SCHEDA DELL'INSEGNAMENTO (SI)

"SOFTWARE TESTING"

SSD ING-INF/05

DENOMINAZIONE DEL CORSO DI STUDIO: LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

ANNO ACCADEMICO 2023-2024

INFORMAZIONI GENERALI - DOCENTE

DOCENTE: PORFIRIO TRAMONTANA
TELEFONO: 0817683901
EMAIL: PORFIRIO.TRAMONTANA@UNINA.IT

INFORMAZIONI GENERALI - ATTIVITÀ

ANNO DI CORSO (I, II, III): II
SEMESTRE (I, II): I
CFU: 6

INSEGNAMENTI PROPEDEUTICI (se previsti dall'Ordinamento del CdS)

Nessuno

EVENTUALI PREREQUISITI

I prerequisiti fondamentali di questo corso sono la conoscenza di base della programmazione ad oggetti, ed in particolare del linguaggio Java. Inoltre, è necessaria una conoscenza di base delle problematiche relative all'Ingegneria del Software. Nessuna conoscenza preliminare relativa al testing è invece necessaria.

OBIETTIVI FORMATIVI

Il corso si propone di definire e approfondire tematiche di Verifica e Validazione del Software, includenti metodologie, strategie, tecniche, strumenti e processi di Software Testing e Debugging. Il corso si propone di contribuire all'acquisizione di competenze relative allo sviluppo di software di qualità e alla sua valutazione, con particolare attenzione all'automazione di tali attività.

Questi obiettivi sono di fondamentale importanza nella realizzazione e valutazione di qualsiasi impianto informatico e sistema di elaborazione, da quelli hardware a quelli software, dai sistemi operativi alle reti di elaboratori, dalle basi di dati ai sistemi informativi, dall'intelligenza artificiale alla robotica.

RISULTATI DI APPRENDIMENTO ATTESI (DESCRITTORI DI DUBLINO)

Conoscenza e capacità di comprensione

Lo studente deve essere in grado di comprendere l'importanza della qualità del software e della sua valutazione, in particolare la ricerca e correzione dei suoi difetti tramite l'applicazione di adeguate strategie e tecniche di testing. Lo studente deve comprendere quali siano le difficoltà intrinseche nel testing di un sistema software e saper scegliere tra le soluzioni che possono essere applicate. Tali conoscenze permetteranno agli studenti di poter integrare le conoscenze relative all'ingegneria del software allo scopo di organizzare processi di sviluppo di software di qualità e ad alta affidabilità. Lo studente deve anche essere in grado di comprendere come le metodologie e tecniche apprese possano contribuire a risolvere problemi relativi alla realizzazione di sistemi hardware o software di qualità.

Capacità di applicare conoscenza e comprensione

Lo studente deve dimostrare di saper organizzare processi di sviluppo software che prevedano una continua valutazione e miglioramento della qualità grazie all'esecuzione di efficaci ed efficienti attività di testing. Lo studente deve essere in grado di progettare e sviluppare piani di test del software a diversi livelli di dettaglio, dal testing di unità fino al testing di sistema e a quello di accettazione. Lo studente deve essere in grado di realizzare casi di test con elevato livello di automazione, sia in fase di generazione dei casi di test, che in fase di esecuzione e valutazione dell'esito. Lo studente deve essere in grado di utilizzare al meglio gli strumenti di testing esistenti e di aver acquisito la capacità di progettare e realizzare strumenti di testing innovativi.

PROGRAMMA-SYLLABUS

Elementi teorici del testing software: Definizioni – Problemi indecidibili – Tassonomia delle attività di testing

Qualità del testing: Adeguatezza - Precisione - Ripetibilità - Capacità di trovare i difetti - Efficacia – Efficienza

Specifica dei casi di test: Input – Output – Oracoli – Pre e post-condizioni

JUnit: Introduzione a JUnit - Implementazione di test di unità con JUnit su programmi Java – Assunzioni e asserzioni – Testing delle eccezioni – Test dinamici e parametrici – Testing Data Driven con JUnit

Testing Black Box. Testing basato sui requisiti e sugli scenari dei casi d'uso - Test con classi di equivalenza e valori limite - Strumenti e tecniche per la generazione combinatoriale dei casi di test - Testing con tabelle di decisione.

Testing White Box – Modelli e metriche di copertura - Strumenti per la misura automatica della copertura del codice.

Testing di integrazione e testing in isolamento – Tecniche di testing in isolamento con driver e stub – Grafi delle dipendenze - Strategie per il testing di integrazione: top-down, bottom-up, sandwich – Testing con Mock Objects – Cenni di dependency injection – Framework per la creazione di Mock Objects

Testing dell'interfaccia utente – Tecniche di testing delle interfacce utente a caratteri - Testing delle GUI - Modellazione delle GUI con macchine a stati finiti - Problema dell'esplosione degli stati e tecnica degli stati equivalenti – Librerie a supporto del testing di GUI - Validazione degli input.

