**Domanda (punti 8/30)**

Lo sviluppo incrementale e quello agile entrambe suddividono il tempo (lungo) di progetto in periodi (più corti) di avanzamento. Questo quesito vi chiede di discutere le differenze (o le somiglianze) che individuate nel modo in cui quei due modelli interpretano l’articolazione di tali periodi, per determinazione della loro durata, l’individuazione dei loro specifici obiettivi, le modalità di gestione (cioè pianificazione, esecuzione, e verifica). Fornirete la vostra risposta in modo strutturato, descrivendo esplicitamente:

* A quale parte di domanda state rispondendo.
* Quale sia la vostra risposta a essa.
* Sulla base di cosa lo avete pensato.

**Risposta**

Determinazione della durata del periodo:

* Sviluppo incrementale: I periodi, o incrementi, sono tipicamente determinati dalla dimensione e dalla complessità dei compiti da completare al loro interno. La durata di ciascun incremento può variare, ma in genere è più breve rispetto allo sviluppo agile.
* Sviluppo agile: I periodi, o sprint, sono tipicamente impostati su una durata fissa, di solito da 1 a 4 settimane. Ciò consente una cadenza costante dei progressi e permette al team di pianificare e impegnarsi a realizzare una serie di risultati entro quel lasso di tempo.

Identificazione di obiettivi specifici:

* Sviluppo incrementale: Gli obiettivi specifici per ogni incremento sono determinati in base agli obiettivi generali del progetto e alle priorità delle parti interessate. Questi obiettivi devono essere specifici, misurabili, raggiungibili, pertinenti e limitati nel tempo (SMART).
* Sviluppo agile: Gli obiettivi specifici per ogni sprint sono determinati dal team in base agli obiettivi e alle priorità generali del progetto. Questi obiettivi sono catturati nello sprint goal, che è una dichiarazione di alto livello di ciò che il team spera di raggiungere nello sprint.

Gestione:

* Sviluppo incrementale: Pianificazione, esecuzione e verifica sono tutti aspetti importanti dello sviluppo incrementale. La pianificazione comporta la determinazione degli obiettivi e dei compiti specifici per ogni incremento, nonché l'identificazione di eventuali dipendenze o rischi. L'esecuzione comporta l'effettivo completamento dei compiti all'interno dell'incremento, che può comportare la collaborazione con altri team o stakeholder. La verifica comporta l'esame dei progressi e dei risultati dell'incremento per assicurarsi che stia raggiungendo gli obiettivi desiderati.
* Sviluppo agile: Anche la pianificazione, l'esecuzione e la verifica sono aspetti importanti dello sviluppo agile. La pianificazione comporta la determinazione degli obiettivi e dei compiti specifici per ogni sprint, nonché l'identificazione di eventuali dipendenze o rischi. L'esecuzione comporta il completamento dei compiti all'interno dello sprint, che può comportare la collaborazione con altri membri del team o con le parti interessate. La verifica comporta l'esame dei progressi e dei risultati dello sprint, per garantire il raggiungimento degli obiettivi desiderati. Una differenza fondamentale tra lo sviluppo incrementale e lo sviluppo agile in termini di gestione è l'uso di cerimonie agili, come stand-up giornalieri, pianificazione dello sprint e revisione dello sprint, per facilitare la comunicazione e la collaborazione tra i membri del team.

**Domanda (punti 8/30)**

Vogliamo ragionare sulla relazione che intercorre tra i concetti di “processo” e di “progetto”, facendo riferimento a situazioni concrete, direttamente sperimentate. Nello specifico, vogliamo capire se (1) un progetto, ovvero l’insieme di attività da esso mobilizzate, possa essere descritto, spiegato, compreso, in termini di una qualche articolazione organizzata di processi, e se (2), conseguentemente, la definizione dei secondi debba o meno preesistere all’organizzazione del primo. Per rispondere alle due domande, scegliete uno tra due approcci: (a) analizzare un dato segmento del vostro progetto didattico, individuando le attività previste in esso, e attribuendo ciascuna di esse a un processo di appartenenza, oppure (b) “dispiegare” alcuni processi per voi essenziali e “mapparli” su un tratto del vostro progetto, scendendo in entrambi i casi a livello di singole attività elementari della dimensione dei vostri normali “ticket”. La risposta dovrà anche spiegare come (3) le relazioni di precedenza intercorrenti tra specifiche attività di progetto si riflettano all’interno dei processi di appartenenza e nelle relazioni tra di essi.

**Risposta**

Per rispondere a queste domande, adotterò l'approccio (a) e analizzerò un determinato segmento di un progetto didattico, identificando le attività in esso previste e attribuendo ciascuna di esse a un processo a cui appartiene.

