PROGRAMMA DEL CORSO DI INGEGNERIA DEL SOFTWARE

SETTORE SCIENTIFICO

ING-INF/05 (IINF-05/A)

CFU

12

OBIETTIVI FORMATIVI PER IL RAGGIUNGIMENTO DEI RISULTATI DI APPRENDIMENTO PREVISTI NELLA SCHEDA SUA

/**/
Il corso di Ingegneria del Software mira a fornire agli studenti una comprensione approfondita dei processi, metodi e strumenti per lo sviluppo di sistemi software complessi.

Gli obiettivi principali includono:

Acquisire una visione globale dei processi e dei modelli di sviluppo software: Comprendere i diversi cicli di vita del software, inclusi i modelli tradizionali e agili, e saperli applicare in contesti pratici per gestire progetti software in modo efficiente. Utilizzare strumenti di modellazione per la progettazione e la documentazione del software: Applicare il linguaggio UML per rappresentare e comunicare in modo formale le strutture e i comportamenti del sistema, facilitando la progettazione e la collaborazione nel team di sviluppo. Gestire l'analisi e la specifica dei requisiti: Raccogliere, analizzare e formalizzare i requisiti funzionali e non funzionali, assicurando che il prodotto software soddisfi le esigenze degli utenti e degli stakeholder. Progettare architetture software efficienti e scalabili: Sviluppare soluzioni architetturali di alto livello, con particolare attenzione alla decomposizione del sistema, gestione dei dati e sicurezza, per garantire affidabilità e manutenibilità del software. Assicurare la qualità del software attraverso tecniche di testing e verifica: Imparare a pianificare ed eseguire test efficaci a diversi livelli, utilizzando strumenti automatizzati per garantire la correttezza, l'integrazione e le prestazioni del sistema software.

RISULTATI DI APPRENDIMENTO ATTESI

/**/
- Conoscenza e capacità di comprensione

Capacità di comprendere i principi fondamentali dell'ingegneria del software, inclusi i diversi modelli di sviluppo e cicli di vita, per gestire progetti software complessi in modo efficace (Obiettivo 1).
Comprensione approfondita dei principali strumenti e tecniche di modellazione, in particolare l'uso del linguaggio UML, per rappresentare strutture e comportamenti del sistema in maniera formale e accurata (Obiettivo 2).
Conoscenza delle metodologie per l'analisi e la specifica dei requisiti, con particolare attenzione alla raccolta e formalizzazione dei requisiti funzionali e non funzionali (Obiettivo 3).
Capacità di progettare architetture software efficienti e scalabili, applicando principi di decomposizione del sistema e gestione della sicurezza e dei dati persistenti (Obiettivo 4).

Comprensione delle tecniche di testing e verifica, inclusi test di unità, integrazione e accettazione, per garantire la qualità e la robustezza del software (Obiettivo 5).

- Capacità di applicare conoscenza e comprensione

Progettare e implementare soluzioni software basate su diversi modelli di sviluppo, selezionando il ciclo di vita più appropriato in base alle esigenze del progetto (Obiettivo 1).
Utilizzare UML per modellare vari aspetti di un sistema software, inclusi diagrammi delle classi, di sequenza e di stato, per facilitare la progettazione e la documentazione del software (Obiettivo 2).
Applicare tecniche di raccolta e specifica dei requisiti per tradurre le esigenze degli stakeholder in specifiche funzionali e non funzionali dettagliate (Obiettivo 3).
Progettare architetture software scalabili e modulari, integrando soluzioni per la gestione della sicurezza, dei dati persistenti e delle condizioni limite (Obiettivo 4).
Sviluppare piani di test e utilizzare strumenti automatizzati per eseguire test unitari, di integrazione e di sistema, garantendo l'affidabilità e la qualità del software (Obiettivo 5).

- Autonomia di giudizio

Valutare in modo critico l'adeguatezza di diversi modelli di sviluppo software rispetto alle caratteristiche del progetto, come dimensione e complessità (Obiettivo 1).
Autonomia nel giudicare l'efficacia delle soluzioni di modellazione adottate tramite UML per garantire la coerenza e la manutenibilità del sistema (Obiettivo 2).
Capacità di valutare la completezza e la correttezza dei requisiti raccolti, identificando eventuali ambiguità o conflitti (Obiettivo 3).
Capacità di giudicare la solidità e la scalabilità delle architetture software progettate, analizzando la loro manutenibilità e sicurezza (Obiettivo 4).
Autonomia nel valutare l'efficacia dei processi di testing e nel proporre miglioramenti per ottimizzare la qualità del software (Obiettivo 5).

