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 prexed 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 o dubbi da chiarire.

- 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 reqeuisito ci dice anche che il robot è tenuto a pulire il pavivento della stanza.
  Per pulire il pavimento della stanza 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. I sonar sono sonar di tipo Me Ultrasonic Sensor V3.0.

- 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.
  La GUI deve essere messa a disposizione dal PC/smartphone, e 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.
  Questo comporta la presenza di un sistema di autenticazione utenti che dovrà essere presente in una delle due entità in gioco o su sistemi esterni.
  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 quanto il robot non è dotato di un proprio sensore in grado di rilevarla. Quindi la temperatura deve essere fornita o reperita da un'entità/servizio esterno/a.

- 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 come il robot tiene traccia dell'andamento temporale, ma il committente ha dichiarato che il robot può avere un proprio clock interno.

- R-BlinkLed: apprendiamo che possa essere presente un robot reale (TODO: DEFINIRE FORMALMENTE ROBOT REALE COME MBOT???), e, se è così, che il robot reale sia dotato di un led ELEGOO, dove led è un dispositivo elettronico a semiconduttore che al passaggio di corrente elettrica emette luce. Il led deve lampeggiare qualora il robot si trovi in stato di "working".
- R-BlinkHue: sappiamo che la casa è dotata di una Led Hue Lamp di marca Philips, che è una lampada a led collegabile a rete internet tramite WiFi e, di conseguenza, controllabile da remoto tramite dipositivi elettronici (smart-

phone). La Led Hue Lamp in questione deve lampeggiare fintanto che il robot virtuale è in stato di working. Apprendiamo 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.
Il robot virtuale è definito come il robot rappresentato nell'applicazione sviluppata tramite javascript da Soffritti.
Non è specificato chi comandi la lampada.

- R-AvoidFix: ci dice che nella stanza possono essere presenti ostacoli fissi, dove l'ostacolo fisso è definito come persona o entità che occupa l'area di azione del robot e che non effettua alcun movimento proprio (ad esempio mobilio). Durante la sua attività il robot deve evitare questo tipo di ostacolo (se presente), dove per "evitare" intendiamo il modificare la propria traiettoria in modo da non entrare in contatto con esso pur continuando la sua attività.

- R-AvoidMobile: ci dice che nella stanza possono essere presenti ostacoli mobili, dove l'ostacolo mobile è definito come persona o entità che occupa l'area di azione del robot e che si muove liberamente. Durante la sua attività il robot deve evitare questo tipo di ostacolo (se presente), dove per "evitare" intendiamo il modificare, o sospendere, la propria traiettoria in modo da non entrare in contatto con esso pur continuando la sua attività.
  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 fornita, l'utente autorizzato possa inviare anche comandi di STOP, i quali dovranno avere come effetto l'interruzione dell'attività del robot.

- R-TempKo: da questo requisito deduciamo che sia necessario fornire al robot informazioni sulle 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, cioè un entità posizionata nell'area di lavoro che si frappone tra il robot e il sonar finale e impedisce, tramite qualsiasi movimento possibile al robot, di raggiungere il sonar finale. Di conseguenza, in questa condizione, il robot deve arrestarsi.

– R-End: Come si capisce che il robot ha raggiunto il sonar? In base R-FloorClean sappiamo che è il sonar finale a rilevare la presenza del robot, quindi, una volta che il robot ha raggiunto il sonar finale deve arrestarsi

5.1 Use cases

5.2 Scenarios

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. A noi interessa modellare in maniera formale.

5.4 Test plan

6 Problem analysis

Scopo dell'analisi del problema è la produzione di un modello formale. Cosa significa? Che può essere eseguito su una macchina. Come può essere fatto? Attraverso un linguaggio (di modellazione, eventualmente) che riesca a catturare gli aspetti fondamentali del sistema, ovvero al giusto livello di dettaglio, mettendo in primo piano aspetti rilevanti, e lasciando in "background" dettagli. Il modello è espresso in termini di STRUTTURA, INTERAZIONE e COMPORTAMENTO. Limitatamente al nostro caso, ci è fornito un linguaggio di modellazione custom, QActor, grazie al quale modelliamo il sistema.

6.1 Logic architecture

6.2 Abstraction gap

6.3 Risk analysis

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