In questo progetto didattico, stiamo creando una serie di video per insegnare una nuova applicazione software ai nostri utenti. Le attività individuate in questo progetto comprendono:

* Ricerca dell'applicazione software e raccolta di informazioni sulle sue caratteristiche e funzionalità. Questa attività rientra nel processo di pianificazione e preparazione.
* Sviluppo di una sceneggiatura per i video basata sulla ricerca condotta. Questa attività rientra nel processo di creazione dei contenuti.
* Registrazione dei video con un software di cattura dello schermo. Questa attività appartiene al processo di produzione.
* Montaggio dei video per aggiungere transizioni, didascalie e altri miglioramenti. Questa attività appartiene al processo di post-produzione.
* Caricare i video sul nostro sistema di gestione dell'apprendimento e creare materiali di accompagnamento come quiz e valutazioni. Questa attività appartiene al processo di distribuzione e promozione.

Le relazioni di precedenza tra queste attività specifiche del progetto si riflettono nei processi a cui appartengono. Ad esempio, l'attività di ricerca dell'applicazione software deve essere completata prima dell'attività di sviluppo della sceneggiatura, che deve essere completata prima dell'attività di registrazione dei video. Allo stesso modo, l'attività di registrazione dei video deve essere completata prima dell'attività di editing dei video, che deve essere completata prima dell'attività di caricamento dei video sul sistema di gestione dell'apprendimento.

In conclusione, un progetto può essere descritto, spiegato e compreso in termini di un'articolazione organizzata di processi. La definizione dei processi deve preesistere all'organizzazione del progetto per guidare le attività e garantire che siano completate nell'ordine corretto. Le relazioni di precedenza tra le attività specifiche del progetto si riflettono all'interno e nelle relazioni tra i processi a cui appartengono. I processi sono singole attività svolte all’interno del progetto che trasformano ingressi (bisogni) in uscite (prodotti) secondo regole date, consumando risorse nel farlo, mentre il progetto è definito uniformemente come insieme di processi, secondo degli ingressi predeterminati e uscite determinate dall’implementazione (design).

# Domanda (punti 8/30)

Per alcuni, la tupla <decisione, azione, verifica> costituisce l’unità base, tassello atomico (unitario e inscindibile), dei metodi di pianificazione, gestione, ed esecuzione di progetto.

Riflettendo sulla vostra esperienza pratica sin qui e su quanto ritenete di avere appreso dallo studio teorico:

* definite singolarmente ogni elemento di tale tupla, in termini pertinenti ai metodi di gestione di progetto che conoscete;
* prendete posizione sull’asserzione di cui sopra, motivandola in modo argomentato, anche attraverso esempi concreti, che la confermino o la confutino.

# Risposta

Gli elementi della tupla <decisione, azione, verifica> nel contesto della gestione del progetto possono essere definiti come segue:

* Decisione: Una decisione è una scelta fatta tra un insieme di alternative sulla base delle informazioni e dei vincoli disponibili. Nella gestione dei progetti, le decisioni sono tipicamente prese dal project manager o da un'autorità decisionale designata e possono riguardare vari aspetti del progetto, come l'ambito, la tempistica, il budget, la gestione del rischio e il coinvolgimento degli stakeholder.
* Azione: L'azione si riferisce all'attuazione di una decisione. Nella gestione dei progetti, le azioni sono tipicamente intraprese dal team di progetto o dai singoli membri del team per eseguire i compiti necessari a raggiungere gli obiettivi del progetto.
* Verifica: La verifica è il processo di controllo della coerenza delle azioni intraprese con la decisione presa. Nella gestione dei progetti, la verifica può comportare l'esame del lavoro completato, il confronto con il piano di progetto e la garanzia che sia conforme agli standard di qualità specificati.

Per quanto riguarda l'affermazione che la tupla <decisione, azione, verifica> è l'unità di base dei metodi di pianificazione, gestione ed esecuzione dei progetti, direi che è generalmente vera, ma con alcune riserve.

Da un lato, il ciclo <decisione, azione, verifica> è un aspetto fondamentale di molti approcci alla gestione dei progetti, comprese le metodologie tradizionali come Waterfall e quelle agili come Scrum. In questi approcci, si prendono decisioni sul lavoro da svolgere, si intraprendono azioni per completare il lavoro e si verificano i risultati per garantire che il progetto sia in linea con i suoi obiettivi.

D'altra parte, è importante notare che il ciclo decisione, azione, verifica non è l'unico elemento della gestione del progetto. Ci sono molti altri fattori che contribuiscono al successo o al fallimento di un progetto, come il coinvolgimento degli stakeholder, la gestione del rischio, la comunicazione e l'allocazione delle risorse. Questi fattori sono spesso interdipendenti con il ciclo di decisione, azione e verifica e devono essere presi in considerazione per pianificare, gestire ed eseguire efficacemente un progetto.

A titolo di esempio, si consideri un progetto per lo sviluppo di una nuova applicazione software. Il project manager potrebbe decidere di utilizzare un certo linguaggio di programmazione e un certo framework di sviluppo, e il team potrebbe agire per implementare l'applicazione utilizzando questi strumenti. Tuttavia, se il project manager non si è impegnato adeguatamente con le parti interessate e non ha raccolto i loro input sui requisiti dell'applicazione, il prodotto finale potrebbe non soddisfare le loro esigenze e il processo di verifica lo rivelerà. In questo caso, il ciclo decisione, azione, verifica non è sufficiente da solo a garantire il successo del progetto; per essere efficace, deve essere integrato con altri fattori, come il coinvolgimento degli stakeholder.

