Requisiti per il progetto di ingegneria del software

AA 2025/26

Versione 1.0.0 del 26/9/25

Versione	*Modifiche*
1.0.0	Versione iniziale da quella dello scorso anno Confermato Papyrus anche quest'anno e aggiunta un frase per valutazione personale

Angelo Gargantini e Silvia Bonfanti

Per il corso di ingegneria del software dovete consegnare un progetto (vedi modalità d'esame spiegate alla prima lezione) in cui dimostrate di sapere applicare ad un caso di studio abbastanza corposo, i concetti e le metodologie e le tecniche studiate durante il corso. Il progetto deve contenere tutto, dai requisiti/planning fino all'implementazione/testing. Per ogni capitolo del libro di seguito sono elencati i requisiti.

Alcune domande tipiche:

Il progetto è obbligatorio?
Sì

Come si consegna il progetto?
È obbligatorio mettere il tutto in un repository github da condividere preventivamente con il professore (account github: *garganti*) e con l'esercitatrice (account github: *silviabonfanti*). Anche per le eventuali comunicazioni si possono usare gli issue. Ad esempio se volete approvato il project plan, aprite un issue e assegnatelo al professore.

Quando si consegna?
Il project plan deve essere pronto un mese circa prima dell'esame. Il progetto deve essere completato 5 giorni prima della sua presentazione.

Posso fare l'esame scritto e pensare al progetto in un secondo momento?
No. Il progetto fa parte integrale con l'esame, così potete mettere in pratica i concetti teorici mentre li studiate direttamente (pensate ad esempio al versioning). Quindi per far sostenere l'esame scritto è obbligatorio avere un progetto e avere il suo project plan approvato (tramite issue). Se il progetto è insufficiente potete ripresentarlo (con una leggera penalizzazione). Se il progetto è fatto in gruppo ma non tutti fate l'esame, allora dovete presentare tutti il progetto che poi verrà sommato quando farete l'esame.

Deve concentrarsi più sulla creazione del software o più sulla documentazione?

Nel corso diamo molta enfasi alla modellazione, alla specifica dei requisiti, alla progettazione, quindi mi aspetto che almeno il 70% del tempo andrà speso su quelle attività.

Quali diagrammi UML devo presentare?
Tutti quelli visti a lezione: use case diagram, class diagram, state machine diagram, sequence diagram, communication diagram/timing diagram, activity diagram, component diagram e package diagram. Si DEVE anche caricare il progetto Papyrus contente i diagrammi.

Dovremo affrontare l'implementazione utilizzando il Papyrus Designer per la generazione del codice oppure scrivendo codice per conto nostro?
Qualche classe mi aspetto che sia generata e poi magari modificata, però gran parte del codice sarà scritta manualmente (ad esempio i corpi dei metodi). Indicate nella relazione se e quale classe avete generato da UML.

Il software finale quanto deve essere implementato rispetto al documento?
Dipende molto dalla grandezza del progetto, ma direi quasi la totalità (per evitare di avere un progetto gigantesco implementato solo in modo molto frammentario).

Il progetto deve essere fatto mediante Java o può essere scritto in un altro linguaggio?
Il progetto dallo scorso anno deve essere scritto in Java come progetti Eclipse.

Il progetto va anche sviluppato?
Sì va fatta anche l'implementazione e i casi di test Junit vanno scritti.

Quante persone possono far parte del team?
Lo scopo del progetto è sviluppare un sw in modo collaborativo. Il team è normalmente composto da 2 (minimo) o 3 persone (massimo). Accettiamo anche team da 4 ma deve essere ancora più chiaro chi ha fatto cosa (ad esempio dall'organizzazione del team e dai log di github). Chi non ha persone con cui formare il team può contattarci e cerchiamo di aggregare i team. In casi particolari accettiamo team con 1 sola persona.

Per quanto riguarda il personale, dobbiamo scrivere i compiti seguiti da noi che abbiamo svolto il progetto, oppure immaginare una situazione reale (quindi con PM, sistemisti e programmatori)?
Come personale e ruoli riferitevi a quello effettivamente fatto (o quello che avete pianificato di fare).

Quanti punti pesa il progetto
Il progetto peserà fino a 5 punti + eventuale lode. L'esame scritto quindi sarà fino a 27 punti.

