FACOLTÀ	Scienze MM.FF.NN.
ANNO ACCADEMICO	2014/2015
CORSO DI LAUREA	Informatica
INSEGNAMENTO	Ingegneria del Software
TIPO DI ATTIVITÀ	Caratterizzante
AMBITO DISCIPLINARE	Discipline Informatiche
CODICE INSEGNAMENTO	03968
ARTICOLAZIONE IN MODULI	No
SETTORI SCIENTIFICO DISCIPLINARI	INF/01
DOCENTE RESPONSABILE	DA ASSEGNARE
CFU	6
NUMERO DI ORE RISERVATE ALLO STUDIO PERSONALE	102
NUMERO DI ORE RISERVATE ALLE ATTIVITÀ DIDATTICHE ASSISTITE	48
PROPEDEUTICITÀ	Tutte le materie del I anno, Algoritmi e Strutture Dati, Informatica Teorica, Basi di Dati
ANNO DI CORSO	III
SEDE DI SVOLGIMENTO DELLE LEZIONI	Consultare il sito www.cs.unipa.it
ORGANIZZAZIONE DELLA DIDATTICA	Lezioni frontali, esercitazioni in aula, attività in laboratorio
MODALITÀ DI FREQUENZA	Facoltativa
METODI DI VALUTAZIONE	Prova orale con discussione di un progetto
TIPO DI VALUTAZIONE	Voto in trentesimi
PERIODO DELLE LEZIONI	II Semestre
CALENDARIO DELLE ATTIVITÀ DIDATTICHE	Come da calendario disponibile presso www.cs.unipa.it
ORARIO DI RICEVIMENTO DEGLI STUDENTI	Disponibile dopo l'assegnazione del corso al docente.

RISULTATI DI APPRENDIMENTO ATTESI

Conoscenza e capacità di comprensione

Acquisizione dei concetti e delle tecniche fondamentali per la raccolta dei requisiti, stesura delle specifiche, pianificazione, progettazione, implementazione, integrazione e manutenzione di un prodotto software. Conoscenza del linguaggio UML (Unified Modeling Language).
Conoscenza degli aspetti fondamentali del processo di sviluppo del software (con particolare riferimento al software object-oriented)

Capacità di applicare conoscenza e comprensione

Capacità di raccogliere e formalizzare i requisiti del sistema. Capacità di progettare un sistema software anche complesso. Capacità di stimare costi e tempi. Capacità di dimensionare correttamente i vari componenti. Capacità di utilizzo dell'UML come linguaggio di modellazione. Utilizzo degli strumenti di ausilio alla progettazione più comuni.

Autonomia di giudizio

Saper identificare i requisiti. Saper formalizzare i requisiti. Saper individuare, a diversi livelli di astrazione, le soluzioni necessarie al raggiungimento degli obiettivi funzionali del sistema. Saper individuare le architetture più idonee per soddisfare i requisiti funzionali e non. Saper modellare un

sistema informatico dai requisiti fino alla implementazione, test e configurazione/installazione. Saper attuare scelte risk-driven nella pianificazione e attuazione del progetto.

Abilità comunicative
Proprietà di espressione nella presentazione delle scelte progettuali e delle motivazioni che hanno portato ad esse.

Capacità d'apprendimento
Sapere affrontare la decomposizione di un problema, identificare i requisiti e le soluzioni tecnologiche necessarie. Sapere contestualizzare le abilità acquisite in problemi concreti nell'ambito lavorativo.

OBIETTIVI FORMATIVI DEL CORSO
Il modulo si propone di fornire allo studente le conoscenze e competenze necessarie per affrontare la progettazione di un sistema informatico. L'obiettivo principale del corso è lo studio del processo di sviluppo del software (con particolare riferimento ai software object-oriented). Verranno esaminate le tecniche di raccolta dei requisiti, stesura delle specifiche, pianificazione, progettazione, implementazione, testing, integrazione e manutenzione.

