Task di Analisi

a.a. 2017-2018

1 Il caso di studio

Un'agenzia turistica avente sede in un luogo vacanziero organizza escursioni. Le escursioni hanno durata giornaliera e per semplicità sono di due tipi: gite in barca e gite a cavallo.

Per ogni escursione sono previsti alcuni optional acquistabili dall'eventuale partecipante. I tipi di optional possibili sono tre: pranzo, merenda, visita a un sito. I tipi di optional associati a una data escursione possono differire da caso a caso. Per esempio: la gita in barca del giorno *x* può prevedere il pranzo e la merenda, la gita in barca di due giorni dopo può prevedere il solo pranzo, l'escursione del giorno *y* può non prevedere optional.

Il prezzo degli optional è fisso ed è determinato solo dal tipo. Invece, ogni singola escursione ha un suo prezzo. Ogni escursione prevede un numero massimo di partecipanti. Il prezzo di una escursione, il limite ai partecipanti e tutto ciò che corrisponde alla definizione dell'escursione stessa viene fissato all'atto della sua immissione nel programma dell'agenzia. Questi dati possono anche essere modificati, in corso di esercizio, dopo che essi sono stati immessi.

Si deve realizzare un sistema in grado di gestire le prenotazioni. A tal fine il sistema deve:

1. Consentire di inserire, eliminare, modificare le escursioni nel programma stagionale dell'agenzia.
2. Permettere di registrare un partecipante ad una data escursione, consentendo, nel caso siano previsti, la scelta di eventuali optional; calcolare il costo dell'escursione comprensivo degli optional.
3. Cancellare un partecipante già iscritto da una data escursione.
4. Aggiungere o eliminare un optional relativamente a un dato partecipante e una data escursione; calcolare il nuovo prezzo dell'escursione.

Osservazione

Una specifica come quella precedente è quanto un ingegnere del software può aspettarsi nella pratica del mestiere. Un esame anche superficiale della medesima mostra che essa è molto approssimata e certamente incompleta. Per esempio, si afferma che è possibile eliminare o modificare una escursione già a programma. In un contesto reale l'eliminazione di una escursione avrebbe conseguenze non trascurabili: avvisare gli iscritti, rimborsarli di quanto pagato, tenere conto della restituzione del danaro nella contabilità, ecc. Si tratta di funzionalità che in un contesto reale non possono essere trascurate. Lo scopo dell'analisi è proprio quello della loro completa identificazione. Risulta evidente che l'analista deve coinvolgere in questo processo il committente e gli utenti finali del sistema.

2 Il metodo seguito: manufatti da realizzare

Il metodo di analisi da applicare prevede la realizzazione di un documento i cui elementi sono di seguito elencati. Si precisa che i diagrammi richiesti devono essere realizzati con il tool Visual Paradigm e consegnati in un unico file "vpp".

1. Analisi dei requisiti
   - Specifica dei requisiti funzionali (scopo, requisiti funzionali, vincoli)
   - Costruzione del modello di dominio (diagramma delle classi di dominio)
   - Analisi dei casi d'uso (diagramma dei casi d'uso, scenari dei casi d'uso)
   - Validazione requisiti–casi d'uso (matrice di tracciabilità)

2. Analisi/Progetto preliminare
    - Analisi di robustezza (stereotipazione BCE: tracciabilità classi - casi d'uso)
    - Raffinamento dei casi d'uso e eventuali aggiornamenti al modello di dominio (diagrammi di sequenza)
    - Modello dei dati (modello E-R)

3 La specifica dei requisiti funzionali

Con il termine "requisiti" si intende l'insieme delle proprietà che un prodotto deve possedere per soddisfare al proprio scopo di uso.

Tradizionalmente i requisiti di un prodotto software vengono raccolti in un documento denominato SRS (specifica dei requisiti software, ovvero *software requirement specification*). Talo documento ha lo scopo di definire esattamente **cosa** il sistema farà (non come **come** lo farà).

Lo standard IEEE 830 (ultima edizione 1998)[1] definisce i criteri (e il formato) secondo cui deve essere redatta una specifica dei requisiti. Essenzialmente, lo standard prevede l'elencazione dei requisiti funzionali e propone alcuni metodi secondo cui essi devono essere descritti. Purtroppo lo standard in questione è piuttosto vecchio e non contempla il ricorso ai casi d'uso, che, almeno da quando si è diffuso UML, costituiscono la tecnica preferita per specificare il "**cosa**".

Il metodo delineato al Paragrafo 2 è fortemente influenzato dall'analisi dei casi d'uso. Tuttavia, esso prevede, come fase iniziale, la stesura delle specifiche in forma testuale, secondo tradizione consolidata. Si ritiene infatti che sia comunque utile stendere un elenco ordinato e strutturato dei requisiti, in forma testuale. Del resto, nella pratica professionale, l'elencazione dei requisiti è sempre il passaggio iniziale attraverso cui l'analista software e il committente arrivano a concordare cosa ci si aspetta dal sistema. Spesso si parla di "raccolta" dei requisiti, proprio per evidenziare che si tratta di un processo tramite il quale si cerca di ottenere, rendere manifeste tutte le caratteristiche che il sistema deve possedere.