# Domanda (punti 8/30)

Il termine *coverage* ricorre sovente, con diverse accezioni, sfumature e ramificazioni, nel dominio dell’ingegneria del *software*. Questa prova d’esame vi chiede di elencare tutte le accezioni di quel termine che avete effettivamente incontrato nella vostra esperienza sin qui, come concetto solo udito oppure anche praticato. Di ognuna di esse spiegherete precisamente il significato, illustrando anche le modalità di determinazione del valore quantitativo corrispondente.

# Risposta

* Copertura del codice: La copertura del codice si riferisce alla misura in cui una determinata suite di test esercita le varie linee di codice di un'applicazione software. In genere viene misurata in percentuale e può essere determinata eseguendo la suite di test e registrando quali linee di codice sono state eseguite.
* Copertura dei test: La copertura dei test si riferisce alla misura in cui una determinata suite di test esercita le varie caratteristiche o funzionalità di un'applicazione software. Ciò può includere il test di diversi valori di input, il test delle condizioni di errore, il test delle prestazioni, ecc. La copertura dei test è tipicamente misurata in termini di percentuale di caratteristiche o funzionalità testate.
* Test di copertura: Il test di copertura è una tecnica utilizzata per garantire che un'applicazione software sia stata testata a fondo, eseguendo una suite di test e misurando la copertura del codice e la copertura dei test raggiunti. Questo può essere fatto manualmente o con l'uso di strumenti di copertura che tracciano e riportano automaticamente la copertura raggiunta.
* Analisi della copertura: L'analisi della copertura si riferisce al processo di revisione dei risultati dei test di copertura per identificare le aree dell'applicazione software che potrebbero essere state testate in modo inadeguato o che potrebbero richiedere ulteriori test. Questa operazione può essere eseguita manualmente o con l'uso di strumenti di copertura che forniscono funzionalità di analisi e visualizzazione.
* Statement coverage: La copertura delle dichiarazioni è un tipo di copertura del codice che misura la percentuale di dichiarazioni in un'applicazione software che sono state eseguite durante il test. È una misura semplice di copertura che non tiene conto della complessità o dell'importanza delle istruzioni eseguite.
* Copertura delle diramazioni: La copertura delle diramazioni è un tipo di copertura del codice che misura la percentuale di diramazioni in un'applicazione software che sono state esercitate durante i test. Un ramo è un punto del codice in cui il flusso di esecuzione può andare in più di una direzione, in base all'esito di un'istruzione condizionale.
* Copertura delle funzioni: La copertura delle funzioni è un tipo di copertura del codice che misura la percentuale di funzioni di un'applicazione software che sono state eseguite durante i test. Una funzione è un blocco di codice autonomo che esegue un compito specifico.
* Copertura del percorso: La copertura del percorso è un tipo di copertura del codice che misura la percentuale di possibili percorsi attraverso un'applicazione software che sono stati esercitati durante il test. Un percorso è una sequenza di istruzioni e rami che possono essere eseguiti in un determinato ordine.
* Copertura integrata dei test: La copertura del test integrato si riferisce alla misura in cui una suite di test esercita i vari componenti o moduli di un'applicazione software quando sono integrati e funzionano insieme. Si può misurare in termini di copertura del codice e di copertura dei test ottenuti tra i vari componenti o moduli.
* Copertura del test di sistema: La copertura del test di sistema si riferisce alla misura in cui una suite di test esercita la funzionalità e le prestazioni complessive di un'applicazione software quando viene distribuita in un ambiente simile alla produzione. Ciò può essere misurato in termini di copertura del codice e di copertura dei test raggiunti tra i vari componenti o moduli, nonché di prestazioni e affidabilità dell'applicazione nel suo complesso.

# Domanda (punti 8/30)

Il cosiddetto “ciclo di Deming” descrive una collaudata pratica a supporto del miglioramento continuo, pensata per essere applicata in qualunque ambito di attività. Per testare tale applicabilità, vogliamo studiare come calarla in uno specifico e limitato insieme di attività di lavoro di uno studente di informatica. La risposta dovrà avere quattro parti: (1) una breve descrizione generale del ciclo di Deming che ne colga l’essenza e non sia un copia-incolla di fonte esterna; (2) la specifica dello scenario di interesse: quali attività di lavoro e quale scenario vogliate considerare; (3) la corrispondenza da voi proposta tra la descrizione al punto precedente e lo specifico scenario di interesse; (4) una concreta procedura di applicazione, redatta normativamente, come parte di un ideale *way of working*,

# Risposta

(1) Il ciclo di Deming, noto anche come ciclo PDCA (Plan-Do-Check-Act), è una metodologia di miglioramento dei processi che prevede quattro fasi:

* Pianificare: Identificare un problema o un'opportunità di miglioramento e sviluppare un piano per affrontarlo.
* Fare: Attuare il piano e raccogliere dati sulla sua efficacia.
* Controllare: Analizzare i dati e valutare i risultati dell'implementazione.
* Agire: Apportare modifiche in base ai risultati della valutazione e continuare il ciclo.

