronte digitali) comporta costi aggiuntivi ed elevati (oltre ad aumentare i tempi di sviluppo), rispetto al metodo tramite username e password.
Utilizzando username e password dobbiamo tenere traccia di questi dati. Mantenerli su Pc/smartphone vorrebbe dire più efficienza, poichè si lavora localmente, ma maggiori rischi in quanto il livello di protezione è basso.
Un'alternativa sarebbe mantenere le credenziali sul robot, questo aumenta i tempi di risposta in fase di autenticazione e il livello di sicurezza è simile al caso precedente. E' quindi una scelta poco conveniente.
Altra possibilità sarebbe quella di appoggiarsi ad un servizio esterno (Google, Facebook, ...) per gestire l'autenticazione utente, questo diminuisce l'efficienza però dà maggiori garanzie dal punto di vista della sicurezza e permette di non dover gestire le credenziali. Scegliere servizi esterni consente anche di evitare l'implementazione di metodi di registrazione, necessaria nei casi descritti in precedenza.
La scelta ricade, quindi, sull'autenticazione (username+password) tramite servizio esterno, in modo da ridurre costi e tempi necessari per l'implementazione di questo aspetto del sistema.

Analisi temperatura:
Si mettono in evidenza due strade principali per affrontare il problema di gestire il requisito sulla temperatura: la prima è quella di sfruttare un'interazione a polling, cioè dove il robot prende l'iniziativa e richiede periodicamente il valore di temperatura o al pc/smartphone o ad un servizio esterno (es. servizio web).
Una seconda opzione può essere quella di far arrivare i valori di temperatura al robot in maniera periodica (o di soglia di temperatura superata), quindi il robot dovrà predisporsi per ricevere dati sulla temperatura. L'invio dei dati, in questo caso, sarebbe a carico del pc/smartphone.
Quest'ultimo approccio sposterebbe parte della logica applicativa sul pc/smartphone, in più, in caso di perdita di comunicazione tra GUI e robot, sarebbe compromessa la verifica del valore di temperatura richiesta nei requisiti.
Una strada alternativa, senza utilizzare la rete internet, sarebbe quella di utilizzare un termometro che rilevi in locale la temperatura e che possa comunicarla tramite protocolli a corto raggio (es. bluetooth). Questo comporta un costo aggiuntivo e presuppone che il robot abbia la capacità di comunicare con il protocollo utilizzato dal termometro.
La nostra scelta ricade sull'idea di utilizzare l'approccio a polling, ossia di fare in modo che sia il robot a procurarsi, periodicamente, le informazioni di temperatura da un servizio esterno. Anche in questo caso rimane la necessità di mantenere una connessione ad internet attiva.

Analisi orario:
Sappiamo, dall'analisi dei requisiti, che esiste un "current clock time"; Esso può essere inteso come orologio interno al robot, oppure un orologio esterno come l'orologio del sistema pc/smartphone oppure quello di un servizio esterno.
Utilizzando il clock interno al robot evitiamo di dover effettuare richieste tramite

rete internet, e miglioriamo l'efficienza ma bisogna fare attenzione che il clock interno sia allineato con quello della città nel quale il robot opera. Nel caso reperissimo i dati dell'orario da un servizio esterno potremmo considerare o il fuso orario del robot o quello del pc/smartphone (sotto l'ipotesi che siano in due ambienti a fuso orario diverso). Nel caso fossero in due ambienti sotto due fusi orari diversi, si considera ragionevole considerare il fuso orario del robot come quello valido.
Si mettono in evidenza due alternative principali per affrontare il problema di gestire il requisito sul tempo: la prima è quella di sfruttare un'interazione a polling, cioè dove il robot prende l'iniziativa e richiede periodicamente il tempo o al pc/smartphone o ad un servizio esterno (es. servizio web).
Una seconda opzione può essere quella di far arrivare i valori al robot in maniera periodica (o di valori temporali fuori range), quindi il robot dovrà predisporsi per ricevere. L'invio dei dati, in questo caso, sarebbe a carico del pc/smartphone. Quest'ultimo approccio sposterebbe parte della logica applicativa sul pc/smartphone, in più, in caso di perdita di comunicazione tra GUI e robot, sarebbe compromessa la verifica del valore del tempo richiesta nei requisiti.
La nostra scelta ricade sull'utilizzo dell'orario interno al robot, in modo da migliorare l'efficienza ed evitare di fare affidamento a servizi esterni, in quanto sappiamo che il robot dispone di un proprio orario locale. Si può pensare, in futuro, di integrare un meccanismo di modifica dell'orario interno del robot tramite interfaccia GUI.
Nel caso di perdita di comunicazione con il servizio che fornisce orario (e temperatura) si decide che il robot non potrà avviare la propria attività.
Per quanto riguarda la frequenza di aggiornamento dei suddetti valori il committente non ha espresso vincoli a riguardo, per cui in fase di progetto si concretizzerà il tempo specifico di intervallo di aggiornamento.

