Syllabus

Descrizione corso

Titolo insegnamento	Software Systems Engineering
Codice insegnamento	76265
Titolo aggiuntivo	
Settore Scientifico-Disciplinare	
Lingua	Inglese; Italiano
Corso di Studio	Corso di laurea in Informatica
Altri Corsi di Studio (mutuati)	
Docenti	prof. Barbara Russo, Barbara.Russo@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2242 prof. Eduardo Martins Guerra, Eduardo.MartinsGuerra@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/43879
Assistente	
Semestre	Secondo semestre
Anno/i di corso	2
CFU	12
Ore didattica frontale	70
Ore di laboratorio	50
Ore di studio individuale	180
Ore di ricevimento previste	
Sintesi contenuti	- Principi dell'architettura del software e dei sistemi - Processo di progettazione dell'architettura software - Componenti architettonici e framework - Approcci per la suddivisione architettonica - Modelli e stili architettonici - Integrazione di componenti di intelligenza artificiale nei progetti

	architettonici - Verifica e validazione - Tecniche di test black-box e white-box - Tecniche e strumenti per l'automazione dei test - Test di integrazione e regressione - Test web - IA per la generazione di casi di test
Argomenti dell'insegnamento	L'obiettivo del modulo Software Systems Architecture è fornire agli studenti una solida comprensione del ruolo dell'architettura software all'interno del ciclo di vita dello sviluppo del software. Gli studenti impareranno a progettare architetture utilizzando modelli consolidati e buone pratiche, acquisiranno una panoramica dei vari stili architetturali e delle tendenze emergenti, valuteranno e bilanceranno i compromessi legati agli attributi di qualità e applicheranno diversi approcci architetturali allo sviluppo di software di alta qualità. Il modulo Tools and Techniques for Software Testing è progettato per fornire agli studenti la capacità di selezionare, utilizzare, personalizzare e distribuire strumenti e tecniche di test del software. Consente inoltre di configurare e integrare tali strumenti per supportare le attività di testing durante l'intero processo di sviluppo collaborativo del software.
Parole chiave	Software testing, prompt engineering per testing, agentic systems per testing
Prerequisiti	Basic courses in Programming and Software Engineering. Familiarity with UML and the basics of object-oriented programming.
Insegnamenti propedeutici	
Modalità di insegnamento	Il corso include lezioni frontali e esercizi di laboratorio
Obbligo di frequenza	Attendance is not compulsory. Non-attending students have to contact the lecturer at the start of the course to agree on the modalities of the independent study.
Obiettivi formativi specifici e risultati di apprendimento attesi	Conoscenza e comprensione - D1.8 Conoscere a fondo i principali fondamenti delle tecniche e dei metodi di progettazione, sviluppo e manutenzione del software.

	Applicare conoscenza e comprensione - D2.5 Essere in grado di applicare le proprie conoscenze alla progettazione, allo sviluppo e al collaudo di sistemi informativi che soddisfino determinati requisiti. - D2.7 Essere in grado di condurre semplici esperimenti sui sistemi informativi raccogliendo misure sul comportamento del sistema. - D2.10 Essere in grado di risolvere problemi tipici dell'informatica basati sulle metodologie dell'ingegneria del software, come la definizione dei requisiti, i possibili metodi per la soluzione, la selezione dei metodi e degli strumenti più appropriati e la loro applicazione. - D2.11 Essere in grado di valutare la qualità dei sistemi informativi e di identificare gli aspetti critici. Capacità di formulare giudizi - D3.1 Essere in grado di raccogliere e interpretare dati utili e di giudicare i sistemi informativi e la loro applicabilità. - D3.2 Essere in grado di lavorare autonomamente in base al proprio livello di conoscenza e comprensione. - D3.3 Essere in grado di assumersi la responsabilità dello sviluppo di progetti o della consulenza informatica. Competenze comunicative - D4.1 Essere in grado di utilizzare una delle tre lingue, inglese, italiano e tedesco, e di utilizzare in modo appropriato termini tecnici e di comunicazione. - D4.3 Essere in grado di negoziare con un cliente per la definizione dei pre-requisiti e delle caratteristiche dei sistemi informativi. - D4.4 Essere in grado di strutturare e scrivere documentazione tecnica. Capacità di apprendimento - D5.1 Aver sviluppato capacità di apprendimento per proseguire gli studi con un alto grado di autonomia. - D5.2 Aver acquisito capacità di apprendimento che consentano di svolgere attività progettuali in aziende, istituzioni pubbliche o in comunità di sviluppo distribuite. - D5.3 Essere in grado di seguire la rapida evoluzione tecnologica e

	di apprendere tecnologie informatiche all'avanguardia e aspetti innovativi dei sistemi informativi di ultima generazione.
Obiettivi formativi specifici e risultati di apprendimento attesi (ulteriori info.)	
Modalità di esame	Un esame scritto finale unificato sarà svolto con domande di verifica provenienti da entrambi i moduli. Ogni modulo prevede inoltre le seguenti attività durante il corso: Software Systems Architecture: la valutazione di laboratorio consiste in esercitazioni che devono essere svolte e consegnate ogni settimana. Attività opzionali possono valere punti extra nel voto finale del modulo. Tools and Techniques for Software Testing: lavoro di progetto con esercitazioni di gruppo.
Criteri di valutazione	L'esame scritto valuta la capacità dello studente di padroneggiare la terminologia del corso, analizzare strumenti e tecniche in relazione al loro specifico ambito di utilizzo e ai dettagli tecnici, risolvere esercizi e riassumere in modo chiaro i concetti teorici. La valutazione dei progetti e del lavoro di laboratorio si concentra sul completamento puntuale delle attività, sulla consegna di soluzioni funzionanti e sullo sviluppo o la personalizzazione di software di alta qualità che soddisfi i requisiti specificati. Il voto finale è calcolato come media dei voti ottenuti in ciascun modulo. Per Software Systems Architecture, il voto si basa su due componenti: 50% proveniente dal lavoro di laboratorio e 50% dalle domande specifiche del modulo presenti nell'esame scritto finale. Le attività opzionali previste nel modulo possono fornire crediti aggiuntivi. Per Tools and Techniques for Software Testing, il voto è composto per l'80% dal lavoro di progetto e per il 20% dalle relative domande nell'esame scritto. Il completamento e la valutazione positiva del progetto sono prerequisiti per l'accesso all'esame scritto. Per gli studenti non frequentanti, i docenti dei moduli possono offrire due opzioni: completare le attività di laboratorio o i progetti alle stesse condizioni degli studenti frequentanti, oppure sostenere