(2) Lo scenario di interesse è quello di uno studente di informatica che lavora a un progetto di programmazione per un corso. Le attività specifiche di questo scenario comprendono la scrittura di codice, il debug e il test.

(3) Nel contesto del progetto di programmazione, il ciclo di Deming può essere applicato come segue:

* Pianificazione: Identificare le aree del codice che necessitano di miglioramenti o le aree in cui si verificano bug. Sviluppate un piano per affrontare questi problemi, ad esempio rifattorizzando il codice o aggiungendo ulteriori test.
* Eseguire: Attuare il piano scrivendo o modificando il codice ed eseguendo i test.
* Controllare: Analizzare i risultati dei test e valutare l'efficacia delle modifiche apportate al codice.
* Agire: Apportare modifiche in base alla valutazione, come la correzione di eventuali bug residui o l'ulteriore ottimizzazione del codice.

(4) Una procedura concreta e normata per l'applicazione del ciclo di Deming come parte di un metodo di lavoro ideale potrebbe includere le seguenti fasi:

* Identificare un problema specifico o un'opportunità di miglioramento relativa al progetto di programmazione.
* Sviluppare un piano per affrontare il problema o l'opportunità, compresi obiettivi e traguardi specifici.
* Attuare il piano scrivendo o modificando il codice ed eseguendo dei test.
* Analizzare i risultati dei test e valutare l'efficacia delle modifiche apportate al codice.
* Apportate modifiche in base alla valutazione e continuate il ciclo fino a quando il problema o l'opportunità di miglioramento non sono stati affrontati in modo soddisfacente.
* Documentate i risultati del ciclo, comprese le modifiche apportate al codice, i risultati dei test e le lezioni apprese.
* Ripetete il ciclo per qualsiasi altro problema o opportunità di miglioramento identificati.

# Domanda (punti 8/30)

Sani principi di gestione di progetto (*project management*) consigliano l’uso del *consuntivo di periodo* come strumento essenziale per misurare il grado di avanzamento e valutare le corrispondenti prospettive di completamento. Per l’efficace attuazione di tale pratica serve dotarsi di una interpretazione “intelligente” (non superficiale) del concetto di consuntivo di periodo. Questa prova d’esame vi chiede di: (1) fornire una definizione di “consuntivo di periodo” sia come nozione in sé che in termini dei dati che esso racchiude e della specifica funzione informativa di ciascuno di essi; (2) spiegare quali indicazioni possano essere tratte da un consuntivo siffatto; (3) suggerire una metodica utile ad alimentare la raccolta dati alla base di un consuntivo di periodo, rendendola veloce e affidabile.

# Risposta

1) Un consuntivo di periodo è un rapporto finanziario che riassume la performance e la posizione finanziaria di un progetto in un periodo di tempo specifico, in genere un mese o un trimestre. Include dati su entrate, spese, profitti e flussi di cassa del progetto. I rendiconti periodici possono essere utilizzati per seguire i progressi di un progetto e misurarne le prestazioni finanziarie nel tempo.

Esistono diversi tipi di rendiconti di periodo, tra cui:

* Conto economico: Un conto economico, noto anche come conto economico, mostra le entrate e le uscite di un progetto in un determinato periodo di tempo e calcola il profitto o la perdita netta.
* Bilancio: Lo stato patrimoniale mostra la situazione finanziaria di un progetto in un determinato momento, includendo attività, passività e patrimonio netto.
* Rendiconto finanziario: Il rendiconto dei flussi di cassa mostra l'afflusso e il deflusso di denaro per un progetto in un determinato periodo di tempo, compreso il denaro generato dalle operazioni, dalle attività di investimento e dalle attività di finanziamento.

2) Le informazioni che si possono ricavare dai rendiconti di periodo includono:

* Prestazioni economiche/di risorse: I rendiconti di periodo possono aiutare i project manager a monitorare le prestazioni finanziarie di un progetto, compresi i ricavi, le spese e i profitti.
* Pianificazione temporale e lavorativa: come è stato impiegato il tempo di calendario per lo svolgimento delle attività utili, vedendo quali persone sono state utilizzate e come, vedendo come migliorare possibilmente queste statistiche

3) Una metodologia utile per raccogliere i dati alla base di un rendiconto di periodo comprende le seguenti fasi:

* Determinare il periodo di tempo per il rendiconto.
* Identificare le fonti di dati che verranno utilizzate per creare il rendiconto, come ad esempio i registri finanziari, le fatture e le ricevute.
* Raccogliere i dati dalle fonti identificate.
* Organizzate i dati nelle categorie appropriate per il rendiconto, come entrate, uscite, attività e passività.
* Utilizzare una checklist delle attività da controllare e riportare in modo aggiornato secondo uno strumento possibilmente automatizzabile costi, scadenze, responsabilità, etc.
* Esaminare il rendiconto periodico per verificarne l'accuratezza e la completezza.
* Aggiornare regolarmente il rendiconto periodico per garantire che rifletta le informazioni più aggiornate.