Analisi led:
Dall'analisi dei requisiti sappiamo che il led è posizionato sul robot, di conseguenza sarà questo a controllare il led affinchè faccia quanto richiesto dai requisiti (blink in caso di attività).

Analisi Led Hue Lamp:
Dall'analisi dei requisiti sappiamo che la Led Hue Lamp è presente nella casa dove il robot agisce. Il comando della lampada viene fatto tramite rete, di conseguenza potrebbe essere a carico sia del pc/smartphone che del robot.
Scegliamo di affidare questa responsabilità al nostro software di controllo dei robot, questo perchè l'interazione tra esso e la lampada sarà locale, con ovvi vantaggi in termini di efficienza e tempi di risposta.

Analisi ostacoli:
Da requisiti non viene specificato cosa si intende per ostacolo fisso o mobile. Per ostacolo fisso noi intendiamo un'entità immobile che ostruisce il percorso del robot dal sensore iniziale a quello finale. L'ostacolo mobile si differenzierà dal

fisso solo per la sua caratteristica di movimento. In ogni caso un ostacolo occupa parte dell'area della stanza.
In entrambi i casi il robot può e deve evitarli. Cosa intendiamo per "evitare"? Fare in modo che il robot riesca a modificare la propria traiettoria al fine di coprire la più grossa porzione di stanza possibile, e che riesca a proseguire verso il sonar finale. Per fare questo può aggirare l'ostacolo (nel caso questo sia fisso) oppure attendere per un periodo limitato di tempo prima di proseguire (nel caso esso sia mobile).
Per risolvere il problema di riconoscimento degli ostacoli lungo il cammino ci avvaliamo del sensore di movimento posto sul muso del robot (fisico e/o virtuale). Alternative potrebbero essere videocamera, sistemi gps, che sono però più costose rispetto all'utilizzo del sensore.
Un ostacolo inevitabile è un ostacolo che non può essere aggirato in alcun modo e che impedisce il raggiungimento, al robot, del sensore finale.

Analisi GUI:
Decidiamo di avere un meccanismo ad eventi tra pc e robot, in modo da disaccoppiare e rendere flessibile il sistema.
In termini di applicazione GUI per pc prendiamo in considerazione l'utilizzo di un'interfaccia web, quindi basata su webserver e client che fa da browser, questo perchè si presta alla struttura della nostra architettura dove pc e robot comunicano via rete internet, in più ci permette di creare in maniera rapida un'interfaccia funzionale.
Ci sono tre strade per realizzarla:

1. mantenere tutto su pc/smartphone, ossia parte web server e client/browser
2. avere solo la parte di visualizzazione e comando (browser) sul dispositivo pc/smartphone, e la parte web server su un server esterno a pc e robot
3. avere solo la parte di visualizzazione e comando (browser) sul dispositivo pc/smartphone, e la parte web server sul dispositivo robot

In base alle valutazioni su semplicità di utilizzo, costi e flessibilità, decidiamo di intraprendere la seconda strada, ossia di avere la parte grafica della GUI su pc/smartphone e la parte web server che risiede su un dispositivo esterno al sistema pc-robot. Questo garantisce anche più flessibilità per eventuali modifiche e facilità di implementazione di estensioni future dell'interfaccia.
Inoltre questa scelta ci permette di poter posizionare il server sia su un nodo ospitato dalla software house, che un nodo ospitato da servizi esterni (ad esempio Amazon AWS).
Da analisi dei requisiti sappiamo che la GUI deve essere in esecuzione (running) sul pc/smartphone, dove per "running" noi intendiamo la componente di controllo che consente l'invio di comandi START/STOP al robot.