Dopo le fasi di raccolta dei requisiti e di analisi dei casi d'uso è buona norma incrociare i requisiti con i casi d'uso, individuando da quali casi d'uso sono "coperti" i vari requisiti, al fine di poter verificare se niente è stato tralasciato.

Di seguito i tre paragrafi che identificano lo scopo, i vincoli e i requisiti funzionali.

- Scopo. La corretta identificazione dello scopo è importante in fase di analisi, in quanto ci aiuta a capire quali sono le entità da modellare e come vanno modellate (evidentemente la scelta deve cadere su entità che hanno senso rispetto allo scopo che il sistema si prefigge).
- Vincoli. I vincoli sono condizioni o proprietà di carattere generale che il sistema deve rispettare
- Requisii funzionali. Funzionalità che il sistema deve esibire, elencate in modo ordinato e non ambiguo.

Specifica dei requisiti

Scopo

Lo scopo del sistema è la gestione delle prenotazioni dell'agenzia.

Vincoli

- Il sistema viene realizzato come installazione unica, per funzionare all'interno di una agenzia avente una sola sede.
- Tutte le operazioni vengono effettuate dal solo personale dell'agenzia e non c'è necessità di distinguere i differenti impiegati. Ciò equivale ad assumere che ci sia un solo attore.

Requisiti funzionali

(elencare i requisiti funzionali)

Vincoli ulteriori

(elencare le "assunzioni", "limitazioni")

4 Il modello di dominio

Il metodo seguito prevede che la costruzione del modello del dominio applicativo preceda l'analisi dei casi d'uso.

Non è questa una scelta convenzionale. Infatti, in larga parte della letteratura si propone di far precedere la costruzione del modello di dominio dall'analisi dei casi d'uso. Ciò può portare alla stesura di casi d'uso troppo astratti, non collegati allo specifico problema, di poca utilità per il programmatore.

In questa fase non è necessario che costruire un modello di dominio dettagliato in ogni suo aspetto, bensì costruire un modello che evidenzi le principali relazioni tra le entità applicative. Quanto dettagliare il modello è una questione di sensibilità e di utilità. Conviene limitarsi agli aspetti strutturali (le relazioni tra le classi), utili nella fase di *robustness analysis*, senza addentrarsi troppo nella definizione degli attributi né tantomeno dei metodi, da rinviare alla fase di rifinitura del modello e di progettazione dettagliata.

Costruzione del modello di dominio

L'analisi del testo del problema permette di evidenziare un certo numero di sostantivi: escursione, programma, optional, gita in barca, costo, ecc. Essi individuano concetti che devono essere trasferiti nel modello, in forma di classe o in forma di attributo. Per esempio, il sostantivo "escursione" è certamente deputato a rappresentare una classe del modello, mentre termini come "costo" e "numero massimo di partecipanti" sono manifestamente attributi della stessa escursione.

5 Analisi dei casi d'uso

(Diagramma dei casi d'uso, scenari dei casi d'uso)

Validazione dei casi d'uso

Al termine dell'analisi occorre effettuare un esame critico al fine di validare quanto realizzato. Questo esame deve essere condotto dall'analista assieme agli utenti/committenti del sistema, in modo che sia evitata la spiacevole situazione in cui vengono a trovarsi non pochi progetti di sviluppo software, quando, in fase avanzata, ci si accorge che quel che fa il sistema (quello che è stato realizzato) non corrisponde a quello che il cliente/committente si aspettava.

Dopo la necessaria validazione non è detto che il lavoro sui casi d'uso sia finito: il passo successivo del processo, l'analisi di robustezza, può evidenziare la necessità di tornarci sopra al fine di eliminare incoerenze ed ambiguità nascoste.

Si precisa che se si mettessero assieme e si riorganizzassero le descrizioni dei casi d'uso (trasformando frasi come "l'attore preme il pulsante ... il sistema mostra la finestra ..." in "premere il pulsante ... viene mostrata la finestra ..."), si otterrebbe il manuale utente.

Eseguire l'analisi dei casi d'uso vuol dire definire esattamente come il sistema si comporterà; molto prima di dar corso alla sua realizzazione.

E' bene che i casi d'uso siano esaminati criticamente dall'analista assieme agli utenti/committenti del sistema.

Un aspetto importante da verificare è se tutti i requisiti sono stati considerati. A tale scopo conviene incrociare i requisiti con i casi d'uso al fine di stabilire se tutti i requisiti sono "coperti" dai casi d'uso.

La verifica di copertura può essere fatta ricorrendo ad una tabella come quella di seguito presentata.