	una valutazione aggiuntiva basata su domande relative ai contenuti di laboratorio o di progetto. Gli studenti devono superare entrambi i moduli per superare l'intero corso. Un voto positivo in un modulo rimane valido per tutte e tre le sessioni d'esame ordinarie all'interno dello stesso anno accademico
Bibliografia obbligatoria	• Robert Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson, London, England, 1st edition, September 2017. ISBN 978-0-13-449416-6. • Mark Richards. Software Architecture Patterns. O'Reilly Media, Inc., 2015. ISBN 978-1-4919-2540-9. • Paul C. Jorgensen, Software Testing: A Craftsman's Approach, Fourth Edition, CRC Press, 2013 ISBN 1466560681 • Mauricio Anniche Effective Software Testing: A developer's guide ¿ 2022 Manning Publications ISBN 9781633439931 https://unibz.primo.exlibrisgroup.com/discovery/fulldisplay?docid=cdi_proq
Bibliografia facoltativa	Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice. Addison-Wesley Professional, Harlow, 3rd edition, September 2012. ISBN 978-0-321-81573-6.
Altre informazioni	- Java (https://openjdk.org) - A Java IDE (e.g., https://eclipseide.org) - Git (https://git-scm.com) - Maven (https://maven.apache.org) - Issue trackers - JaCoCo (https://www.eclemma.org/jacoco/) - FindBugs (https://findbugs.sourceforge.net) - JUnit 5 (https://junit.org) - FitNesse (https://fitnesse.org)
Obiettivi di Sviluppo Sostenibile (SDGs)	Istruzione di qualità

Modulo del corso

Titolo della parte costituente del corso	Software Systems Architecture
Codice insegnamento	76265A

Settore Scientifico-Disciplinare	IINF-05/A
Lingua	Inglese
Docenti	prof. Eduardo Martins Guerra, Eduardo.MartinsGuerra@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/43879
Assistente	
Semestre	Secondo semestre
CFU	6
Docente responsabile	
Ore didattica frontale	40
Ore di laboratorio	20
Ore di studio individuale	90
Ore di ricevimento previste	
Sintesi contenuti	- Software and systems architecture principles - Software architecture design process - Architectural components and frameworks - Approaches for architectural partitioning - Architectural patterns and styles - Integrating AI Components into Architectural Designs
Argomenti dell'insegnamento	my SNS HomeSyllabiSyllabi OverviewCourse 76265Module 76265A Module (course constituents)
Modalità di insegnamento	Il corso comprende lezioni frontali ed esercitazioni di laboratorio.
Bibliografia obbligatoria	Robert Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson, London, England, 1st edition, September 2017. ISBN 978-0-13-449416-6. Mark Richards. Software Architecture Patterns. O'Reilly Media, Inc., 2015. ISBN 978-1-4919-2540-9.

Bibliografia facoltativa	Kyle Brown, Bobby Woolf, and Joseph Yoder. Cloud Application Architecture Patterns: Designing, Building, and Modernizing for the Cloud, 2025, Oreilly & Associates Inc, ISBN 978-1098116903

Modulo del corso

Titolo della parte costituente del corso	Tools and Techniques for Software Testing
Codice insegnamento	76265B
Settore Scientifico-Disciplinare	INFO-01/A
Lingua	Italiano
Docenti	prof. Barbara Russo, Barbara.Russo@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2242
Assistente	
Semestre	Secondo semestre
CFU	6
Docente responsabile	
Ore didattica frontale	30
Ore di laboratorio	30
Ore di studio individuale	90
Ore di ricevimento previste	
Sintesi contenuti	- Verification and validation - Techniques for black-box and white-box texting - Techniques and tools for test automation - Integration and regression testing - Web testing - AI for test case generation
Argomenti dell'insegnamento	Il corso Tools and Techniques for Software Testing è progettato per fornire agli studenti la capacità di selezionare, utilizzare, personalizzare e distribuire strumenti e tecniche di testing

	software. Consente inoltre di configurare e integrare tali strumenti per supportare le attività di verifica durante l'intero processo collaborativo di sviluppo software e nei sistemi agentici
Modalità di insegnamento	Il corso si svolge con lezioni frontali e esercizi di laboratorio
Bibliografia obbligatoria	· Paul C. Jorgensen, Software Testing: A Craftsman's Approach, Fourth Edition, CRC Press, 2013 ISBN 1466560681 · Mauricio Anniche Effective Software Testing: A developer's guide ¿ 2022 Manning Publications ISBN 9781633439931 https://unibz.primo.exlibrisgroup.com/discovery/fulldisplay?docid=cdi_proquest_e
Bibliografia facoltativa	