Analisi movimento/planning ostacoli:
Si ipotizza di lavorare con un robot capace di muoversi in quattro direzioni (destra, sinistra, avanti, indietro).
L'attività del robot inizia con il comando START, che impartisce ad una logica

di controllo l'ordine di avviarsi. Questa, a sua volta, pilota il robot fisico avvalendosi delle quattro direzioni (destra, sinistra, avanti, indietro) al fine di portarlo all'altezza del sonar finale, coprendo tutta la superficie a sua disposizione ed evitando ostacoli presenti sul suo cammino.
Il robot deve essere nella posizione iniziale, cioè vicino al sonar1, per poter iniziare la propria attività.
Il controllo che permette di coprire tutta la stanza ed evitare gli ostacoli verrà affrontato nella fase di progettazione.

Analisi contesto e tipologia collegamento:
Il nostro scenario si colloca all'interno di un sistema distribuito, nel quale varie entità e servizi sono distribuiti su nodi computazionali distinti, o per meglio dire, eterogenei. Si potrebbe optare per varie opzioni al fine di far comunicare le varie parti: scambio di messaggi, o eventi, utilizzando un infrastruttura di supporto alla comunicazione offerta dal linguaggio o più generale. Nel caso attuale, dopo valutazioni, si è deciso di adottare un approccio a eventi, per ragioni di estendibilità, flessibilità e comodità. Infatti sono disponibili servizi che offrono questo supporto gratuitamente, evitando di vincolarsi a strumenti del linguaggio per permettere la comunicazione di elementi eterogenei. Il vocabolario con la quale queste entità dovranno dialogare sarà definito formalmente nell'architettura logica.

Analisi Mappa:
Per percorso ottimo intendiamo individuare la sequenza di azioni con cui arrivare dal punto iniziale a quello finale considerando risparmio di energia. Questo vuol dire avere il minor consumo di energia possibile coprendo la maggior superficie possibile del pavimento.
La mappa è una rappresentazione grafica/logica dell'ambiente in cui il robot si muove, composta da posizioni libere ed occupate. Per occupate si intendono posizioni i cui spazi sono occupati da un ostacolo fisso.

Costi:
E' necessario fare una valutazione delle spese e dei costi da affrontare.
Per l'utilizzo dell'infrastruttura meteo è stato valutato di utilizzare un servizio esterno, che offre diverse soluzioni di scalabilità (e di conseguenza costi) in base al n° di richieste effettuate al minuto. Per servizio di esempio vedere https://www.openweathermap.org/price.
Un'altra valutazione economica è quella del servizio di host del server frontend. Appoggiandoci a un servizio esterno alla software house abbiamo vantaggi dal punto di vista della scalabilità, con costi crescenti in proporzione agli utenti. Per servizio di esempio e costi vedere https://aws.amazon.com/it/ec2/pricing/on-demand/.
Per dotare il robot Mbot di un modulo middleware si rende opportuno l'acquisto e l'utilizzo di un dispositivo montato e collegato con il robot, come ad esempio

un Raspberry Pi (https://www.raspberrypi.org/products/raspberry-pi-3-model-b/).

6.1 Logic architecture

L'obiettivo è produrre modelli formali. Un modello deve rispecchiare gli aspetti essenziali delle risorse al fine di far risaltare ciò che è rilevante per il problema o il dominio applicativo nel quale si colloca il problema stesso.
Occorre formalizzare per rendere non ambiguo il contratto che ci lega al committente.
Si vedano i modelli prodotti durante l'analisi del problema.

6.2 Abstraction gap

Se identifichiamo una distanza significativa tra quello che abbiamo e quello che dovremmo avere, allora siamo in presenza di un abstraction gap che dobbiamo colmare.
Utilizzando il linguaggio di modellazione QActor anche per la fase di progettazione del software, e' possibile incorrere in un abstraction gap. In particolare la modellazione del movimento e della logica di aggiramento ostacoli risulta particolarmente ostica, e si percepisce una difficolta' di fondo dovuta a mancanza di espressivita' totale del linguaggio in termini di cambi di stato dovuti a determinate precondizioni.