Quante ore di lavoro posso prevedere per il progetto?
Secondo I nostri calcoli, almeno circa 30-40 ore a testa.

Si può usare un tema d'esame del corso di programmazione a oggetti o è troppo semplice come progetto?
No è troppo semplice. Per un quel tema d'esame che fate il 4 ore (al I anno), posso immaginare un impegno di 10 ore, troppo poche.

Esistono dei progetti tipo a cui ispirarsi o tracce già pronte?
Potete usare gli esercizi fatti in classe soprattutto per UML, tipo la biblioteca.

Quando presentiamo i progetti?

Il progetto va presentato il giorno dell'orale. Il tempo a disposizione è limitato: max 15 min per 2 partecipanti, 20 min per 3 partecipanti per la presentazione.

Come devo fare la presentazione del progetto?
Quando presentate il progetto usate una presentazione. Nella presentazione dovete far capire cosa avete fatto delle cose richieste (vedi qui sotto dopo), non tanto spiegare quello che fa il vostro programma. Abbiamo preparato un template per la presentazione che trovate sotto "Documentazione progetto".

Quanti documenti devo presentare e in che formato?
A parte il project plan che deve essere pronto prima, consigliamo di organizzare tutta la documentazione in un unico pdf (magari organizzato a capitoli) così basta consegnare quello. In seguito è indicato il contenuto dei capitoli. Potete anche organizzare di documenti separati o usare markdown su github (vedi dopo)

Deve essere funzionante?
Il programma deve essere funzionante, quindi dovete caricare tutti i file che servono per farlo funzionare e spigare come si dovrebbe fare per farlo funzionare. Se è un progetto java semplice ad esempio basta mettere sul repo il progetto o i progetti java che servono per eseguire i casi di test e se necessario farlo andare. Se invece è un progetto più complesso potrebbe contenere diversi sotto-progetti.

All'esame orale devo eseguirlo?
Sì all'orale potrebbe venirvi richiesto di fare una piccola demo o eseguire i test. Quindi preparate almeno un PC su cui potete fare vedere il vostro progetto in esecuzione. Il progetto verrò eseguito anche dai docenti per controllare il suo funzionamento.

Per quanto è valido il progetto?
Se discutete il progetto insieme ad un vostro compagno che ha passato l'esame ma voi non l'avete passato o sostenuto (vedi tabella seguente), il progetto vale anche per la sessione successiva. Ad esempio se discutete il progetto a febbraio, potrete fare l'esame anche d'estate (fino a settembre). Se invece il progetto non è sufficiente ma avete passato lo scritto, sostenete la discussione del progetto migliorato nell'appello successivo.

Devo usare per forza github?
Sì dovete usare github o un sistema per la collaborazione/versioning per ospitare il vostro progetto. Nota che non dovete usare il repository solo per consegnare i file (ad esempio, non va ve ben caricare alla fine del progetto lo zip su github). Deve proprio far parte del vostro processo di sviluppo (caricare i file di documentazione a mano a mano li modificate). Sia che sia in java che qualsiasi linguaggio di programmazione. Dovete committare l'intero progetto, non semplicemente caricare i file. Per verificare che avete fatto un buon uso di github verrà verificato che tutti abbiano fatto i commit sul repository, eventuali branch utilizzati, e le issue aperte/chiuse. Sul repository non caricate file non necessari usando i file .gitignore. Ricordatevi di caricare su github il file ".project"

Come funziona la consegna nel caso in cui i diversi componenti del team fanno l'esame in tempi diversi?

