Drone mission

Carlo Antenucci, Leonardo Iannacone, Gonzalo Junquera

17 marzo 2013

Indice

Elenco degli algoritmi

Capitolo 1

Introduzione

Capitolo 2

Visione

Lo sviluppo di un prodotto software necessita di un processo di produzione maturo, che, al fine di garantire un'elevata qualità e produttività, necessita di una opportuna organizzazione. Per migliorare il processo produttivo il *Software Engineer Institute* (`SEI`) ha introdotto il sistema *Capability Maturity Model* (`CMM`). Tale sistema suddivide le organizzazioni in cinque fasi:

Livello 1: *Initial* (Chaotic): i processi sono ad-hoc, caotici, o pochi processi sono definiti

Livello 2: *Repeteable*: i processi di base sono stabiliti e c'è un livello di disciplina a cui attenersi in questi processi

Livello 3: *Defined*: tutti i processi sono definiti, documentati, standardizzati ed integrati a vicenda

Livello 4: *Managed*: i processi sono misurati raccogliendo dati dettagliati sui processi e sulla loro qualità

Livello 5: *Optimized*: è in atto il processo di miglioramento continuo tramite feedback quantitativi e la fornitura di linee guida per nuove idee e tecnologie

La costruzione di un software, inoltre, è spesso legata alle piattaforme operative su cui il prodotto dovrà operare, che in ogni caso hanno una espressività molto maggiore della *Macchina di Minsky*. Per questo motivo, solitamente, le organizzazioni tendono ad utilizzare approcci diversi per la produzione del software. Le possibili strategie prevedono:

- l'elaborazione di una soluzione partendo da un'analisi del problema, che porta alla stesura di un codice ad hoc per quel determinato contesto (*Top Down*)
- lo sviluppo di una soluzione utilizzando le funzionalità messe a disposizione di una tecnologia (*Bottom Up*)

- la realizzazione di un modello del sistema software da realizzare in modo tale da rendere il prodotto che si sta sviluppando indipendente dalla tecnologia e, allo stesso tempo, riutilizzabile in più contesti (*Model Driven Software Development*)

Le figure professionali che entrano in gioco all'interno di un processo di produzione software sono principalmente tre:

Project manager è colui che coordina lo svolgimento del progetto. Avvalendosi di consulenze tecniche prenderà decisioni in merito alle risorse necessarie per il progetto e distribuirà i compiti agli altri due soggetti in gioco definendo cosa dovrà essere realizzato e come.

System designer è colui che specifica cosa il sistema software deve essere in grado di fare. Il suo compito è quello di specificare la struttura del sistema, i suoi componenti, le sue interfacce ed i suoi moduli. Nell'approccio *Model Driven Software Development* il lavoro che svolge consiste nel modellare le entità del sistema su tre dimensioni: struttura, interazione e comportamento.

Application designer è colui che specifica come le entità descritte dal system designer interagiscono e si comportano al fine di ottenere quanto richiesto dal committente. Il suo compito è, quindi, quello di definire, utilizzando gli strumenti messi a disposizione dalla tecnologia scelta, ed eventualmente dal system designer stesso, la business logic del sistema software.

Il compito del system designer, quindi, è quello di realizzare un modello concettuale del sistema, definendo come detto le entità che entrano in gioco, le loro interazioni e il loro comportamento, e fornire all'application designer un meta-modello dello stesso.

L'idea di utilizzare un approccio Model Driven, anziché uno Top Down o Bottom Up, consente, quindi, lo sviluppo di un prodotto software, non legato in maniera eccessiva alla tecnologia alla base del sistema, né tanto meno alla business logic. Questo garantisce al sistema sviluppato un alto grado di riutilizzabilità in quanto, una volta definito il modello, sarà sufficiente modificare il comportamento o l'interazione dei componenti per far si che questo si adatti ad un nuovo problema. Inoltre l'approccio Model Driven consente di formalizzare ed esplicitare, attraverso la costruzione di una serie di diagrammi che non lasciano spazio ad ambiguità, le conoscenze utili alla risoluzione del problema da risolvere.

Capitolo 3

Obiettivi

L'applicazione intende fornire alla protezione civile un maggiore supporto per l'esplorazione territoriale senza la necessità di mettere a repentaglio vite umane.

