UNIVERSITA' DEGLI STUDI DI PADOVA

Corso di Ingegneria del Software 2017/2018

IRONWORKS

Costruire software robusto

ZUCCHETTI

IL SOFTWARE CHE CREA SUCCESSO

Oggetto dell’ appalto

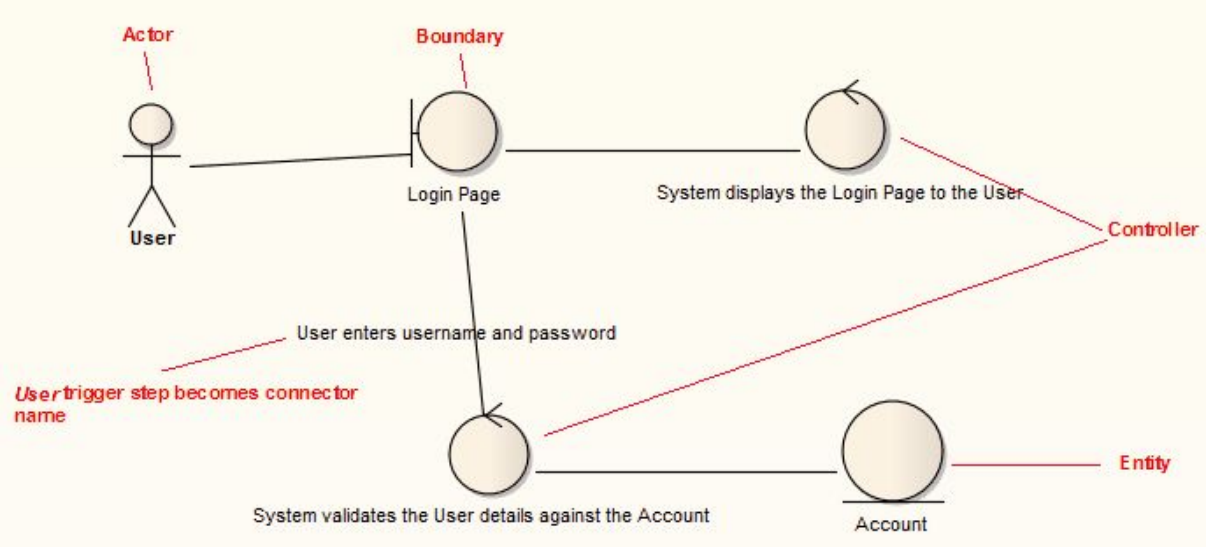
Il presente capitolato ha per oggetto l’affidamento della fornitura per la realizzazione di un software di costruzione di diagrammi di “robustezza” con la relativa generazione di codice Java per le entità persistenti.

Il software oggi permea quasi ogni attività umana: tutti i settori merceologici, qualunque sia l’oggetto prodotto, possono trovare il modo di aumentare il servizio per l’utente e l’esperienza d’uso sfruttando le possibilità offerte dall’informatica. Questo fenomeno è noto con il nome di “trasformazione digitale”.

La costante richiesta di nuovo software si scontra con la cronica mancanza di esperti e con la bassa qualità del software prodotto; per aumentare la qualità e la velocità di produzione occorre rendere questa attività un processo industriale ingegnerizzato allontanandosi dall’ artigianalità che ancora a volte lo caratterizza.

La progettazione del buon software, in modo controllato e con alti standard qualitativi, vede nell UML la proposta di molti diagrammi che possono aiutare tutte le fasi dello sviluppo del software, sia nella forma di “sketch” che in quella di “blueprint”.