# Domanda (punti 8/30)

Nel mondo SWE, vi è chi sostiene che il modello di sviluppo ispirato alla “*continuous integration*” sia in contrasto, e quindi incompatibile con la salvaguardia dei principi ispiratori del “modello a V”. Altri voci di quel mondo sostengono invece che i due modelli possano bene coesistere e contaminarsi a vicenda. Per rispondere a questo quesito: (1) fornite una definizione, concisa e ragionata di ciascuno dei due modelli, separatamente, ponendo l’accendo sui loro rispettivi obiettivi e le corrispondenti precondizioni di attuazione; (2) prendete posizione sul punto in questione, argomentando la vostra conclusione; (3) in caso affermiate compatibilità, illustrate un modello di sviluppo che ne combini i tratti; in caso contrario, suggerite la scelta a vostro avviso migliore (più conveniente, per rapporto costi/benefici) tra i due modelli.

# Risposta

1) Il modello di integrazione continua (CI) è una pratica di sviluppo del software che prevede l'integrazione regolare delle modifiche al codice in un repository condiviso e la costruzione e il collaudo automatico della base di codice risultante per identificare e risolvere i problemi nelle prime fasi del processo di sviluppo. Gli obiettivi principali della CI sono la riduzione del tempo e dello sforzo necessario per integrare le modifiche al codice, il miglioramento della qualità della base di codice e l'aumento della velocità di sviluppo. I prerequisiti per l'implementazione del CI includono la disponibilità di un repository condiviso, processi di compilazione e test automatizzati e un team che si impegna a integrare e testare regolarmente le modifiche al codice.

Il modello a V è un modello di sviluppo del software che prevede una sequenza lineare di attività, in cui ogni attività rappresenta una fase del processo di sviluppo. Il V Model è spesso utilizzato nel contesto dell'ingegneria dei sistemi e sottolinea l'importanza di verificare e validare il sistema in ogni fase dello sviluppo. Gli obiettivi principali del modello V sono garantire che il sistema soddisfi i requisiti specificati e identificare e correggere i difetti il più presto possibile nel processo di sviluppo. I prerequisiti per l'implementazione del modello V includono la presenza di un insieme chiaro di requisiti e di un processo di sviluppo strutturato che segue una sequenza lineare di attività.

2) Ritengo che il modello CI e il V model siano compatibili e possano coesistere e contaminarsi a vicenda. Mentre il V Model enfatizza l'importanza di verificare e validare il sistema in ogni fase dello sviluppo in modo bidirezionale, il modello CI sottolinea l'importanza di integrare e testare regolarmente le modifiche al codice per migliorare la qualità della base di codice. Combinando i principi di entrambi i modelli, è possibile creare un processo di sviluppo che combini i vantaggi di entrambi gli approcci.

3) Un modo per combinare le caratteristiche del modello CI e del modello V è quello di adottare un processo di sviluppo che segua una sequenza lineare di attività, simile al V Model, ma che incorpori l'integrazione e il test regolari in ogni fase, simile al modello CI. Ciò potrebbe comportare le seguenti fasi:

* Identificare e definire i requisiti del sistema.
* Sviluppare un piano per l'implementazione del sistema, che includa una tempistica e delle tappe fondamentali per ogni fase di sviluppo.
* Implementare il sistema secondo il piano, integrando e testando regolarmente le modifiche al codice che vengono apportate.
* Verificare e convalidare il sistema in ogni fase di sviluppo per assicurarsi che soddisfi i requisiti specificati.
* Identificare e correggere i difetti non appena vengono scoperti.
* Ripetere il ciclo fino al completamento del sistema.

Questo processo di sviluppo combina i vantaggi del modello CI e del modello V, assicurando che il sistema sia regolarmente testato e integrato e verificando e convalidando il sistema in ogni fase dello sviluppo per garantire che soddisfi i requisiti specificati.

# Domanda (punti 8/30)

Nel colloquio di assunzione presso l’organizzazione dei vostri sogni, il vostro interlocutore, che ha un curriculum professionale importante, vi chiede di fornire linee guida operative per l’utilizzo delle nozioni di baseline e di milestone nella pianificazione di progetto, spiegando anche se e come tali nozioni siano legate tra loro. Il vostro interlocutore desidera risposte chiare, nella loro formulazione e nei concetti che essa utilizza, e concrete, non limitate alle definizioni “da libro”, ma pronte all’uso in contesti reali, possibilmente anche personalmente sperimentate.

# Risposta

Le baseline e le milestone sono strumenti importanti per la pianificazione e la gestione dei progetti, in quanto forniscono un punto di riferimento per misurare i progressi e garantire che un progetto rimanga in linea.

Una baseline è un insieme di standard o criteri concordati che un progetto deve soddisfare. Può includere un'ampia gamma di elementi, come l'ambito, il calendario, il budget e la qualità. La linea di base viene stabilita all'inizio del progetto e serve come punto di riferimento per misurare i progressi.