6.3 Risk analysis

Elenchiamo i rischi collegati alle valutazioni fatte in fase di analisi del problema:

- Autenticazione: utilizzando il modello di autenticazione con username e password potremmo avere rischi aggiuntivi in termini di furto di dati di accesso, dovuto a terze parti che possono introdursi nella comunicazione tra pc/smarpthone e servizio esterno. L'utilizzo di sistemi di accesso esterni noti e sicuri può ridurre questo rischio.
- Temperatura: l'approccio a polling può esporre a problemi dovuti a caduta della linea internet tra robot e servizio metereologico, oltre che la caduta del servizio stesso considerato.
- Orario: utilizzando l'orario interno al robot possono esserci disallineamenti sull'orario in caso di spostamento del robot in una città con fuso orario diverso rispetto a quello con cui il robot è configurato. Per coprire il rischio si potrebbe permettere la modifica dell'orario interno al robot tramite interfaccia GUI.
- Led Hue Lamp: la perdita della connessione alla rete da parte del software di controllo del robot potrebbe impedire l'accensione, o lo spegnimento, della lampada
- Ostacoli: possono esserci ostacoli imprevisti che compromettono la corretta esecuzione dell'attività del robot (ad esempio caduta oggetti)

7 Work plan

In questa fase si valutano tempi, costi, strategie di investimento, formazione del personale e persone necessarie.
Tempi e costi sono ridotti in quanto utilizziamo o prodotti open-source oppure sistemi già pronti messi a disposizione dal committente. Non necessitiamo di particolare formazione del personale, mentre per quanto riguarda le persone necessarie riteniamo sufficienti i componenti del team attuale (ossia tre persone).
Ci occupiamo di scegliere l'opportuno processo produttivo, in cui ci sono due scuole in contrasto: l'agile (di cui scrum è un esempio) e il modello a cascata.
Agile development consiste in lavorare in gruppi auto organizzati, multifunzionali e orientati allo stretto contatto con il cliente finale, al fine di ottenere prodotti utilizzabili in tempo rapido, di migliorare costantemente i prodotti stessi e di rimanere flessibili a ipotetici cambiamenti resisi necessari nel tempo.
Il processo di sviluppo a cascata impone che, prima di trattare il codice, sia necessario occuparsi della fase di modellazione. Solo in un secondo momento si passerà alla fase di sviluppo vero e proprio, tenendo opportunamente aggiornata la parte di modello.
In base alle nostre esigenze riteniamo sia più opportuno lavorare tramite modello Agile, in particolare Scrum. Questo ci permette di focalizzarci sulle richieste del committente, affrontando quelle più prioritarie per prime, e assicurandoci un ciclo di lavoro flessibile e costantemente monitorato.
Lavoreremo quindi tramite "sprint" periodici di breve durata, dove ci suddivideremo le attività da svolgere, proponendoci meeting brevi e costanti per valutare e controllare il flusso di lavoro.
Al fine di agevolare l'utilizzo della metodologia Scrum useremo uno strumento del tutto gratuito, Meistertask, che permette di tracciare le attività da svolgere, di valutarle in termini di priorità e di assegnarle ai componenti del team. Ogni task avrà diversi stati, personalizzabili, come ad esempio "open" nel caso sia ancora da prendere in carico, "in progress" nel caso qualcuno lo stia affrontando, e "done" quando il task sia stato completato.
Utilizzeremo anche il sistema di versionamento Git, con una repository comune nel quale saranno presenti codice, modelli e documentazione. Tramite l'utilizzo di branch ottimizzeremo il flusso di lavoro e la parallelizzazione delle attività da svolgere.
Il team inoltre si avvale di lavoro cooperativo, svolto incontrandosi personalemente nella gran parte dei casi. Quando questo non è possibile si ricorre a strumenti come TeamViewer, il quale mette a disposizione una modalità "meeting", con la quale è possibile condividere schermi e lavorare in modo condiviso. Altro supporto utilizzato è Skype.

8 Project

La fase di progetto, svolta dal progettista, prende in input il lavoro svolto dall'analista, ovvero il modello formale che descrive il sistema e propone, a sua

volta, un ulteriore modello, nel quale vengono specificati e risolti problemi lasciati in sospeso dall'analista. Anche in questo caso il sistema è specificato in termini di struttura, interazione e comportamento.