	INGEGNERIA DEL SOFTWARE
ORE FRONTALI	**LEZIONI FRONTALI**
2	Introduzione all'Ingegneria del Software. Concetti di: progetto, attività, risorsa, task, workproduct, sistema, modello, documento, obiettivi (goal), requisiti, vincoli, notazioni, metodi e metodologie.
2	Le fasi principali dello sviluppo: raccolta requisiti, analisi dei requisiti, progetto di sistema, progetto esecutivo o degli oggetti, implementazione, gestione del progetto, testing, ciclo di vita del software.
3	Introduzione all'UML, diagrammi dei casi d'uso, diagrammi delle classi, diagrammi di sequenza e collaborazione. Diagrammi di stato diagrammi di attività, diagrammi di dislocazione, organizzazione dei diagrammi, estensione dei diagrammi, concetti di sistema, modello e vista. Modellazione object-oriented.
4	Introduzione alla raccolta dei requisiti. Concetti fondamentali: requisiti funzionali, requisiti non funzionali e pseudo-requisiti, livelli di descrizione. Principali attributi delle specifiche (correttezza, completezza, …). Classificazione delle attività di raccolta del requisiti.
6	Analisi dei requisiti: Identificazione degli attori, degli scenari, dei casi d'uso, dei casi d'uso, delle relazioni tra attori e casi d'uso. Identificazione degli oggetti d'analisi, identificazione dei requisiti non funzionali. Modelli di analisi: funzionale, degli oggetti, dinamico. Concetti di analisi: oggetti di tipo entità, di confine e di controllo. Passaggio dai casi d'uso agli oggetti, identificazione degli oggetti entità, di confine, di controllo, modellazione delle interazioni, identificazione delle associazioni e degli attributi, modellazione del comportamento degli oggetti.
5	Progettazione di sistema. Introduzione, concetti e attività principali. Architetture software: client-server peer to peer, pipe and filter. Attività della progettazione di sistema. Identificazione dei sottosistemi. La mappatura dei sottosistemi su processori e componenti. La definizione dei depositi di dati. La definizione dal controllo l'accesso. La progettazione del flusso di controllo: procedure-driven, event-driven, threads. La progettazione delle condizioni di confine.

5	Progettazione di dettaglio. Introduzione, concetti della progettazione del modello ad oggetti. Attività della progettazione del modello oggetti; oggetti d'analisi e oggetti della soluzione. Tipi, signature, visibilità. Contratti: invarianti, pre-condizioni, post-condizioni. Object Constraint Language. Identificazione degli attributi e delle operazioni mancanti; specifica di tipi, signature e visibilità. Specifica dei vincoli (constraint), delle eccezioni; identificazione e adattamento delle librerie di classi; realizzazione delle associazioni. Incrementare il riuso.
4	Testing. Introduzione al testing; tecniche di controllo della qualità; tecniche per evitare guasti; tecniche per la scoperta dei guasti tecniche per tollerare i guasti. Concetti di test: componente, guasto, errore, malfunzionamento,test case, test stub/driver, correzione. Attività di testing: ispezione dei componenti; unit testing; test di integrazione; test di sistema. - test di integrazione: strategie del test di integrazione (big bang, bottom-up, top-down, sandwich, modified sandwich). - test di sistema: test funzionale, test di performance, test pilota (alpha test, beta test), test di accettazione e test di installazione. Pianificazione del test; documentazione del test.
4	Project Management. Elementi fondamentali di project management Caratteristiche fondamentali del progetto Attività (ordinarie, di riepilogo, cardine). Struttura delle attività. Relazioni tra attività. Le risorse (il calendario, i costi). I costi fissi del progetto. Le relazioni di progetto. Il progetto iniziale. Il progetto con previsioni. Il progetto con variazioni.

LEZIONI DI LABORATORIO ED ESERCITAZIONI	
13	Utilizzo di CASE tool per la modellazione UML. Esempi ed esercizi di: - raccolta e documentazione dei requisiti. - analisi e documentazione dei requisiti. - definizione e documentazione della progettazione di sistema. - definizione e documentazione della progettazione di dettaglio. - definizione, pianificazione e documentazione dei casi di test - project management

TESTI CONSIGLIATI	Ian Sommerville Ingegneria del software 8/Ed. 2007 pp. 848 ISBN 9788871923543 Object-Oriented Software Engineering Using UML, Patterns, and Java, 3/E Bernd Bruegge, Adjunct, Carnegie Mellon University Allen H. Dutoit, Technical University of Munich ISBN-10: 0136061257 ISBN-13: 9780136061250 Publisher: Prentice Hall Copyright: 2010