Ingegneria del Software

Corso di Laurea in Ingegneria Informatica

Lezione 26

Project Assignment

P. Foggia – N. Capuano

DIEM – Università di Salerno

Il problema da risolvere

Sviluppare un'applicazione che implementa una rubrica per gestire i contatti telefonici e/o e-mail.

Al termine delle slide c'è una descrizione informale delle funzionalità attese.

Tasks:

- Requirements Engineering
- Design
- Implementation
- Testing

Deadlines

Attività del progetto

- **Project Setup:** 24 Novembre
- **Requirements Engineering:** 30 Novembre
- **Design/Design di dettaglio:** 8 Dicembre
- **Implementazione e testing:** 15 Dicembre

Presentazione progetto: 16-19 Dicembre

Project Setup

- Definizione dei gruppi (**4 studenti**; in via eccezionale sono ammessi gruppi di 3 studenti)
- Scelta del **portavoce** di ciascun gruppo
- Creazione del repository GitHub
  - Il progetto deve essere configurato in modo da essere compilabile attraverso **Maven**
  - **IMPORTANTE: Il repository GitHub deve essere pubblicamente accessibile (in lettura)**
- Compilazione **Google sheet** con le informazioni (vedi link su e-learning)

- **Deadline: 24 novembre**

Requirements Engineering

- Definire i requisiti dell'applicazione
- Scrivere un documento in formato libero che includa:
  - **Descrizioni dei casi d'uso**
  - **Diagrammi dei casi d'uso**
  - **Ogni altra informazione necessaria a specificare e chiarire i requisiti funzionali e non funzionali (es. identificazione, classificazione, priorità ecc.)**
- Caricare il documento in una sottocartella su GitHub

Deadline: **30 Novembre**

Design

- Progettare il sistema
- Scrivere un documento in formato libero che includa :
  - **Diagrammi delle classi**
  - **Diagrammi di sequenza** per le interazioni più significative
  - **Eventuali altri diagrammi** se necessari
  - Commenti ai diagrammi, discutendo le scelte effettuate in termini di coesione, accoppiamento e principi di buona progettazione
- Caricare il documento in una sottocartella su GitHub
- Se necessario, aggiornare il documento dei requisiti su GitHub

Deadline: 8 Dicembre

Design di dettaglio

- Creare gli scheletri delle classi
- Documentare le interfacce pubbliche delle classi usando doxygen
- Mantenere aggiornato su GitHub il codice sorgente

Deadline: 8 Dicembre

Implementazione e Testing

- Implementare le classi progettate
- Realizzare Unit Test automatizzati tramite JUnit per tutte le classi (a eccezione di quelle relative all'interfaccia utente)
- Assicurarsi che tutti gli unit test siano superati con successo
- Mantenere aggiornato su GitHub il codice sorgente e il codice dei test
- Se necessario, aggiornare il documento dei requisiti e/o il documento sul design su GitHub

Deadline: 15 Dicembre

Discussione del progetto

Ogni team presenterà brevemente il progetto:

- Illustrazione e commento dei casi d'uso
- Illustrazione e commento di diagrammi delle classi
- Discussione di eventuali cambiamenti in corso d'opera (es. Modifiche ai requisiti, oppure Refactoring)
- Demo del programma funzionante

Durata: 8 minuti per team

Data: 16-19 Dicembre

Criteri di valutazione

- La valutazione terrà conto del contributo individuale e della qualità complessiva del progetto. Il contributo individuale verrà valutato anche sulla base dei commit su GitHub
  - **Individual tasks: 50%**
  - **Overall project evaluation: 50%**
- Alla fine, a ogni membro del gruppo verrà richiesto di indicare (in forma confidenziale) il contributo degli altri membri
  - **No "free riders"!**

Criteri di valutazione

- Qualità dei **Requirements**: 20%
- Qualità del **Design**: 30%
- Qualità dell' **Implementazione**: 20%
  (include appearance and usability)
- Qualità dei **Test**: 20%
- Uso degli **strumenti** (git, doxygen, maven) 10%

Requirement checklist

- Le principali funzionalità e operazioni sono adeguatamente descritte?
- I requisiti sono adeguatamente codificati e la loro priorità è chiaramente definita?
- I requisiti identificati rispettano le principali caratteristiche di verificabilità, comprensibilità, tracciabilità, coerenza e completezza?

Design checklist

- I diagrammi e i documenti descrivono in maniera chiara le scelte progettuali?
- Le classi sono progettate per fornire un set di servizi altamente correlati (elevata coesione)?
- Sono evitate dipendenze non necessarie (basso accoppiamento)?
- Sono utilizzati meccanismi di astrazione per migliorare la riutilizzabilità e/o interrompere le catene di dipendenza?
- Sono evitate duplicazioni/ripetizioni nella progettazione?
- Sono utilizzati i principi di "buona progettazione" quando appropriato?

Implementation checklist

- Le convenzioni di naming sono seguite in modo coerente? I nomi sono scelti in modo da essere facilmente comprensibili?
- La formattazione del codice (ad esempio l'indentazione) è coerente e leggibile?
- Le costanti "hard coded" sono evitate?
- Il codice è adeguatamente commentato? (i commenti dovrebbero descrivere le interfacce e il motivo per cui si stanno facendo le cose in un certo modo, non essere ridondanti)
- Vengono evitate duplicazioni/ripetizioni nel codice?
- La struttura del codice è leggibile? (ad esempio, suddividere le operazioni troppo complesse in più metodi)

Testing checklist

- I test unitari sono automatizzati?
- I test coprono la parte pubblica di ogni classe? (almeno per le classi che contengono la logica dell'applicazione)
- I casi di test coprono adeguatamente lo spazio di input (ad esempio, includendo valori limite, condizioni speciali, ecc.)?
- È stata adeguatamente testata la soluzione complessiva (test di integrazione)?

Checklist sull'uso degli strumenti

- Ciascuno sviluppatore ha sottomesso il codice che ha realizzato tramite git?
- I commit sul repository sono stati frequenti?
- È possibile compilare il progetto tramite Maven?
- È possibile lanciare i test tramite Maven?
- È possibile generare la documentazione delle interfacce delle classi tramite Doxygen?
- La documentazione Doxygen è adeguata?

Descrizione informale delle funzionalità

- Il programma gestisce un insieme di **Contatti**
- Ad ogni contatto sono associate le seguenti informazioni:
  - Un **nome** e un **cognome**; una delle due informazioni *può* essere vuota (ma non entrambe)
  - Da zero a tre **numeri di telefono**
  - Da zero a tre **indirizzi e-mail**
- Il programma deve avere un'interfaccia di tipo grafico (GUI)

Descrizione informale delle funzionalità

- Deve essere possibile creare, modificare, cancellare contatti
- Deve essere possibile salvare le informazioni dei contatti su file, e caricare le informazioni da file (il salvataggio/caricamento riguarda l'intera rubrica, non singoli contatti)
- Deve essere possibile accedere ai contatti visualizzando una lista ordinata alfabeticamente (per cognome e nome)
- Deve essere possibile cercare un contatto inserendo la sottostringa iniziale del nome o del cognome