Punti aperti dalla fase di analisi del problema:

- Quale tipologia di server utilizzare per comunicazione tra pc/smartphone e server?
- Autenticazione attraverso provider esterni (quale utilizzare e come)
- Come effettuare la pulizia della stanza?
- Come rilevare ed evitare gli ostacoli?
- Come comunicare con led e led hue lamp?
- Come comunicare con provider esterni e quale provider utilizzare?
- Con quale frequenza interrogare il provider di temperatura o controllare l'orario?
- Quale protocollo utilizzare per comunicazione tra server e robot?
- Come comunicano i sensori presenti nella stanza con il robot?
- Come realizzare la mappa della stanza?

Server:
In base alle considerazioni dell'analisi sappiamo di dover utilizzare un server su piattaforma separata da quella del pc o del robot. Alla luce di questo possiamo utilizzare un web server RESTful (utilizzando tecnologia NodeJs ed Express) ospitato su un nodo esterno, che è Amazon AWS, perchè questo comporta benefici in termini di scalabilità, nel caso il n° di utenti cresca nel tempo.

Autenticazione:
Come provider di autenticazione decidiamo di utilizzare il servizio di autenticazione di Google, in quanto molto sicuro, diffuso e gratuito. Per fare questo ci appoggeremo ad una libreria fornita da Google stesso (https://github.com/google/google-auth-library-nodejs) per integrare il sistema di autenticazione nel frontend server.

Pulizia stanza:
Decidiamo di effettuare la pulizia della stanza facendo l'ipotesi che i sonar permettano di inquadrare l'area di azione, in termini di larghezza e lunghezza.
L'operazione di pulizia verrà eseguita partendo dal sonar iniziale, muovendosi in linea retta fino ad intercettare il segnale del sonar finale. In base alla distanza dal sonar finale si eseguira' una svolta o si fermera' l'attivita'. Si possono ricavare larghezza e lunghezza della stanza, e si può riconoscere la presenza dei muri in maniera automatica e invertire la direzione del robot una volta che questi vegono trovati.
Nel caso, prima di arrivare al segnale del sonar finale, e quindi al primo muro, si incontri un ostacolo sarà necessario effettuare una procedura di aggiramento che porti il robot in corrispondenza del segnale del sonar finale, dandogli modo di conoscere la dimensione della stanza.
Nell'operazione di svolta, da eseguire quando si incontra un muro o il sonar2, è necessario spostarsi di una casella verso il sonar finale, in seguito invertire la direzione e riprendere il movimento rettilineo fino al muro successivo.

Rilevazione e aggiramento ostacoli:
Per quanto riguarda la rilevazione degli ostacoli ci avvaliamo del sensore posto sul muso del robot. Il sensore ha una distanza di rilevazione di 4 metri, ma noi consideriamo solo gli ostacoli entro i 15 cm.
Una volta rilevato un ostacolo durante il movimento, il robot si fermerà ed effettuerà un secondo tentativo dopo alcuni secondi, per verificare che l'ostacolo sia ancora presente. Se è così l'ostacolo è considerato fisso, altrimenti mobile.
In caso di ostacolo fisso il robot deve cercare di evitarlo fintanto che non trova uno spazio per passare. Se durante la ricerca di uno spazio trova un muro, il robot invertirà la direzione di ricerca. Nel caso venga trovato un secondo muro, il robot dichiarerò l'ostacolo come insormontabile e si fermerà.
In caso di ostacolo mobile prosegue il suo moto nella stessa direzione.

Comunicazione con led e led hue lamp:
Per comunicare con il led posto sul robot fisico utilizzeremo degli script bash che permettono di cambiare stato del pin (GPIO del raspberry, montato su mbot).
Per quanto riguarda la led hue lamp, sappiamo che è possibile comunicare con essa tramite rete internet (Wifi) e con metodologie RESTful (https://www2.meethue.com/en-us/about-hue).

Comunicazione con provider meteo:
Come servizio per reperire informazioni su meteo si utilizza OpenWeatherMap, che è un servizio web che offre API per acquisire le informazioni di nostro interesse. Per richiedere i dati occorre formulare richieste in un determinato formato (https://www.openweathermap.org/current).
La struttura dei dati ricevuta è in formato JSON, a noi interessa il campo "temp" che identifica la temperatura attuale. Occorre specificare, tramite parametro dell'API, se ricevere dati in formato Celsius o Fahrenheit.