Capitolo 4

Requisiti

4.1 Lo scenario applicativo

La protezione civile decide di inviare su un luogo diffcilmente accessibile un aeromobile senza pilota (*drone*), capace di operare in modo teleguidato. Il drone è dotato di un insieme di *sensori di stato* in grado di rilevare la velocità corrente (*speed*) e il carburante disponibile (*fuel*). Il drone dispone anche di un dispositivo *GPS* in grado di determinarne la posizione in termini di latitudine e longitudine.

Il compito del drone è scattare fotografie del territorio ogni DTF (DTF>0) secondi e inviare le immagini a un server installato presso una unità operativa. Il server provvede a memorizzare le immagini ricevute (in un file o in un database) associandole ai dati dei sensori di stato disponibili al momento dello scatto della foto. Il server provvede inoltre a visualizzare su un display dell'unità operativa i valori di stato ricevuti dal drone in una dashboard detta *DroneControlDashboard*.

La *DroneControlDashboard* viene concepita come un dispositivo composto di due parti: una parte detta *GaugeDisplay* e una parte detta *CmdDisplay*. La parte *GaugeDisplay* della *DroneControlDashboard* visualizza i dati provenienti dai sensori del drone riconducendoli ciascuno a uno specifico strumento di misura; uno *Speedometer* (velocità in km/h) un *Odometer* (numero di km percorsi) un *FuelOmeter* (livello corrente di carburante in litri) e un *LocTracker* (posizione del drone). La *GaugeDisplay* può visualizzare i dati in forma digitale e/o grafica; la posizione viene preferibilmente visualizzata fornendo una rappresentazione del drone su una mappa del territorio. La parte *CmdDisplay* della *DroneControlDashboard* include pulsanti di comando per fissare la valocità di crociera (*setSpeed*) avviare (*start*) e fermare (*stop*) il drone[1] e per incrementarne (*incSpeed*) e decrementarne (*decSpeed*) la velocità corrente di una quantità prefissata DS (DS>0 km/h).

[1] Il drone si suppone abbia un sistema di controllo capace di eseguire i comandi di *start* e di *stop* in modo opportuno.

I dati dei sensori del drone sono anche resi disponibili sugli smart device in dotazione al responsabile della protezione civile (*Chief*) e al comandante (*Commander*) della unità operativa. Ogni smartdevice provvederà a visualizzare (su richiesta dell'utente) i dati in una dashboard (*SmartDeviceDahboard*) opportunamente definita per lo specifico dispositivo, preferibilmente in modo analogo alla *GaugeDisplay*.

Il server deve operare in modo che :

- la missione del drone possa iniziare solo dopo che il drone ha dato conferma della ricezione del comando *setSpeed* che fissa la velocità iniziale di crociera;
- la speed del drone sia sempre compresa tra due valori-limite prestabiliti `speedMin` a `speedMax`[2];
- all'avvio di ogni missione, ogni smartdevice `Android` sia messo in grado di generare una *notification* all'utente, la cui selezione provvede ad aprire una applicazione che mostri la *SmartDeviceDahboard*.
- gli smartdevice siano in grado di visualizzare lo stato del drone anche in caso di guasto del server centrale.
- il comando di stop sia inviato in modo automatico non appena il livello del carburante risulta inferiore a un livello prefissato `MinFuel`.

4.2 Il lavoro da svolgere

In questo quadro, si chiede di definire il software da installare sul server della unità operativa e su smartdevice dotati di sistema operativo Android[3]. Opzionalmente: si chiede di definire uno strumento capace di visualizzare le informazioni memorizate dal server dopo una missione del drone. Si chiede anche di costruire un opportuno simulatore delle attività del drone con riferimento ai seguenti parametri:

```
          Parametri per la simulazione del drone
DTF=5 sec, DS=10 km/h, livello fuel iniziale = 30 litri
livello minimo fuel per operatività:  MinFuel = 0,5 litri
speed di crociera compresa tra:  speedMin=60 e speedMax=120 km/h
consumo di carburante = (speed * 30) litri/h
percorso del drone:  in linea retta a una quota fissa di 100m.
```

4.3 Remark

Si ricorda che l'obiettivo del lavoro non è solo la produzione di un sistema software in grado di soddisfare i requisiti funzioniali ma anche (e in primis)

[2] Le fasi di decollo e atterraggio sono qui ignorate.