Le milestone sono punti specifici nel tempo o eventi che segnano il completamento di una particolare fase o attività all'interno di un progetto. Servono come punti di controllo per misurare i progressi e possono essere utilizzati per identificare le aree in cui potrebbero essere necessari degli aggiustamenti per rimanere sulla strada giusta.

Le baseline e le milestone sono collegate tra loro, in quanto sono entrambe utilizzate per misurare i progressi e garantire che un progetto rimanga in linea con i tempi. La baseline serve come punto di riferimento per l'intero progetto, mentre le milestone segnano punti specifici di avanzamento all'interno del progetto. Infatti, normalmente, una baseline serve a sostanziare le milestone individuate.

Per utilizzare efficacemente le baseline e le milestone nella pianificazione del progetto, è importante:

* Definire chiaramente la linea di base del progetto all'inizio, includendo tutti i criteri rilevanti come l'ambito, il calendario, il budget e la qualità.
* Identificare le tappe fondamentali che segnano il completamento di fasi o compiti importanti del progetto.
* Misurare regolarmente i progressi rispetto alla linea di base e alle tappe fondamentali per garantire che il progetto sia in linea con i tempi.
* Apportare le modifiche necessarie per rimanere in linea con i tempi, come ad esempio l'adeguamento del calendario o del budget.
* Comunicare i progressi e gli eventuali aggiustamenti necessari alle parti interessate, tra cui il team di progetto, le parti interessate e la direzione.

# Domanda (punti 8/30)

Lo standard ISO/IEC 12207 specifica l’attività ***configuration control*** (parte del processo di configurazione) come segue:

“*This activity consists of the following (traced) tasks:  
- identification and recording of change requests;   
- analysis and evaluation of the changes;   
- approval or disapproval of the request;   
- (in case of approval) implementation, verification, and release of the modified software item.”*

Vogliamo comprendere se e come la concreta attuazione di tale attività all’interno di un progetto dipenda dal modello di sviluppo adottato. Più precisamente, vogliamo rispondere, in modo argomentato, a queste due domande:

1. Tale specifica, nella formulata di cui sopra, implica uno specifico modello di sviluppo? In caso affermativo, come potremmo rendere neutra la sua formulazione?
2. Come potremmo istanziare, adattandola, tale attività nelle nostre norme di progetto, in modo che essa si integri bene con uno sviluppo veramente incrementale?

# Risposta

1) Le specifiche dello standard ISO/IEC 12207 per il controllo della configurazione non implicano esplicitamente un modello di sviluppo specifico. Delinea un insieme di compiti relativi alla gestione delle modifiche agli elementi del software, indipendentemente dal modello di sviluppo utilizzato.

Per rendere la formulazione di questa specifica neutrale rispetto al modello di sviluppo, potrebbe essere modificata per includere un linguaggio più generale che non faccia riferimento a processi o tecniche specifiche. Per esempio, la specifica potrebbe essere modificata nel modo seguente: "Questa attività consiste nei seguenti compiti: identificazione e registrazione delle richieste di modifica; analisi e valutazione delle modifiche; approvazione o disapprovazione della richiesta; implementazione, verifica e rilascio dell'elemento software modificato, come appropriato per il modello di sviluppo scelto".

2) Per istanziare l'attività di controllo della configurazione nel contesto di uno sviluppo realmente incrementale, la si potrebbe adattare come segue:

* Identificare e registrare le richieste di modifica man mano che vengono identificate durante il processo di sviluppo.
* Analizzare e valutare le modifiche per determinarne l'impatto sull'intero progetto e sul programma di consegna incrementale.
* Ottenere l'approvazione o la disapprovazione della richiesta dalle parti interessate, come il team di progetto e la direzione.
* Se la richiesta è approvata, implementate la modifica nella consegna incrementale successiva e verificate che funzioni come previsto.
* Rilasciare l'elemento software modificato come parte della consegna incrementale successiva.

Questo adattamento dell'attività di controllo della configurazione tiene conto della natura incrementale del processo di sviluppo, incorporando l'implementazione, la verifica e il rilascio delle modifiche nel programma di consegna incrementale. Inoltre, consente la flessibilità necessaria per adattarsi alle modifiche che vengono identificate e approvate durante il processo di sviluppo.

# Domanda (punti 8/30)

Vi chiediamo di discutere quale rapporto vi sia tra il numero di versione assegnato a ogni singolo prodotto di progetto software e la storia delle modifiche (cioè operazioni di scrittura) effettuate sui contenuti del *repository* ove tale prodotto risieda. Per dare un fondamento alla vostra risposta, converrà che definiate preliminarmente: (1) cosa sia secondo voi un “prodotto” di progetto, e cosa lo caratterizzi rispetto a qualunque altro artefatto di sviluppo; (2) quanti “*repo*” sia desiderabile avere in circostanze analoghe al vostro progetto didattico, se uno o più, e a quali destinatari, sviluppatori o clienti/utenti, essi siano rivolti; (3) quando e perché debba avvenire un incremento nel numero di versione assegnato a qualsivoglia prodotto nell’accezione di cui sopra.