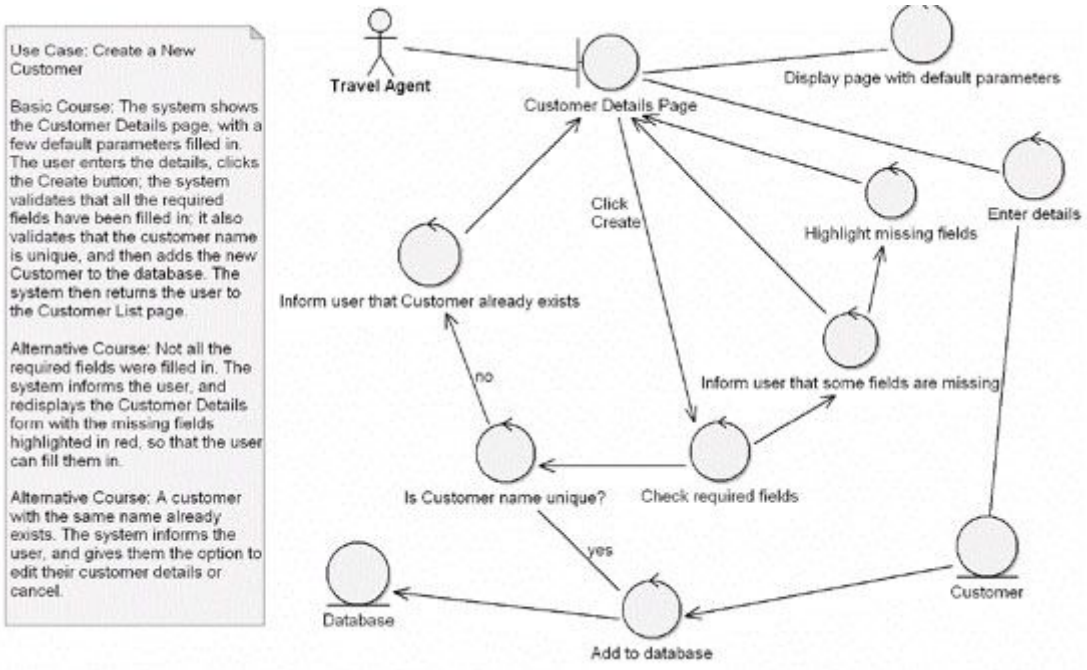
Nello standard UML è previsto un diagramma, il “robustness diagram”, poco conosciuto e ancor meno utilizzato. E’ un diagramma che segue immediatamente i casi d’ uso, infatti faceva parte della proposta “Objectory” di Ivar Jacobson.
Confluendo nell’UML è stato schiacciato dall’onnipresenza del “class diagram” che è utilizzabile anche per le fasi di analisi.



Nel diagramma di robustezza vengono indicati tre tipi di oggetti: le interfacce, le procedure e le entità persistenti. Questa divisione può essere fatta fin dalle prime fasi di sviluppo e catalogando gli oggetti che via via si scoprono secondo questa tassonomia è molto più probabile costruire dei programmi ben organizzati e “robusti” rispetto agli errori ed ai cambiamenti nel tempo di quanto si può ottenere procedendo senza questa guida.

Identificare e isolare le interfacce, sia utente che programmatiche (“boundary objects”), riconoscere le procedure che elaborano i dati che ricevono dalle interfacce (“control objects”), separare la gestione delle entità persistenti (“entity objects”) conduce in modo naturale alle architetture Model-View-Controller, Model-View-Presenter e Model-View-Viewmodel che hanno dimostrato di essere le più valide in programmi che interagiscono con gli utenti.