Piero e Anna sono due amici e fanno il progetto insieme				
Ad un appello:				
Piero	Anna	Piero	Anna	
SCRITTO		**PROGETTO**		**RISULTATO**
NON PASSA	NON PASSA/ NON SI PRESENTA			Non possono discutere il progetto, devono fare lo scritto entrambi
PASSA	PASSA	PASSA	PASSA	Entrambi registrano l'esame
PASSA	NON PASSA/ NON SI PRESENTA	PASSA	PASSA	Piero registra, Anna dovrà rifare l'esame, ma il suo progetto è valido (deve presentarsi comunque all'orale per discutere il progetto)
PASSA	PASSA	NON PASSA	NON PASSA	Anna e Piero devono ripresentare il progetto.
PASSA	PASSA	NON SI PRESENTA	PASSA	Anna passa, Piero (a meno che abbia qualche valido motivo) viene considerato assente e dovrà rifare l'esame e il progetto.
PASSA	PASSA	NON PASSA	PASSA	(raro) Anna registra e Piero presenta lo stesso progetto al prossimo orale, oppure se vuole rifa lo scritto
PASSA	NON PASSA/ NON SI PRESENTA	PASSA	NON PASSA/ NON SI PRESENTA	Piero registra, Anna dovrà rifare l'esame e presentare un progetto diverso all'orale.
PASSA	NON PASSA/ NON SI PRESENTA	NON SI PRESENTA		Nessuno può presentare il progetto all'orale, Piero dovrà fare l'orale (e può rifare lo scritto) Anna deve sostenere l'intero esame

Presentazione:
per la presentazione dovrai usare il template

Linguaggio di programmazione: Java
Il progetto dovrà essere scritto in Java – usando il più possibile le caratteristiche di java (quindi packages, interfacce, ma anche collection framework, magari la parte funzionale e le novità del linguaggio.
Per la parte web se è una web app usate JSP, vaadin o JEE con le servelt
Motivazione: è una occasione per usare java in un progetto complesso, imparare quindi alcuni dettagli del linguaggio che aiutano ad usarlo in modo professionale. Permette a noi docenti di capire e valutare meglio il progetto

Uso di eclipse come IDE
Come IDE è obbligatorio l'uso di eclipse (se uno vuole può usare anche un altro editor, l'importante che si possa poi aprire senza errori come progetto eclipse).
Dovrà quindi essere uno o, meglio, più progetti eclipse.

Motivazione: usiamo uno standard di IDE professionale e ci adeguiamo a quello

Gestione delle dipendenze
Usiamo MAVEN per le dipendenze
Ci dovrà essere almeno una dipendenza (tipo log4j)

Uso efficace di github
Github dovrà essere usato come piattaforma per la cooperazione (quindi ad esempio la programmazione da parte di diversi team), con tutte le funzionalità. Non vogliamo che sia usato come strumento per semplicemente caricare i file elaborati in locale.

Caratteristiche dei progetti eclipse

- Dovranno oltre avere le dipendenze gestite con maven e quindi essere progetti maven,
- I test in un folder separato – ad esempio per il progetto maven-demo:

```
maven-demo/
├ pom.xml
└ src
  ├ main
  │ └ java
  │   └ com
  │     └ trcooke
  │       └ mvndemo
  │         └ App.java
  └ test
    └ java
      └ com
        └ trcooke
          └ mvndemo
            └ AppTest.java
```

- Non contenere alcun riferimento a file in modo assoluto al computer dove vengono eseguiti
- Database dovrà essere embedded
- No servizi esterni da installare (nel caso siano necessari concordare con il professore)
- L'intero codice dovrà essere su github con le modifiche non upload

Nome dei progetti eclipse
Per favore nominate i progetti eclipse con una sigla iniziale che ricorda chi siete (ad esempio le vostre iniziali) oppure date un nome caratteristico alla app e usate quello. Ad esempio, se volessi usare il mio nome, potrei nominare i progetti così:
GAR_database progetto che contiene il db e il codice per accedere
GAR_gui progetto che contiene il codice con la cui
...
Mi aiuta quando importo i vostri progetti. Non usare nomi generici tipo "progetto ing del sw". Il nome serve per identificare il vostro progetto.

Come organizzare i folder di github
Separate il codice dai documenti – per esempio i progetti eclipse potrebbero essere sotto code e i documenti sotto documenti

Ad esempio una struttura del genere:

spiegazione	Nome del folder/sottofolder			
Root del repository (nome del repo su github) e nome del mio software	BiblioMat			
Folder con tutti i documenti	→	documenti		
	→	codice		
Progetto eclipse		→	bibliomat	
File del progetto			.project pom.xml .classpath	
Folder con i sorgenti (con maven potrebbe differire)			→	src
Folder con i test			→	test

Cosa mettere nel .gitignore
Mettere nel .gitignorre i file generati tipo .class e il folder target

ATTENZIONE: non mettete i .project nel .gitignore altrimenti quando farò il checkout non riuscirò ad importare i vostri progetti.

