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, in un questo istante, del processo di sviluppo del software. Avere una documentazione completa circa tutte le fasi (Analisi, Progettazione, Implementazione, Testing) è importante per far 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" è una frase che fa capire come ci si approcci alle cose, ovvero come affrontare problemi. Lo scopo dell'analisi dei requisiti è quello di capire il problema, analizzando i requisiti ed evidenziando aspetti problematici, successivamente, produrre (in modo formale) uno o più modelli che rappresentino il sistema.
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 ho nulla su cui orientarmi).

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 con 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

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.

- 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 (fornita dal pc al robot? reperita dal robot tramite un'entità/servizio esterno/a?).
  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. 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. 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.
  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 raggiungere 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. Val-

gono 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

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.

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 dellla 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.
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

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.

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 tempo:
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 strade 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'idea di utilizzare l'approccio a polling, ossia di fare in modo che sia il robot a procurarsi, periodicamente, le informazioni sul tempo da un servizio esterno, anche in relazione al fatto che facciamo già polling per reperire la temperatura, quindi si può pensare di rivolgersi ad un servizio metereologico che fornisca entrambe le informazioni. Anche in questo caso rimane la necessità di mantenere una connessione ad internet attiva.
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.

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 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 non entri in contatto con l'ostacolo ma 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, prevediamo che il robot sia dotato di un sensore di movimento, posto sul muso. 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 evitato e che impedisce il raggiungimento, al robot, del sensore finale.

Analisi GUI:
Decidiamo di avere un meccanismo a scambio di messaggi 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 terza strada, ossia di avere la parte grafica della GUI su pc/smartphone e la parte web server che risiede sul dispositivo robot. Questo garantisce anche più flessibilità per eventuali modifiche e facilità di implementazione di estensioni future dell'interfaccia.
Da analisi dei requisiti sappiamo che la GUI deve essere in esecuzione (running) sul pc/smartphone, per "running" noi intendiamo la componente di controllo che consente l'invio di comandi START/STOP al robot, quindi la parte di "browser" web.

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 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 ad eventi, per ragioni di estendibilità, flessibilità e comodità, appoggiandosi ad un'infrastruttura mqtt (usa protocollo mqtt). 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 del modello e degli strumenti:
Messi in luce vari problemi derivati dall'analisi dei requisiti, a questo punto, ci occorre trovare un linguaggio suffientemente espressivo e potente, il quale ci possa permettere di formalizzare le scelte di questa fase (analisi, COSA, NON COME), in modo essenziale e conciso. Ricordiamo che fare un modello di un sistema o di un'entità significa mettere in evidenza tre aspetti cardine: 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 un 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.

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.

6.2 Abstraction gap

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.

- Temperatura: l'approccio a polling può esporre a problemi dovuti a caduta della linea internet tra robot e servizio metereologico, oltre che la sola caduta del servizio considerato.
- Orario: l'approccio a polling può esporre a problemi dovuti a caduta della linea internet tra robot e servizio "timestamp", oltre che la sola caduta del servizio considerato.
- Led Hue Lamp: la perdita della connessione alla rete da parte del robot, potrebbe impedire l'accensione, o lo spegnimento, della lampada

7 Work plan

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 model based.
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.
Model based development 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 questa fase entra in gioco anche il testing, che è fondamentale per assicurare solidità e riddure al minimo i rischi di bug critici in fase di produzione. I test possono essere fatti non solo alla fine della costruzione del sistema, ma anche, e soprattutto, durante. Un'ulteriore aspetto da considerare è che i test possono anche essere resi automatizzati.

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 (e.g. quale protocollo?). Anche in questo caso il sistema è specificato in termini di STRUTTURA, INTERAZIONE e COMPORTAMENTO.

8.1 Structure

8.2 Interaction

8.3 Behavior

9 Implementation

10 Testing

Molto importante 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.
Possibile 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.

11 Deployment

12 Maintenance

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

13 Information about the author

Photo of the author

References