# Risposta

1) Un prodotto di progetto è un deliverable, quindi un oggetto che risulta dal processo di sviluppo ed è destinato all'uso da parte di stakeholder esterni, come clienti o utenti. Un prodotto di progetto è tipicamente un prodotto finito e pronto per l'uso, a differenza degli artefatti di sviluppo intermedi come il codice sorgente o gli artefatti di compilazione. Le caratteristiche di un prodotto di progetto possono includere la presenza di un'interfaccia utente, l'essere autonomo e l'essere testato e documentato.

2) In un progetto simile a un progetto didattico, è generalmente auspicabile avere almeno un repository per conservare il codice del progetto e altri artefatti di sviluppo. Questo repository dovrebbe essere accessibile al team di sviluppo e potrebbe anche essere accessibile a stakeholder esterni, come clienti o utenti, a seconda delle esigenze specifiche del progetto. In particolare, è desiderabile avere delle repo per ogni artefatto rilevante (es. una per la documentazione, una per il codice, etc.), adoperando vari branch/diramazioni dal ramo principale qualora necessario per implementare alcune particolari caratteristiche o idee di progetto.

3) Il numero di versione del prodotto di un progetto software deve essere aumentato quando ci sono cambiamenti o aggiornamenti significativi al prodotto. Ciò potrebbe includere l'aggiunta di nuove funzionalità, la correzione di bug significativi o altre modifiche che influiscono sulla funzionalità o sulla qualità complessiva del prodotto. Il numero di versione deve essere incrementato in modo coerente e logico, ad esempio utilizzando un sistema di versioning semantico.

La relazione tra il numero di versione di un prodotto di progetto e la storia delle modifiche apportate al repository in cui risiede il prodotto è che il numero di versione è tipicamente usato per indicare il livello di modifiche apportate al prodotto. Quando la storia del repository riflette le modifiche apportate al prodotto nel corso del tempo, il numero di versione viene incrementato per riflettere tali modifiche e aiutare gli stakeholder a identificare lo stato attuale del prodotto.

# Domanda (punti 8/30)

Riflettiamo sui concetti di efficienza ed efficacia, prima inquadrandone il significato astratto, e poi trasponendolo in situazioni concrete, di progetto, attraverso esempi – almeno uno per ciascun concetto – che illustrino: (1) come, in cosa, in che modo, si manifestano; (2) come si misurano, con quali tecniche, quali strumenti, e quando; (3) come si migliorano.

# Risposta

L'efficienza si riferisce alla capacità di utilizzare le risorse in modo efficace per raggiungere un risultato desiderato. Si tratta di ridurre al minimo l'uso delle risorse ottenendo lo stesso livello di output.

L'efficacia si riferisce alla capacità di raggiungere gli esiti o i risultati desiderati. Si tratta di raggiungere gli obiettivi desiderati di un progetto.

In una situazione di progetto, l'efficienza e l'efficacia possono manifestarsi in modi diversi. Ad esempio:

Efficienza:

* Un team di progetto che è in grado di completare le attività in modo tempestivo utilizzando meno risorse, come tempo e denaro, sta dimostrando efficienza.
* L'uso di strumenti di automazione per ridurre il tempo e l'impegno necessari per completare le attività è un esempio di efficienza.

L'efficienza può essere misurata utilizzando tecniche come l'analisi costi-benefici o confrontando le risorse utilizzate con i risultati ottenuti. Per monitorare e misurare l'efficienza si possono utilizzare strumenti come i software di gestione dei progetti.

L'efficienza può essere migliorata identificando ed eliminando le fasi o le attività non necessarie, utilizzando strumenti di automazione o ottimizzando l'uso delle risorse.

Efficacia:

* Un progetto che raggiunge gli obiettivi dichiarati dimostra efficacia.
* Un progetto che fornisce un prodotto o un servizio che soddisfa le esigenze degli utenti previsti è un esempio di efficacia.

L'efficacia può essere misurata valutando gli esiti o i risultati ottenuti rispetto agli obiettivi del progetto. Questo può essere fatto attraverso metodi come il feedback dei clienti, i test di usabilità o le metriche di performance.

L'efficacia può essere migliorata stabilendo obiettivi chiari, rivedendo e regolando regolarmente il piano di progetto e raccogliendo il feedback degli stakeholder per assicurarsi che il progetto soddisfi le loro esigenze.

Varie:

* *Presentare almeno tre qualità da perseguire nella progettazione software, indicando anche come esse possano essere verificate e misurate quantitativamente. Ove possibile, motivare la risposta tramite riferimenti espliciti e concreti, positivi o negativi, alla propria esperienza di progetto didattico*

1) Manutenibilità: Questa qualità si riferisce alla facilità con cui un sistema software può essere modificato o aggiornato nel tempo. È importante progettare i sistemi software in modo da renderli facilmente modificabili, in modo da poterli aggiornare e adattare al mutare dei requisiti aziendali. La manutenibilità può essere verificata e misurata attraverso tecniche come la revisione del codice, i test e la documentazione.