- Abilità comunicative

Capacità di spiegare in modo chiaro e coerente le diverse fasi del ciclo di vita del software e le loro implicazioni tecniche a un pubblico sia tecnico che non tecnico (Obiettivo 1).
Abilità nel presentare e discutere modelli UML complessi, comunicando in modo efficace le scelte di progettazione e la logica del sistema (Obiettivo 2).
Capacità di comunicare i requisiti del software e le relative priorità agli stakeholder, garantendo una comprensione condivisa degli obiettivi del progetto (Obiettivo 3).
Capacità di spiegare architetture software avanzate e le loro implicazioni tecniche in termini di prestazioni, sicurezza e scalabilità (Obiettivo 4).

- Capacità di apprendimento

Capacità di aggiornarsi costantemente sui nuovi modelli di sviluppo e sulle metodologie emergenti nell'ingegneria del software, come DevOps e CI/CD (Obiettivo 1).
Sviluppare competenze di autoapprendimento per approfondire nuove tecniche di modellazione e design pattern, mantenendo la capacità di applicare le migliori pratiche nella progettazione del software (Obiettivo 2).
Capacità di apprendere in autonomia strumenti e tecniche per l'analisi e la gestione dei requisiti, con particolare attenzione ai requisiti non funzionali (Obiettivo 3).
Impegno continuo nell'acquisire nuove competenze di testing e verifica del software, comprendendo l'evoluzione degli strumenti e delle tecniche di automazione (Obiettivo 5).

PREREQUISITI

/**/

Nessuno.

PROGRAMMA DIDATTICO: ELENCO VIDEOLEZIONI/MODULI

/**/ Introduzione ai processi software Sviluppo Agile Introduzione al ciclo di vita del software Sviluppo dei processi del ciclo di vita Modelli di ciclo di vita del software Introduzione alla modellazione Diagrammi delle classi e degli oggetti Diagrammi di sequenza Diagrammi di package e di deployment Casi d'uso Diagrammi di macchina a stati Diagrammi di attività Comunicazione, strutture e componenti Collaborazione, interazione generale e temporizzazione Elicitazione dei requisiti: Introduzione e concetti chiave Elicitazione dei requisiti: Attività principali Gestione del processo di specifica dei requisiti Analisi dei requisiti e modellazione a oggetti Attività principali dell'analisi dei requisiti Gestione del processo di analisi dei requisiti Progettazione di interfacce grafiche Introduzione al system design System Design: concetti principali System Design: Dagli oggetti ai sottosistemi System Design: Obiettivi di progettazione Gestione del processo di system design Introduzione all'object design Specifica delle interfacce: concetti generali Attività della specifica delle interfacce Gestione del processo di object design Mapping tra modelli e codice Introduzione al testing Attività di testing Gestione del processo di testing Tecniche per la selezione dei casi di test Testing white box Test-Driven-Development Introduzione a Software Project Management Risk Management Quality Management Introduzione a Rationale Management Rationale Management: dai problemi alle decisioni Configuration Management People management Strumenti di collaborazione Strutture di collaborazione e cenni su community smells Didattica Innovativa: intervista a professore ordinario di Ingegneria del Software Didattica Innovativa: intervista a Senior Product Engineer Storia, principi e sintassi del linguaggio Java Tipi primitivi, selezione e iterazione JDK e JRE: compilare ed eseguire programmi in Java Classi, oggetti, ereditarietà Polimorfismo, interfacce e classi astratte Accesso ai dati con JDBC e MySQL Introduzione ai design pattern Caso studio: design pattern Composite e Strategy Caso studio: design pattern Decorator Caso studio: design pattern Abstract Factory, Singleton e Bridge

AGENDA

/**/
Le attività di Didattica Interattiva (TEL-DI) consistono, per ciascun CFU, in 2 ore erogate in modalità sincrona su piattaforma Class, svolte dal docente anche con il supporto del tutor disciplinare, e dedicate a una o più tra le seguenti tipologie di attività:

sessioni live, in cui il docente guida attività applicative, stimolando la riflessione critica e il confronto diretto con gli studenti tramite domande in tempo reale e discussioni collaborative; webinar interattivi, arricchiti da sondaggi e domande dal vivo, per favorire il coinvolgimento attivo e la costruzione della conoscenza; lavori di gruppo e discussioni in tempo reale, organizzati attraverso strumenti collaborativi come le breakout rooms, per sviluppare strategie di problem solving e il lavoro in team; laboratori virtuali collettivi, in cui il docente guida esperimenti, attività pratiche o l'analisi di casi di studio, rendendo l'apprendimento un'esperienza concreta e partecipativa; Tali attività potranno essere eventualmente supportate da strumenti asincroni di interazione come per esempio: forum; wiki; quiz; glossario.