Frequenza di interrogazione provider meteo / controllo orario:
Poniamo un vincolo massimo di attesa tra una richiesta e la seguente. Di conseguenza decidiamo di effettuare almeno una richiesta ogni minuto, fino ad un massimo di una richiesta al secondo, in modo da tenere opportunamente aggiornate le informazioni del modello logico.
Applicheremo gli stessi vincoli di frequenza anche al controllo sull'orario.

Protocollo per comunicazione:
L'infrastruttura QActor lavorerà ad eventi, mentre il protocollo sottostante sarà a scambio di messaggi.
La nostra scelta ricade sul protocollo Mqtt, in quanto si può sfruttare un'infrastruttura locale di supporto (Mosquitto Broker) oppure un'infrastruttura esterna (https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment).

Comunicazione tra sensori e robot:
Consideriamo l'ambiente virtuale, dotato di sensori di ambiente virtuali (sonar1 e sonar2), in quanto non disponiamo di sensori da applicare al mondo fisico. La comunicazione tra sensori virtuali e software di controllo viene realizzata tramite eventi, con struttura(basata su linguaggio JSON) stabilita da chi ha realizzato l'infrastruttura virtuale (https://github.com/PierfrancescoSoffritti/configurable-

threejs-app). Ci dovrà essere una entità di intermezzo che interpreta questi messaggi e li rimappa su eventi comprensibili dalla "mente".
Analizzando l'ambiente virtuale messo a nostra disposizione scopriamo che i sonar posti nella stanza emettono eventi con un'alta frequenza, per cui decidiamo di emettere eventi captati dalla mente del robot solo quando ci sono variazioni nelle rilevazioni dei sonar stessi.
Per quanto riguarda il sensore presente su robot fisico, riusciamo ad intercettare le rilevazioni fisiche del sonar, cioè gli ostacoli rilevati, ma dobbiamo rimappare questi eventi in una struttura a messaggi (attraverso infrastruttura mqtt), ed inviarli alla "mente" del sistema.

Realizzare la mappa:
Per realizzare la mappa della stanza è opportuno suddividere l'area in caselle, aventi area identica a quella del robot. In questo modo, ipotizzando che ogni movimento del robot lo sposti di una casella alla volta, è possibile tenere traccia delle posizioni libere e di quelle occupate da ostacoli fissi. Gli ostacoli mobili vanno ignorati e non bisogna tenerne traccia, mentre gli ostacoli insormontabili, essendo fissi, sono riportati sulla mappa (occupando le caselle).

8.1 System

Il nostro sistema si compone di diverse entità: pc/smartphone dotata di GUI, server frontend, server per scambio di messaggi (mqtt), robot (fisico/virtuale), led montato su robot fisico, led hue lamp presente nella stanza, sonar ambientali.

Software di controllo del Robot:
è suddiviso in diverse sotto-entità:

- mind
- core di movimento
- notificatore meteo/orario
- gestore led/led hue lamp

Pc/Smartphone:
è strutturato come un browser che fa da GUI e che si collega al server di frontend.

Server frontend:
server web RESTFul realizzato grazie al supporto di NodeJs ed Express.

Server per scambio di messaggi:
Server con struttura publish/subscribe, utilizza il protocollo mqtt con implementazione Mosquitto.

Robot fisico:
Robot Mbot, per riferimenti ufficiali sulla struttura vedere https://www.makeblock.com/steam-kits/mbot

Robot virtuale:
Robot costruito con linguaggio Javascript, per riferimento vedere https://github.com/PierfrancescoSoffritti/con threejs-app.

Led:
Led ELEGOO, per riferimento https://www.elegoo.com/product/elegoo-electronic-fun-kit-bundle/.

Led Hue Lamp:
Lampada a led Philips, per riferimento https://www2.meethue.com/en-us/about-hue.