2) Scalabilità: Questa qualità si riferisce alla capacità di un sistema software di gestire un carico di lavoro o una base di utenti sempre più ampia senza che le prestazioni diminuiscano. È importante progettare i sistemi software tenendo conto della scalabilità, in modo che possano continuare a funzionare efficacemente anche quando la domanda di utilizzo cresce. La scalabilità può essere verificata e misurata attraverso test delle prestazioni, test di carico e monitoraggio dell'utilizzo delle risorse.

3) Sicurezza: Questa qualità si riferisce alle misure adottate per proteggere un sistema software da accessi non autorizzati o usi impropri. È importante progettare i sistemi software tenendo conto della sicurezza, poiché qualsiasi vulnerabilità può avere gravi conseguenze sia per l'utente che per l'azienda. La sicurezza può essere verificata e misurata attraverso test di penetrazione, revisioni del codice e l'uso delle migliori pratiche di sicurezza.

* *Discutere le differenze e le correlazioni tra le attività dei processi di versionamento e configurazione nel ciclo di vita dei sistemi software*

Il versioning è il processo di assegnazione di identificatori unici alle diverse versioni di un sistema software. In genere viene utilizzato per tenere traccia delle modifiche apportate al software nel corso del tempo e per consentire agli utenti di scegliere la versione del software che desiderano utilizzare. Può essere particolarmente utile per la correzione di bug e per mantenere la compatibilità con altri sistemi o dipendenze.

La configurazione è il processo di adattamento di un sistema software per soddisfare le esigenze specifiche di un particolare utente o ambiente. Ciò può comportare la regolazione delle impostazioni, l'installazione di funzionalità o componenti aggiuntivi o l'integrazione con altri sistemi. La configurazione viene generalmente eseguita per personalizzare il comportamento o la funzionalità del software in modo da soddisfare le esigenze specifiche dell'utente o dell'ambiente.

Esiste una chiara correlazione tra versioning e configurazione nel ciclo di vita dei sistemi software. Quando un sistema software viene aggiornato o modificato, alle modifiche viene in genere assegnato un nuovo numero di versione. Questo permette agli utenti di tenere traccia delle modifiche apportate e di scegliere la versione più adatta alle proprie esigenze. Allo stesso tempo, potrebbe essere necessario aggiornare la configurazione del software per riflettere le modifiche apportate, al fine di garantire che sia pienamente funzionale e soddisfi le esigenze dell'utente.

In generale, il versionamento e la configurazione sono entrambe attività importanti nel ciclo di vita dei sistemi software. Il versionamento aiuta a tenere traccia delle modifiche e a mantenere la compatibilità, mentre la configurazione aiuta a personalizzare il software per soddisfare le esigenze specifiche dell'utente. Entrambe le attività sono fondamentali per garantire che i sistemi software siano efficaci e affidabili nel tempo.

* *È da ritenersi che la buona (efficace ed efficiente) gestione di progetto, project management, si giovi di strumenti tecnologici di supporto; confutate tale affermazione con appropriate argomentazioni oppure provate ad illustrare i requisiti principali che ritenete debbano essere soddisfatti da soluzioni professionali. Qualora nel vostro progetto didattico siate rimasti soddisfatti di qualche particolare strumento di supporto rilevante al quesito, descrivetene succintamente le caratteristiche pertinenti più significative.*
* *Immaginando di dover redigere un glossario come quello del progetto, dare una definizione dei seguenti termini: progetto, processo, attività, fase. Ove possibile citare fonti autorevoli.*
* *Dare una definizione ben fondata del concetto di "architettura software". In relazione a tale concetto, dare una definizione ai termini "framework" e "design pattern" spiegando come questi si integrino fra loro E all'interno di una architettura*
* *Gran parte dei gruppi che intraprende il progetto didattico dichiara di aderire al modello di sviluppo incrementale, per poi deflettere significativamente da esso. Descrivere le caratteristiche salienti di quel modello di sviluppo e discutere in breve le principali difficoltà che si possono incontrare nell’adesione ad esso.*
* *Illustrare le principali differenze, per obiettivi e modalità di svolgimento, tra le tecniche di inspection e di walkthrough.*
* *Fornire una definizione di “requisito”, applicabile al dominio dell’Ingegneria del Software e descrivere, succintamente ma con precisione, il ciclo di vita all’interno i un progetto del tipo di quello didattico, rappresentandolo come un’apposita macchina a stati, specificando anche le attività poste sugli archi di transizione.*
* *Fornire una definizione del concetto di framework, specificamente collocato in relazione alla progettazione software, insieme ad una descrizione sintetica dei criteri applicati per selezione ed eventuale uso di qualcuno di essi, come sperimentato nel proprio progetto didattico.*
* *Presentare, per obiettivi, criteri di valorizzazione, possibilità di automazione, due metriche significative per la misurazione della qualità della progettazione software e del codice ( quindi almeno una metrica per ciascun ambito). Ove possibile, fare riferimento a esperienze specifiche e personali per valutare l'esito osservato dall'eventuale uso pratico delle metriche discusse.*
* *Facendo riferimento allo standard ISO/IEC 12207, discutere la differenza di obiettivi, attività coinvolte, strategie di conduzione e strumenti, tra i processi di verifica e validazione.*