Tecniche User Session: Tassonomia delle tecniche Capture & Replay per la generazione di test sull'interfaccia utente - Problematiche relative alla generazione di locatori robusti – Problematiche legate alla replicabilità dei test catturati – Strumenti di Capture & Replay per applicazioni Web

Tecniche di testing automation: Automazione nella generazione/esecuzione/valutazione dell'esito dei casi di test - Generazione e valutazione automatica di oracoli - Crash testing - Smoke Testing - Regression Testing.

Random testing: Caratteristiche e parametri del random testing - Problema della terminazione del random testing - Tecniche e strumenti per l'esplorazione automatica della GUI - Riduzione e prioritizzazione delle test suite.

Mutation Testing: Test case mutation – Mutation Analysis – Mutation based Testing – Strumenti per la generazione di mutant - Search based software testing: utilizzo di EvoSuite.

Experience based Testing: Exploratory Testing – Error Guessing e Checklist based Testing – Beta Testing - Crowdtesting – Piattaforme di CrowdTesting - Software Testing Gamification – Testing di unità con Code Defenders

Testing in Continuous Integration: Cenni su tecniche, linguaggi e strumenti per la build automation - Cenni su tecniche e strumenti per la gestione delle versioni concorrenti – Automazione di attività di testing in Github con Github Actions

Analisi statica del codice sorgente: Tecniche per l'analisi statica - Strumenti automatici di analisi statica: Checkstyle, PMD, Findbugs, Android Lint - Cenno all'utilizzo di SonarQube per il monitoraggio continuo della qualità del software

Debugging: Localizzazione e correzione dei difetti - Tecniche per il debugging: forza bruta - backtracking, eliminazione delle cause - Strumenti per il debugging: breakpoint, breakpoint condizionali, watch, watchpoint.

Testing delle applicazioni Android. Introduzione al sistema Android e alla programmazione di app Android - Testing di unità: utilizzo di JUnit e Robolectric - Testing della GUI: utilizzo di Robotium e Android Espresso - Utilizzo di Espresso Recorder per il Capture & Replay di casi di test - Testing di sistema: utilizzo di UIAutomator - Strumenti di testing a basso livello: Monkey - Strumenti di monitoraggio - Testing dei memory leaks - Strumenti di testing sistematico: Android Ripper - Testing di applicazioni con risorse remote: Cloud testing con Firebase TestLab, Alpha Testing, Beta Testing.

MATERIALE DIDATTICO

Il materiale didattico comprende un insieme completo di presentazioni realizzate dal docente che accompagnano tutte le lezioni del corso. In queste presentazioni sono riportati riferimenti a numerosi libri, articoli scientifici, strumenti di sviluppo, materiale sperimentale, esempi ed altre risorse online, che sono tutte messe a disposizione degli studenti.

Tra i libri consigliati:

- Ian Sommerville, Ingegneria del Software, Pearson Addison Wesley
- Mauro Pezzè and Michal Young, Software Testing and Analysis, John Wiley & Sons, 2008

MODALITÀ DI SVOLGIMENTO DELL'INSEGNAMENTO

Il corso si compone di 48 ore di lezione complessive, nelle quali si alterna didattica frontale (36 ore) con presentazione e discussione degli argomenti del corso ed esercitazioni (12 ore), nelle quali vengono mostrati e provati interattivamente tecniche e strumenti software a supporto delle attività di testing. Per alcuni argomenti sono proposte delle sperimentazioni basate sulla realizzazione di casi di test per sistemi software con l'applicazione di tecniche innovative di testing.
Vengono fornite le registrazioni di tutte le lezioni (prese in diretta mentre si tengono), il codice di tutti gli esempi ed esercitazioni svolte. Per le attività pratiche sono utilizzati esclusivamente software aperti e di libero utilizzo e vengono fornite le indicazioni fondamentali sul loro reperimento e utilizzo.

VERIFICA DI APPRENDIMENTO E CRITERI DI VALUTAZIONE

a) Modalità di esame:

L'esame si articola in prova	
scritta e orale	
solo scritta	
solo orale	X
discussione di elaborato progettuale	X
Altro	

L'esame si articola in uno o due elaborati progettuali, seguiti da una prova orale. Uno dei due elaborati progettuali consiste nella progettazione e realizzazione di un piano di test su di un software specifico, eventualmente proposto dal docente, applicando le tecniche e strumenti presentati al corso. Il secondo elaborato, invece, consiste nell'approfondimento di una tematica di testing innovativa rispetto ai contenuti del corso o nella realizzazione di uno strumento innovativo di testing del software.
La consegna e discussione con esito positivo degli elaborati di natura progettuale consente di accedere alla prova orale. La prova orale consiste di una valutazione complessiva del raggiungimento degli obiettivi di apprendimento previsti, tramite interrogazione su argomenti relativi a tutto il programma del corso.
Il voto finale terrà in considerazione sia della correttezza e innovatività degli elaborati progettuali, che della valutazione della prova orale.