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

Ai problemi ci si può approcciare in modo olistico (top-down), o riduzionistico (bottom-up). Cosa intendiamo?
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 ipotesi 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". Quando si può dire se si è in presenza di una cosa del genere? Se nella business logic per ogni byte 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.
Si parla di "tecnology-lock" se l'applicazione è strettamente contaminata dalla tecnologia, ovviamente 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

Our current robot system can be controlled in remote way in different ways:

1. by an human user using the web interface provided by the robot executor;
2. by an human user using the web interface provided by the real robot;
3. by a machine that sends command messages to the robot or that emits command events that can be understood by the robot.

These multiple possibilities are very useful during software development and testing, but are source of confusion when we want to allow the usage of a robot as a resource conceptually owned by a single user and controlled by means of a single, certified interface.
Thus, we want extend our system with a set of new requirements:

1. The physical robot must expose in a visible way a Led and:
- the Led must be on when the robot is engaged by an user (human or machine);
- the Led must be off when the robot is available for booking.
2. the robot system does not expose any public available usage interface; 3. in order to use the robot, an user must first of all send 'to the system' a booking request. The system must return an answer including an access token if the robot

is available. If the answer is negative, (robot already engaged) and the request includes a 'notify-me flag', the system must notify the user when to robot becomes again available;
4. the user that receives the access token must send within a given acquisition-deadline (e.g. 30 sec) the request for a robot-driving command interface, by appending to the request the access token. If the acquisition-deadline expires, the robot returns in its 'available state';
5. the user can use the robot-driving command interface at most for a prefixed usage-duration time;
6. the user can explicitly release the robot resource by sending a booking release message;
7. if many users attempt to book the robot resource 'at the same time', the system could operate in two different ways:
(a) by selecting the first emitted request;
(b) by selecting the first received request;

5 Requirement analysis

La principale entità coinvolta in questo sistema è il robot fisico, cioè un componente descrivibile come un sotto-sistema controllabile remotamente, in grado di muoversi nello spazio fisico. 1. Un led è un componente elettronico in grado di emettere luce; il led dovrà essere collocato in maniera visibile sul robot fisico e collegato ad esso;
2. Il robot non dovrà essere accessibile pubblicamente come lo era in precedenza;
3. Il robot sarà controllabile da un solo utente alla volta, e, per poter controllare il robot, sarà necessario che l'utente utilizzi un sistema di prenotazione; questo sistema, in fase di robot libero, garantirà l'acceso immediato al controllo dello stesso, altrimenti porrà l'utilizzatore, nel caso di opportuno flag nella request, in uno stato di attesa che gli venga notificato il cambiamento di stato del robot;
Un access token è un elemento consegnato, dal sistema di prenotazione, all'utilizzatore quando gli viene concesso l'accesso al robot;
4. L'utente che comanderà il robot potrà essere sia una persona che un sistema software; L'utente in questione ha un tempo limite (deadline) per prendere il controllo dell'interfaccia di comando, utilizzando il token acquisito; Nel caso la deadline scada il robot torna disponibile e il token viene invalidato, cioè non più utilizzabile ai fini di controllo del robot;
5. Una volta preso il controllo l'utente potrà utilizzare il robot per un tempo massimo da stabilire;
6. L'utente ha la possibilità di cedere il controllo del robot prima della scadenza prefissata tramite apposita richiesta;
7. In caso di conflitto tra due richieste di prenotazione il sistema dovrà scegliere se dare precedenza alla prima richiesta emessa o alla prima richiesta ricevuta;

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. Nel caso non formale si descrivono le entità in gioco, mediante linguaggio naturale, in termini di struttura, interazione e comportamento.

1. Utente: entità che può essere uomo o macchina, in grado di interagire col sistema in qualità di utilizzatore finale. E' in grado di comandare il robot, utilizzando una intefaccia di comando che gli viene messa a disposizione;
2. Robot: dispositivo fisico in grado di ricevere comandi e di muoversi di conseguenza; E' dotato di ruote che gli permettono il movimento nello spazio fisico, di motori che azionano le ruote stesse, di un sonar posto nella parte anteriore e di un led visibile. E' composto, a basso livello, da un edge, cioè un elemento che rende "smart" e connesso a internet il robot stesso. E' dotato anche di un middleware, architettura computazionale che rende possibile l'interazione tra il basso livello e il cloud. La parte cloud non è presente fisicamente sul robot ma su server esterni ad esso; Per comandare il robot è necessario inviare ad esso comandi opportunamente strutturati;
3. Booking system: è un sistema software che comunica via rete, in grado di gestire richieste di utilizzo del robot da parte degli utenti. In fase di ricezione della richiesta deve controllare lo stato del robot, e, nel caso esso sia disponibile, concedere un token di accesso al richiedente, mentre nel caso il robot sia occupato deve tenere conto dell'identità dell'utilizzatore che si è prenotato e notificarlo quando il robot si rende disponibile;
4. Led: è un componente elettronico in grado di emettere luce in seguito a determinati impulsi. E' collegabile ad un middleware e pilotabile dallo stesso.;

Utilizziamo il linguaggio formale di modellazione QActor per modellare le entità che comunicano tramite rete, mentre le entità più semplici vengono modellate come POJO.

Modello formale del robot fisico: realRobotExecutor.qa (-httpserver flag rimosso perchè si vuole passare dal sistema di prenotazione per richiedere i diritti di accesso all'interfaccia di comando del robot)
Modello formale del robot virtuale: virtualRobotExecutor.qa (-httpserver flag rimosso perchè si vuole passare dal sistema di prenotazione per richiedere i diritti di accesso all'interfaccia di comando del robot)
Modello formale dell'utente-macchina: mindOfRobot.qa
Modello formale dell'utente-umano: humanUser.qa
Modello formale booking system: bookingSystem.qa

Il led viene modellato come POJO, tramite interfaccia da implementare.

5.4 Test plan

6 Problem analysis

Dove risiede il servizio di booking?
Cosa succede nel caso di malfunzionamento (suo o della rete)?
Come gestire il caso in cui l'utente si "prenota" correttamente ma cade senza inviare alcun comando?
Come stabilisco i tempi di timeout in fase di prenotazione o di invio comandi?
E cosa cambia se dimuisco o aumento questi tempi?
Cosa cambia se dò precedenza alla prima richiesta emessa o alla prima ricevuta (in caso di richieste arrivate contemporaneamente al sistema di prenotazioni)?
A chi viene data la precedenza nel caso di robot che si libera, tra gli utenti che sono in attesa di notifica?
Cosa succede se l'utente notificato del fatto che il robot è tornato disponibile non è attivo?

6.1 Logic architecture

6.2 Abstraction gap

Il linguaggio per descrivere il modello (QActor) risulta, in questo caso, un po' limitato per modellare le entità in gioco, infatti esso non possiede il concetto di utente esterno.

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

8.1 Structure

8.2 Interaction

8.3 Behavior

9 Implementation

10 Testing

11 Deployment

12 Maintenance

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

13 Information about the author

Photo of the author

References