Principali informazioni sull'insegnamento	**A.A. 2017-2018**
Titolo insegnamento	Ingegneria del Software (A-L)
Corso di studio	Informatica
Crediti formativi	9
Denominazione inglese	Software Engineering
Obbligo di frequenza	No
Lingua di erogazione	Italiano

Docente responsabile	Nome Cognome	Indirizzo Mail
	Filippo Lanubile	filippo.lanubile@uniba.it
Luogo ed Orario di Ricevimento	Dip. Informatica 6° Piano	Venerdì dalle 12:00 alle 14:00

Dettaglio credi formativi	Ambito disciplinare	SSD	Crediti
	Informatico	INF/01	9

Modalità di erogazione	
Periodo di erogazione	Secondo Semestre
Anno di corso	Secondo Anno
Modalità di erogazione	Lezioni frontali Esercitazioni in aula e laboratorio

Organizzazione della didattica	
Ore totali	225
Ore di corso	71 (56 lezioni frontali e 15 esercitazioni/laboratorio)
Ore di studio individuale	154 (119 lezioni frontali, 10 esercitazioni/laboratorio, 25 progetto)

Calendario	
Inizio attività didattiche	26 febbraio 2018
Fine attività didattiche	1giugno 2018

Syllabus	
Prerequisiti	Deve essere stato colmato l'eventuale debito formativo secondo quanto previsto dal Regolamento Didattico del Corso di Studi. È fortemente consigliato il superamento degli esami dei seguenti insegnamenti del I anno: Programmazione., Laboratorio di Informatica, Linguaggi di Programmazione.
Risultati di apprendimento previsti (declinare rispetto ai Descrittori di Dublino) (si raccomanda che siano coerenti con i risultati di apprendimento del CdS, compreso i risultati di apprendimento trasversali)	• *Conoscenza e capacità di comprensione* Conoscere principi fondanti dell'ingegneria del software Conoscere le diverse tipologie di applicazioni software Conoscere le differenze tra i processi di sviluppo del software, in particolare tra modello waterfall e modello agile Conoscere i diversi aspetti modellabili del software e le possibili finalità della modellazione Conoscere le soluzioni di successo a problemi noti e

	ricorrenti nella progettazione del software • *Conoscenza e capacità di comprensione applicate* Sviluppare competenze applicabili allo sviluppo ed evoluzione di sistemi software, aventi caratteristiche di qualità, economicità e tempestività. Sviluppare la capacità di creare modelli, sia con prospettiva concettuale che con prospettiva software. Acquisire familiarità con lo sviluppo di sistemi software la cui realizzazione richiede lo sforzo congiunto e prolungato di un team di lavoro. • *Autonomia di giudizio* Mostrare di aver acquisito autonomia di giudizio sulle scelte relative allo sviluppo dei sistemi software. • *Abilità comunicative* Mostrare di essere in grado di comunicare in modo appropriato le caratteristiche di prodotto e di processo nello sviluppo del software. • *Capacità di apprendere* Mostrare di aver sviluppato capacità di intraprendere in autonomia ulteriori approfondimenti su argomenti attinenti l'ingegneria del software.
Contenuti di insegnamento	• Introduzione all'ingegneria del software • Processi software: agile vs. waterfall • Controllo delle versioni; git e GitHub • Ingegneria dei requisiti • Introduzione a UML • Requisiti funzionali; user story e validazione mediante meeting di revisione • Modellazione del dominio di un problema • Progettazione dell'architettura; pattern architetturali • Modellazione della struttura • Modellazione del comportamento • Realizzazione di una storia utente; design pattern • Verifica: analisi statica, code review e automazione del test • Manutenzione ed evoluzione

Programma	
Testi di riferimento	Martin Fowler. UML Distilled. Pearson. P. Bourque and R.E. Fairley, eds., Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014; www.swebok.org.
Note ai testi di riferimento	Altri libri consigliati: Carlo Ghezzi, Medhi Jazayeri, Dino Mandrioli, "Ingegneria del Software, Fondamenti e Principi" Pearson.

	Steven Metsker "Design pattern in java: manuale pratico", 2003, Pearson Italia. A. Fox, D. Patterson. Engineering Software as a Service: An Agile Approach Using Cloud Computing, 1st edition. ISBN-13: 978-0984881246 Ian Sommerville. Ingegneria del software. Pearson I libri di testo sono integrati con gli appunti presi a lezione e con le slide del docente disponibili sul sito web del corso.
Metodi didattici	Lezioni frontali supportate da slide, esercitazioni in aula, assegnazioni di esercizi di programmazione di rete con verifica in laboratorio.
Metodi di valutazione (indicare almeno la tipologia scritto, orale, altro)	Progetto e prova scritta. La prova scritta consiste nel rispondere a un questionario contenente domande a risposta chiusa o aperta. Il progetto consiste nel completare lo sviluppo di un sistema software assegnato dal docente, seguendo direttive sui deliverable anche queste dettate dal docente. Per gli studenti che hanno almeno il 70% di presenze del numero di ore di lezioni effettuate nel periodo precedente l'interruzione delle lezioni è prevista la possibilità di svolgere le attività di progetto con feedback intermedi fino alla conclusione del corso. Per gli studenti che non frequentano attivamente, la comunicazione di consegna del progetto secondo le date degli appelli. Un progetto valutato positivamente non può essere sottoposto a ulteriore valutazione. La valutazione positiva di un progetto è valida per tutti gli appelli dell'anno accademico corrente. La valutazione positiva Sia la prova scritta che la prova di progetto sono valutate in trentesimi. Il voto finale è la media aritmetica delle due prove.
Criteri di valutazione (per ogni risultato di apprendimento atteso su indicato, descrivere cosa ci si aspetta lo studente conosca o sia in grado di fare e a quale livello al fine di dimostrare che un risultato di apprendimento è stato raggiunto e a quale livello)	Saranno valutati i risultati di apprendimento previsti così come descritti precedentemente.
Altro	