[3] Per il primo protoipo lo smartdevice può essere un computer convenzionale.

il rapporto tra il prodotto e il processo adotttato per generarlo. 2 Le fasi di decollo e atterraggio sono qui ignorate. 3 Per il primo protoipo lo smartdevice può essere un computer convenzionale

Capitolo 5

Analisi dei requisiti

5.1 Use cases

Dalle specifiche del committente si è capito che questi desidera il sistema fornisca tre funzionalità principali: ricezione di informazioni territoriali, controllo della missione e ricezione di informazioni relative ai sensori di stato del drone.

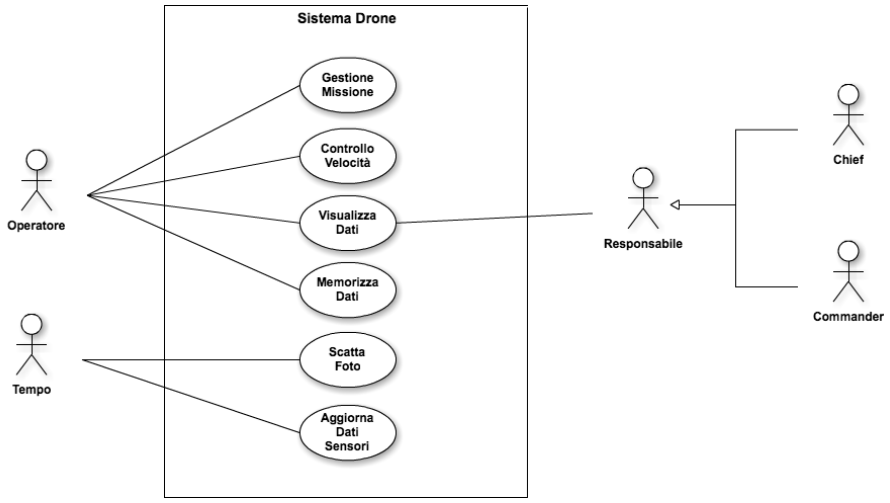

Figura 5.1: Use case

5.2 Glossario

Termine	Significato
Centrale Operativa	**Elemento esterno al sistema da sviluppare. Ha il compito di controllare la missione, ricevere dal drone informazioni relative ai suoi sensori e memorizzare le fotografie scattate.**
Smartphone	**Elemento esterno al sistema. Consente al Chief e al Commander di avere informazioni sullo stato del drone.**
Drone	**Elemento esterno al sistema. È un velivolo privo di pilota che ha il compito di esplorare un territorio difficilmente accessibile, di comunicare i dati relativi ai suoi sensori e di inviare ad intervalli ditempo regolari fotograrfie dell'ambiente esplorato.**
Fotografie territoriali	**Immagine jpg, acquisite dal drone tramite una fotocamera, che riposta informazioni relative alle condizioni ambientali del luogo esplorato e chela centale operativa provvederà a memorizzare.**
Sensori	**Elementi attivi del sistema. Inviano alla centrale operativa informazioni sullo stato del drone, quali chilometri percorsi, velocità attuale, quantità di carburante residuo e le coordinate geografiche del punto in cui si trova.**
DroneControlDashboard	**Elemento del sistema che consente alla centrale operativa di visualizzare le informazioni ricevute dal drone (GaugeDisplay) e, allo stesso tempo, di inviare al velivolo comandi (CmdDislpay).**
GaugeDisplay	**Componente della DroneControlDashboard che consente la visualizzazione delle inforamzioni del drone. È composta da una mappa su cui viene visualizzata la sua posizione e da tre strumenti di misura che riportano i dati rilevati dei sensori, sia in forma analogica che digitale.**
Odometer	**Strumento di misura che consente la visualizzazione del numero di chilometri percorsi dal drone.**
Speedometer	**Strumento di misura che consente la visualizzazione della velocità attuale del drone**
FuelOmeter	**Strumento di misura che consente la visualizzazione della quantità di carburante disponibile nei serbatoi del drone.**

5.3 Scenari

5.4 (Domain) Model

5.5 Test plan

Capitolo 6

Analisi del problema

6.1 Logic architecture

6.2 Abstraction gap

6.3 Risk analysis

Capitolo 7

Piano di lavoro

Capitolo 8

Progetto

8.1 Struttura

8.2 Interazione

8.3 Behavior

Capitolo 9

Implementazione

Capitolo 10

Testing

Capitolo 11

Deployment

Capitolo 12

Maintenance