VALUTAZIONE INDIVIDUALE DEI PROGETTI (new 2526)

Chiederemo ad ognuno dei partecipanti nel progetto di valutare il proprio impegno e quello dei propri compagni in modalità privata (non verrà resa pubblica, probabilmente mediante mail).

Se tutti si sono impegnati in modo uguale – come auspicabile -- (facendo le medie di tutti i partecipanti) il voto sarò uguale per tutti.

Se qualcuno si sarà impegnato di meno (comunque entro una misura ragionevole) il voto sarà diverso per i partecipanti al progetto. Se i partecipanti sono n, sul totale dei punti, il 1/n viene dato comunque a tutti mentre i n-1/n vengono distribuiti in modo proporzionale all'impegno. Ad esempio in 2, metà del voto è distribuito.

Questo ha due obiettivi:

1. Spingere tutti a contribuire in modo equo al meglio delle proprie possibilità
2. Riconoscere se qualcuno per diverse ragioni ha contribuito di più

Non vuol dire però che un partecipante possa non fare nulla o quasi nulla, in quel caso dovrà rifare il progetto trovando un altro gruppo. NON è ammessa la partecipazione senza un minimo significativo contributo.

RELAZIONE FINALE

In questa sezione vengono indicati i contenuti del documento di relazione finale. Ogni parte deve essere o un capitolo delle relazione, o un documento (tipo pdf o md). Vedi dopo.

Software Engineering Management

(capitolo 2 del libro)

DEVE contenere un PROJECT PLAN per il progetto - con i 14 punti come indicato in sezione 2.1 del libro.

Il project plan DEVE essere consegnato (condiviso) **almeno 1 mese prima** dell'esame.

POTREBBE essere modificato tenendo traccia delle versioni (vedi tabellina all'inizio di questo file).

Documentazione: il documento deve contenere queste sezioni

1. Introduction
2. Process model
3. Organization of the project
4. Standards, guidelines, procedures
5. Management activities
6. Risks Potential
7. Staffing
8. Methods and techniques
9. Quality assurance
10. Resources
11. Budget and schedule
12. Changes
13. Delivery

Software Life Cycle

(capitolo 3 del libro)

DEVE indicare il tipo di processo di sviluppo seguito

PUO' formalizzare con diagramma UML una piccola parte del sw development (ad esempio una change request come viene trattata)

DOVREBBE anche usare un approccio MDA o parte di esso (con Yakindu o Papyrus UML come visto in classe)

POTREBBE presentare il progetto come SPL

Documentazione: breve documento in cui indicare il tipo di processo seguito effettivamente. Eventuali differenze rispetto a quanto previsto nel project plan vanno indicate qui. Anche ad esempio come sono stati organizzati il timebox o gli sprint (ad esempio con le date) vanno indicati qui.

Configuration Management

(capitolo 4 del libro)

DEVE usare un sistema per il configuration management (consigliato github) con i relativi comandi (git add, commit, push etc.)

DEVE dimostrare di aver usato issuses, branches, pull requests, e code reviews

PUO' usare la project board (tipo kanban o varianti simili)

Documentazione: aggiungere al documento eventuali strumenti usati. Gli issue e l'uso di github è da vedere sul sito github (non è necessario documentarlo ma si possono portare alcune statistiche come numero di commit, di issues e cose del genere).

People Management and Team Organization

(capitolo 5 del libro)

DEVE dire le persone e l'organizzazione del lavoro tra le persone (rifacendosi ad uno schema del libro)

Documentazione: aggiungere al documento (1) la documentazione riguardante questo punto.

Software Quality

(capitolo 6 del libro)

POTREBBE contenere una lista di qualità che hanno guidato il progetto con la loro spiegazione relativa al contesto

Documentazione: aggiungere al documento la documentazione riguardante questo punto.

