UNIVERSITÀ
DEGLI STUDI DI MILANO-BICOCCA

SYLLABUS DEL CORSO

Evolution of Software Systems and Reverse Engineering

2526-2-F1801Q158

Obiettivi

Lo studente acquisirà competenze relative alle problematiche principali dell'evoluzione del software e della reverse engineering, sarà in grado di effettuare analisi empiriche del software e utilizzare diversi strumenti di reverse engineering, di supporto alla comprensione, evoluzione e manutenzione del software e strumenti per la gestione del debito tecnico.

Conoscenza e capacità di comprensione
Al termine del corso, lo studente avrà acquisito una solida conoscenza sulla manutenzione ed evoluzione di progetti software complessi e sulla gestione del debito tecnico.
Conoscenza e capacità di comprensione applicata
Lo studente sarà in grado di effettuare diverse analisi del software, anche analisi di correlazione fra determinati problemi del software e analisi di predizione. Ogni studente dovrà conoscere ed utilizzare strumenti /tools molto noti nelle aziende per la comprensione del codice, gestione del debito tecnico e reverse engineering.
Autonomia di giudizio
Il corso promuove lo sviluppo del pensiero critico attraverso la valutazione comparativa di strumenti per l'analisi e manutenzione del software, e attraverso il contributo individuale nel progetto d'esame. Gli studenti avranno anche la possibilità di seguire uno o due seminari di aziende su tematiche di grande interesse per l'evoluzione dei sistemi software e reverse engineering che stimoleranno ulteriormente le capacità di giudizio.
Abilità comunicative
Lo studente sarà in grado di comunicare efficacemente attraverso una o due presentazione individuali in aula, attraverso il progetto d'esame relativo all'utilizzo di strumenti visti a lezione per effettuare analisi empiriche di progetti software, attraverso l'elaborato prodotto sul progetto d'esame e la prova individuale orale.
Capacità di apprendere
Il corso promuove lo sviluppo dell'autonomia nello studio attraverso un approccio didattico attivo. Al termine del percorso, lo studente sarà in grado di approfondire in modo indipendente tematiche avanzate sull'evoluzione e manutenzione del software che possono sfociare anche in una tesi, consolidando così le basi per un apprendimento continuo anche oltre il contesto curriculare.
Le tematiche delle tesi possono riguardare ad esempio la valutazione dell'impatto del refactoring di problemi architetturali sulla sicurezza, prestazioni, consumo energetico e su altre metriche di qualità o sulla migrazione

verso architetture a microservizi, o sull’utilizzo degli LLM per il riconoscimento e refactoring di problemi architetturali nel codice.

Contenuti sintetici

Introduzione alle principali problematiche di reverse engineering, software evolution e program comprehension.
Object-oriented reverse engineering e reengineering patterns.
Analisi della qualità del software:metriche e tools.
Sperimentazioni e confronto di diversi tool di supporto alla reverse engineering e alla valutazione del debito tecnico.

Programma esteso

1 Introduzione alla Software evolution, Reverse Engineering, Sistemi Legacy. Comprensione e manutenzione del software.

2 Tecniche e tools per la Reverse Engineering.

3 Object-oriented patterns per la reverse engineering e reeengineering.

4 Metriche di Qualità del Software, software quality assessment. Application Portfolio Management: tools e tecniche .

5. Modernizzazione dei sistemi legacy: Migrazione dei sistemi legacy verso sistemi a microservizi.
6. Tool e tecniche per la software architecture reconstruction.
7. Riconoscimento di antipattern, code smell e architectural smells nel codice, il loro refactoring.
8. Impatto delle tecniche di refactoring sulla qualità del codice, sulle prestazioni e sul consumo di energia.
9. Esempi di diverse analisi empiriche: Analisi di correlazione fra diverse metriche di qualità del codice e violazioni/probelmi nel codice. Analisi della predizione di problemi del software attraverso tecniche di machine Learning e data mining.
10. Introduzione a tecniche di hacking, decompiling and code obfuscation per la protezione del codice. Analisi statica e dinamica per la reverse engineering .
11. Tecniche e strumenti per l'identificazione e gestione del Technical Debt.

Prerequisiti

Conoscenza approfondita del linguaggio Java,

Modalità didattica

Le lezioni si dovrebbero svolgere tutte in presenza.

6 lezioni da 2 ore in modalità erogativa in presenza..
15 lezioni da 2 ore in modalità erogativa ed interattiva in presenza..
4 lezioni da 2 ore in modalità erogativa ed interattiva in presenza.

Lezioni frontali ed esercitazioni, approfondimenti di alcuni contenuti e sperimentazione di alcuni strumenti in aula attraverso presentazione da parte degli studenti.

Il corso viene erogato in italiano, ma se uno studente straniero è presente, il corso verrà erogato in inglese.

Materiale didattico

Slides del docente, articoli, survey e tutorial forniti dal docente, tesi di laurea e di dottorato svolte presso il laboratorio di ricerca Essere e presso altre Università straniere con cui il laboratorio collabora.

Testi:

Ingegneria del Software, Sommerville, solo 3 capitoli.

Object Oriented Reeengineering patters, Oscar Nierstrasz -Disponibile online

La maggior parte del materiale su cui dovrete studiare sarà disponibile online.

Periodo di erogazione dell'insegnamento

I semestre

Modalità di verifica del profitto e valutazione

Uno o due task assegnati durante il corso che prevedono una presentazione in aula. Valutazione nel range di 0-3 punti.

Un progetto finale singolo o al massimo in due studenti relativo alla sperimentazione di alcuni tool di reverse engineering e/o analisi empiriche sui dati raccolti dai tool utilizzati per il progetto. Valutazione 0-22 punti

Discussione orale sul progetto finale. Valutazione 0-8 punti.

Orario di ricevimento

Per appuntamento.

Sustainable Development Goals