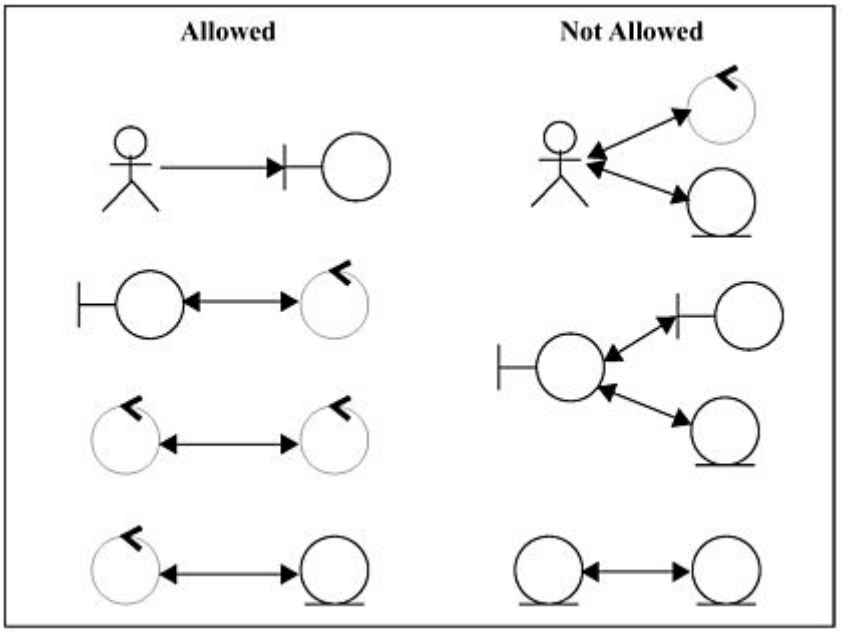
In un certo senso questi tre “pattern” costituiscono la forma più vasta possibile di un altro strumento, l’analisi di dominio, che aiuta ad affrontare i problemi con il supporto di un pregresso ragionato ed organizzato. In questo caso il dominio è l’intera informatica e i tre “archetipi” sono le suddivisioni più naturali per qualsiasi problema possiamo incontrare.



Con questo appalto l’azienda Zucchetti chiede di realizzare un disegnatore di diagrammi di robustezza, estendendo la definizione delle entità persistenti in modo che sia possibile generare il codice sia delle classi Java che possono contenerle che dei programmi per scriverle e leggerle in un database relazionale.

Caratteristiche e Requisiti Obbligatori

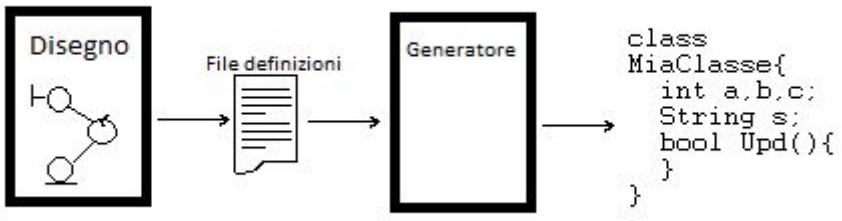
Il sistema che richiediamo sarà composto di un disegnatore di diagrammi di robustezza e di un generatore di codice che a partire dalle definizioni contenute in un diagramma deve produrre le classi Java per ospitare i dati delle entità persistenti ed i metodi per leggere e scrivere queste classi in un database relazionale.



Il software richiesto dovrà svolgere almeno i seguenti compiti:

1. Realizzare il diagramma di robustezza
2. Estendere le entità persistenti includendo la descrizione dei dati contenuti; devono essere trattati i tipi stringa, numero, data, logico e gli array monodimensionali di tipi di base
3. Produrre le classi Java per ospitare i dati previsti dalle entità persistenti
4. Produrre i metodi di scrittura e lettura verso un database relazionale
5. Produrre le istruzioni di gestione della transazione in modo che la scrittura dell'entità avvenga in modo atomico
6. Produrre il codice di creazione delle tabelle associate alle entità persistenti nel database relazionale
7. Produrre il codice di manutenzione delle tabelle nel database relazionale quando viene variata la definizione delle classe
8. Definire l'architettura completa dell'applicazione con le regole per sviluppare manualmente il codice degli oggetti "boundary" e "control" che possano utilizzare le classi generate in automatico

Il ciclo di produzione dovrà essere:



Ad esempio, disegnato un diagramma ed aumentate le specifiche di una “entità” con la lista dei campi memorizzati, lanciando il generatore di codice si dovrebbero ottenere del sorgente di questo tipo:

file: entità.java

```
class MiaEntita extends Entita {
  public String codice;
  public String nome;
  public String cognome;
  public int eta;
  public int votoDiLaurea;
  public boolean laureaConLode;
  public boolean create(){
      …
     sql.exec(“insert into miaentita (codice,nome, … ) values
(?,?, …) ”,codice,nome,...);
      …
  }
  public boolean update(){ … }
  public boolean delete(){...}
}
```

file: script.sql

```
CREATE TABLE MIAENTITA (
  CODICE:CHAR(10) PRIMARY KEY,
  NOME:CHAR(20),
  COGNOME: CHAR(20),
  ETA:NUMERIC(3,0),
  VOTODILAUREA:NUMERIC(3,0),
```

```
  LAUREACONLODE: BIT
)
```

Il sistema dovrà essere realizzato con tecnologie Web. In particolare richiediamo che la parte server venga realizzata in Java con server Tomcat o Javascript con server Node.Js. La parte client dovrà essere eseguibile in un browser HTML5 ed utilizzare fogli stile CSS per l'aspetto estetico e Javascript per la parte attiva.

Requisiti Opzionali

Il software potrebbe avere le seguenti caratteristiche:

1. Definire le entità "singleton", cioè archiviate in singola copia su database o su file di testo (es: entità che rappresentano file di configurazione in XML o JSON)
2. Implementare la descrizione della struttura delle classi persistenti utilizzando il formalismo di Warnier-Orr
3. Specificare delle regole di calcolo e controllo per le classi persistenti generando il codice di verifica
4. Estendere il modello transazionale in modo che un oggetto "control" possa definire transazioni che trattano più di una entità persistente
5. Utilizzare un ORM già esistente (es. Hibernate) per gestire la persistenza creando i file di configurazione anziché il codice di gestione del database e delle frasi SQL.
6. Estendere la descrizione degli oggetti "boundary" in modo che sia possibile generare anche i programmi di "data entry".
7. Produrre il codice prototipale per gli oggetti "procedura" in modo da ottenere il sistema completo di data entry ed archiviazione
8. Estendere la descrizione in modo da modellare le referenze tra entità e programmare corrispondentemente i vincoli di integrità referenziale

A titolo di ispirazione ed esempio segnaliamo l'esistenza di parecchi software, anche open-source, che permettono di disegnare diagrammi UML e alcuni hanno anche il diagramma di robustezza, mentre altri ricorrono al diagramma delle classi con sterotipi.
Nell'ambiente desktop indichiamo ArgoUML, StarUML, Software Idea Designer, UMLet e PlantUML.
In ambiente web parecchi disegnatori permettono la costruzione di diagrammi UML, come ad esempio LucidChart.

Variazioni ai requisiti

In corso d'opera non sarà possibile variare/modificare i requisiti minimi (obbligatori per accettare il prodotto). Sarà invece possibile variare i requisiti opzionali, in quanto saranno i gruppi vincitori dell'appalto a modificarli / eliminarli / aggiungerli.

Documentazione

Il progetto dovrà essere supportato dalla documentazione minima richiesta per il corso di Ingegneria del software e dovrà essere fornito un manuale per l'utilizzo ed un manuale per chiunque voglia estendere l'applicazione.

Garanzia e Manutenzione

L'azienda Zucchetti SPA è interessata a questo progetto come dimostrazione della fattibilità dell'obiettivo utilizzando le tecnologie web. Costituirà titolo preferenziale nella valutazione delle proposte la pubblicazione del progetto sul sito “github.com” o altri repository pubblici, in conformità con i relativi requisiti di natura open-source, per favorire la continuità del prodotto risultante.

Rinvio

Per tutto quanto non previsto nel presente capitolato, sono applicabili le disposizioni contenute nelle leggi e nei collegati per la gestione degli appalti pubblici.

A titolo informativo ed esemplificativo preferiamo ricevere software che assomigli più all’immagine della copertina che non a questo modello.