Sonar ambientali:
Sonar per rilevazione entità nell'ambiente, il modello utilizzato è lo stesso montato sull'Mbot. (Me Ultrasonic Sensor v3.0)

9 Implementation

Nella fase di implementazione ci siamo suddivisi i compiti sfruttando servizi di tracking delle attività, in modo da bilanciare il carico in maniera equilibrata. Alcune delle attività principali sono quelle di implementazione del server e del client frontend con autenticazione, realizzazione del meccanismo di blink/spegnimento led e led hue lamp, realizzazione del movimento del robot fisico e virtuale, implementazione della logica di ritrovamento dati meteo ed orario, creazione della logica di aggiramento ostacoli.

Precisamente, sono stati realizzati:

- POJO per il led fisico e script per avviarlo
- Modulo per recuperare data e temperatura
- Modulo per collegare l'ambiente virtuale
- Una serie di utils per vari scopi

Sono stati riutilizzati e riadattati moduli per pilotare il robot fisico. E' stato adattato il frontend per far sì che supportasse l'autenticazione su un provider esterno.

Il linguaggio di implementazione da noi utilizzato maggiormente per questo progetto e' il linguaggio Java, questo perche' e' un linguaggio ampio, diffuso, multipiattaforma e anche perche' il linguaggio di modellazione QActor e' in grado di produrre codice Java gia' di default.

10 Testing

Molto importante e' la fase di testing in cui viene testato il prodotto ottenuto. Il tutto dovrebbe essere fatto il linea con la filosofia di testare tutto al fine di

provare il sistema il prima possibile.
Si può parlare di testing manuale o automatico. Nel primo caso occorrerà dare in pasto al sistema dei valori ed assicurarsi che l'output sia in linea con le aspettative. Nel secondo caso si possono utilizzare strumenti di testing automatico (e.g. JUNIT), i quali, una volta eseguiti, testano le componenti specificate, riportando in output eventuali problemi e dove essi si sono verificati.
Abbiamo deciso di utilizzare il sistema di testing Junit (versione 5). I nostri test sono visibili nel file allegato.

11 Deployment

Nel nostro sistema necessitiamo di due pacchetti di deploy, per due diversi sottosistemi:

1. frontend server: il pacchetto generato conterrà il codice che realizza il frontend server, il quale sarà poi da avviare in una qualsiasi macchina (ad esempio server su Amazon AWS) accessibile tramite rete internet dal browser di pc/smartphone (GUI).
2. controllo del robot: pacchetto contenente il software di controllo del robot (fisico o virtuale), generato tramite routine Gradle. Il deploy verrà effettuato sul middleware di controllo (Raspberry) nel caso del robot fisico, oppure su un sistema che comunica con l'ambiente nel quale è presente il robot virtuale.

Nel nostro caso abbiamo bisogno dell'ambiente virtuale (con relativo robot virtuale) per poter simulare i sonar1, sonar2 ed i muri. Quindi occorrerà avviare anche l'environment virtuale (software di Soffritti).

12 Maintenance

Per mantenere il codice è opportuno avvalersi di tecniche di integrazione continua e di monitoraggio costante del processo produttivo e del codice prodotto.
Un esempio molto diffuso in tal senso è DevOps, una pratica che punta a unificare il flusso di lavoro di sviluppo software e operazioni legate al software prodotto. Il punto focale è automatizzare e monitorare i passaggi che portano al software finale, come l'analisi, la progettazione, l'integrazione, il testing e il deployment del prodotto finale. Il tutto è pensato per ridurre i tempi entro i quali una modifica si ripercuote sul prodotto finale, facendo sempre attenzione a mantenere alti standard qualitativi.
Alcuni strumenti che possono aiutare a mettere in pratica questi principi sono Jenkins e JetBrains TeamCity. E' possibile delegare a questi strumenti numerosi compiti, in modo che, una volta configurati, il processo produttivo sia del tutto automatizzato ed efficiente. In questo modo lo sviluppatore si occupa quasi solo dello sviluppo del codice, dopodiché sarà compito degli strumenti di reperire le modifiche che sono state effettuate e applicare le dovute operazioni, come ad

esempio compilare il codice, validare il codice tramite suite di test, produrre report sui risultati dei test e sui tempi di produzione, effettuare operazioni di manutenibilità, produrre i pacchetti in uscita, completare il deploy.

See [?] until page 11 (CMM) and pages 96-105.

13 Information about the author

Photo of the author

References