	R1	R2	R3	R4
CU1	X	X		
CU2			X	
CU3				X
CU4				

6 Analisi di robustezza

Ora si svolge l'analisi del come siano ottenute le funzionalità che il sistema deve presentare. Si tratta della cosiddetta analisi di robustezza (*robustness analysis*). Essa consiste, per ciascun caso d'uso nell'identificare le funzioni da svolgere e le classi del modello che intervengono. Le funzioni verranno poi allocate a eventuali classi applicative aggiuntive e, in parte, quando sia conveniente, alle classi del modello di dominio. Lo scopo dell'analisi di robustezza è collegare l'analisi dei casi d'uso alla

specifica dettagliata di progetto in cui vengono definite le operazioni di sui è responsabile ciascuna classe.

Nel fare l'analisi si seguirà questo criterio: inizialmente si identificano le funzioni applicative prevedendo per ciascuna di esse uno specifico controllore, successivamente si accorpano più funzioni applicative in un medesimo controllore. Nel fare l'analisi approfondiremo anche il ruolo delle classi del modello. Ciò potrà suggerire (piccole, sperabilmente) variazioni al modello stesso.

L'analisi fa riferimento ai tre stereotipi (di analisi) Boundary, Controller, Entity.

- Entity (entità): classi di oggetti che rappresentano la semantica del dominio applicativo;
- Controller (controllori): classi di oggetti che determinano il modo in cui l'applicazione risponde agli input degli attori;
- Boundary (Viste, Interfacce): classi di oggetti che rappresentano l'interfaccia tra gli attori e il (resto del) sistema.

Regole

Devono essere seguite queste regole:

- Entità e viste possono comunicare con i controllori (e viceversa), ma non fra di loro.
- I controllori possono comunicare con entità e viste (e viceversa) e possono comunicare tra di loro. Queste regole sono dettate dall'idea di identificare le funzioni.

Si tratta ora di considerare ciascun caso d'uso e tracciare per esso il relativo diagramma di analisi.

Si noti che nello sviluppare l'analisi si utilizza il criterio di mettere in evidenza le funzioni necessarie al caratterizzare l'esecuzione dei casi s'uso, evitando i dettagli inutili, per non perdere la focalizzazione sugli aspetti caratterizzanti il problema. Ciò può comportare una parziale riscrittura dei casi d'uso.

L'analisi di robustezza potrebbe mettere in mostra la necessità di apportare modifiche anche al modello di dominio e di maggiori interventi sui casi d'uso.

L'analisi si conclude con la verifica che tutti i casi d'uso sono stati trattati e che le opportune modifiche sono state apportate.
Il passo successivo è la stesura dei diagrammi di sequenza, in modo da identificare le reponsabilità delle classi, ovvero i metodi di interfaccia. I diagrammi possono essere stesi a partire dai diagrammi di robustezza, rappresentando le funzioni identificate come altrettanti controllori indipendenti.

Poiché nel fare l'analisi si può scoprire che il sistema si compone dei controllori associati alle differenti schermate e da eventuali altri controllori che interagiscono direttamente con gli oggetti del modello, conviene prevedere sin d'ora quali saranno questi possibili controllori e allocare ad essi le funzioni applicative identificate (Inserisci, Rimuovi, ecc.).

7 Specifica delle classi (diagrammi di sequenza)

(Elencare i diagrammi di sequenza).

Dettaglio dei diagrammi di sequenza

I diagrammi di sequenza sono un ottimo strumento per capire come avvengono le interazioni tra gli oggetti. Ma non si deve pensare di poterli usare fino a "definire" ogni operazione. Cercare di definire tutte le operazioni delle classi attraverso i diagrammi di sequenza sarebbe una inutile perdita di tempo. I diagrammi di sequenza devono darci le interazioni fondamentali, le operazioni di carattere secondario e di appoggio si fissano meglio al tempo di progettazione. Al tempo di progettazione certe funzionalità possono essere raggruppate o suddivise o spostate da una classe all'altra a seconda della convenienza. Ciò porta facilmente a rimettere in discussione eventuali scelte di dettaglio.

Quanto si debba andare a fondo è in larga parte dettato dalla sensibilità dell'analista e dal contesto in cui opera (quanto è esperto e capace il progettista, se diverso dall'analista).

In fase di progettazione converrà equipaggiare le classi del modello con metodi `get` e `set` degli attributi.

8 Modello dei dati

(Modello E-R dei dati).

Task di Progettazione ed implementazione

a.a. 2017-2018

Progettare ed implementare cinque pattern, a propria scelta, tra quelli presentati a lezione, in linguaggio Java.

La progettazione di ogni pattern prevede la realizzazione di:

- un diagramma delle classi, con indicazione della strategia adottata
- descrizione della responsabilità delle classi
- un diagramma di sequenza

Manufatti da consegnare:

1. n. 1 file Visual Paradigm contenente i manufatti di progettazione richiesti
2. n. 1 file di progetto Eclipse contenente il codice delle 5 implementazioni