Si prevede l'organizzazione di almeno due edizioni di didattica interattiva sincrona nel corso dell'anno accademico. Si precisa che il ricevimento degli studenti, anche per le tesi di laurea, non rientra nel computo della didattica interattiva.

ATTIVITÀ DIDATTICA EROGATIVA (DE)

/**/

Le attività di Didattica Erogativa consistono, per ciascun CFU, nell'erogazione di 5 videolezioni della durata di circa 30 minuti. A ciascuna lezione sono associati:

una dispensa (PDF) di supporto alla videolezione oppure l'indicazione di capitoli o paragrafi di un ebook di riferimento, scelto dal docente tra quelli liberamente consultabili in piattaforma da studentesse e studenti; un questionario a risposta multipla per l'autoverifica dell'apprendimento.

TESTO CONSIGLIATO

/**/
Ingegneria del Software, 10 ed. Ian Sommerville, Pearson

UML DIstilled, 4 ed. Martin Fowler, Pearson
Applicare UML e Pattern - analisi e programmazione orientata a oggetti 5 ed. Craig Larman, Pearson.
Introduzione all'ingegneria del software moderna. Ian Sommerville, Daniela Micucci, Pearson
Cay Horstmann, "Concetti di Informatica e Fondamenti di Java", Settima Edizione, Apogeo Education, Maggioli Editore, 2020.

Si specifica che i testi consigliati sono solo per approfondimento volontario, e che essi non saranno oggetto specifico di esame, essendo il modello didattico basato sull'utilizzo delle dispense del docente, soprattutto per la verifica in sede di esame.

MODALITÀ DI VERIFICA DELL'APPRENDIMENTO

/**/
L'esame può essere sostenuto sia in forma scritta che in forma orale. L'esame orale consiste in un colloquio con la Commissione sui contenuti dell'insegnamento. L'esame in forma scritta consiste nello svolgimento di un test composto da 31 domande. Per ogni domanda lo studente deve scegliere una delle 4 possibili risposte. Solo una risposta è corretta e, in caso di risposte errate o mancanti, non sarà attribuita alcuna penalità. Rispondendo correttamente a tutte le 31 domande, si conseguirà la lode.

Oltre alla prova d'esame finale, il percorso prevede attività di didattica interattiva sincrona e prove intermedie che consentono alle studentesse e agli studenti di monitorare il proprio apprendimento, attraverso momenti di verifica progressiva e consolidamento delle conoscenze.

La partecipazione alle attività di didattica interattiva sincrona consente di maturare una premialità fino a 2 punti sul voto finale, attribuiti in funzione della qualità della partecipazione alle attività e dell'esito delle prove.

Per accedere alle prove intermedie è necessario aver seguito almeno il 50% di ogni ora di didattica interattiva. Le prove intermedie possono consistere in un test di fine lezione o nella predisposizione di un elaborato. Le prove intermedie si considerano superate avendo risposto correttamente ad almeno l'80% delle domande di fine lezione.

In caso di prove intermedie che prevedano la redazione di un elaborato, il superamento delle stesse ai fini della premialità sarà giudicata dal docente titolare dell'insegnamento. I punti di premialità, previsti per le prove intermedie, sono sommati al voto finale d'esame solo se la prova d'esame è superata con un punteggio pari ad almeno 18/30 e possono contribuire al conseguimento della lode.

Le modalità d'esame descritte sono progettate per valutare il grado di comprensione delle nozioni teoriche e la capacità di applicazione delle stesse e consentiranno di valutare il livello di competenza e l'autonomia di giudizio

maturati dalla studentessa e dallo studente. Le abilità di comunicazione e la capacità di apprendimento saranno valutate anche attraverso le interazioni dirette che avranno luogo durante la fruizione dell'insegnamento.

RECAPITI

/**/
roberto.vergallo@unipegaso.it

fabiano.pecorelli@unipegaso.it

massimiliano.pirani@unipegaso.it

OBBLIGO DI FREQUENZA

/**/
A studentesse e studenti viene richiesto di partecipare ad almeno il 70% dell'attività di didattica erogativa (70% della TEL-DE).

AGENDA

/**/
Nella sezione Informazioni Appelli, nella home del corso, per ogni anno accademico vengono fornite le date degli appelli d'esame.

Le attività di didattica interattiva sincrona sono calendarizzate in piattaforma nella sezione Class.

Le attività di ricevimento di studenti e studentesse sono calendarizzate nella sezione Ricevimento Online.

CALENDARIO

/**/
Calendario lezioni sincrone 2a edizione 2026:

Massimiliano Pirani UML: Diagrammi strutturali 04/02/2026 Ore 17.00 UML: Diagrammi comportamentali 04/02/2025 Ore 18.00