Requirement Engineering

(capitolo 9 del libro)

DOVREBBE indicare come i requisiti sono stati elicitati

DEVE contenere la specifica dei requisiti (ad esempio seguendo lo IEEE 830)

Documentazione: Documentare i requisiti in documento. La descrizione dei requisiti può seguire lo schema visto in classe dei requisiti presentati in

https://github.com/foselab/abz2024_casestudy_MLV/blob/main/Mechanical_Lung_Ventilator%201_5.pdf

Al suo intero il documento dovrebbe riportare i casi d'uso ed eventuali altri diagrammi UML (ad esempio macchine di stato)

Modelling

(Libro UML @ Classroom)

DEVE contenere i diagrammi UML visti a lezione, se non già presentati in altre sezioni della documentazione:

- use case diagram (si consiglia di metterlo nella sezione dei requisiti)
- class diagram (dal quale si DEVE generare una prima versione del codice in Java usando Papyrus Designer - si consiglia di metterlo nella sezione software design)
- ALMENO UNO state machine diagram (NON DEVE essere troppo semplice, si deve cercare di usare la maggior parte degli elementi che costituiscono uno state diagram come visto a lezione)
- ALMENO UN sequence diagram (NON DEVE essere troppo semplice, si deve cercare di usare la maggior parte degli elementi che costituiscono un sequence diagram come visto a lezione).
- UN diagramma TRA communication diagram e timing diagram
- ALMENO UN activity diagram (NON DEVE essere troppo semplice, si deve cercare di usare la maggior parte degli elementi che costituiscono un activity diagram come visto a lezione)
- component diagram (si consiglia di metterlo nella sezione in cui si presenta la software architecture)
- package diagram (si consiglia di metterlo nella sezione in cui si presenta la software architecture o insieme al class diagram)

Documentazione: immagini nel documento o in una cartella separata (specificare qui quale è la cartella)

Software Architecture

(capitolo 11 del libro)

DEVE contenere la descrizione dell'architettura con almeno un paio di architectural views (per differenti punti di vista)

DOVREBBE avere almeno una vista con connettori e componenti con la descrizione dello stile architetturale (11.4)

DEVE utilizzare almeno una libreria esterna con maven. Ad esempio l'uso di log4j è molto consigliata.

Software Design

(capitolo 12 del libro)

DEVE contenere una descrizione del design (mediante i diagrammi UML va bene)

POTREBBE contenere un calcolo di complessità (ad esempio con McCabe) di una piccola parte

DOVREBBE contenere qualche misurazione del codice, (con qualche metrica che abbiamo visto). Alcuni tools che vedremo a lezione: stanide, jdepend, struture101, sonarlint, PMD …

DEVE applicare un paio di design pattern visti a lezione

Documentazione: Documentare l'architettura e il design in documento.

Software Testing

(capitolo 13 del libro)

PUO' avere un documento di plan per l'attività di test

DEVE contenere dei casi di test di unità implementati con la loro descrizione nel documento

DOVREBBE avere qualche misura di copertura per i casi di test

Documentazione: Documentare i test in un documento.

Software Maintenance

(capitolo 14 del libro)

POTREBBE contenere un di attività di reverse engineering (se si è partiti da codice esistente)

DOVREBBE documentare **alcune** attività di refactoring che sono state fatte.

Documentazione: Documentare eventuali attività di refatoring in un documento.

Quali documenti devono essere consegnati per la relazione finale? Cosa devono contenere?

Per ogni contenuto è indicato come documentarlo (vedi sezione precedente).

Il contenuto deve essere suddiviso in documenti come indicato sotto.

Con documento si intende o un **documento** separato (doc o pdf) o un **capitolo** oppure anche una **pagina** markdown.

- DOCUMENTO 1: PROJECT PLAN
  Contenuto:
  - Software Engineering Management
- DOCUMENTO 2: GESTIONE DEL PROGETTO
  Contenuto:
  - Software Life Cycle
  - Configuration Management
  - People Management and Team Organization
- DOCUMENTO 3: REQUISITI
  Contenuto:
  - Requirements Engineering
  - Software Quality
- DOCUMENTO 4: DESIGN
  Contenuto:
  - Modelling (I diagrammi possono anche essere distribuiti come suggerito sopra)
  - Software Architecture
  - Software Design
    pattern
- DOCUMENTO 5: TESTING
  Contenuto:
  - Software Testing

- DOCUMENTO 6: MAINTENANCE
  Contenuto:
    - Software Maintenance