**Domande compito 1° parte:**

1. Descrizione del progetto e analisi di fattibilità.
2. Organizzazione del progetto.
3. Tempistica del progetto.
4. Gestione del rischio.

**Domande compito 2° parte:**

***Progettazione software-Differenze, vantaggi e svantaggi tra architetture a oggetti distribuiti e architetture client/server.***

* Nell’architettura a oggetti distribuiti, non c’è distinzione tra server e client. Il sistema può essere considerato come un insieme di oggetti interagenti la cui posizione è irrilevante; non c’è distinzione tra fornitore e utente dei servizi.

Vengono comunque mantenute le importanti nozioni di servizio e interfaccia.

Viene adottato un paradigma ad oggetti.

Gli *oggetti* sono i componenti fondamentali del sistema, forniscono servizi e ricevono servizi da altri oggetti senza distinzione logica tra client e server.

Gli oggetti possono essere distribuiti su diversi computer di una rete e comunicare attraverso un middleware chiamato **ORB**(object request broker).

Vantaggi:

* + Permette ai progettisti di del sistema di ritardare le decisioni su dove e come i servizi dovrebbero essere collocati, poiché gli oggetti che forniscono servizi possono essere posti su ogni nodo della rete.
  + Permette di aggiungere nuove risorse quando sono richieste.
  + Il sistema è flessibile e scalabile.
  + Possibilità di riconfigurare il sistema dinamicamente con oggetti che migrano attraverso la rete.

Svantaggi:

* + Maggiore complessità rispetto ai sistemi client/server.
  + La sicurezza beneficia dell’incapsulamento dei dati, ma la frammentazione dei dati influisce negativamente.

Nell’architettura client/server l’applicazione viene modellata come un insieme di servizi forniti da server a un insieme di client.

I client devono conoscere i server disponibili ma non sanno nulla sull’esistenza di altri client.

Ciascun servizio è caratterizzato da un’interfaccia, definisce protocollo e formato dei messaggi scambiati.

Esistono architetture client/server a uno, a due, a tre o n livelli.

L’applicazione è strutturata in tre strati:

* + Strato di presentazione provvede alla rappresentazione dei dati e a tutte le interazioni con l’utente.
  + Strato di elaborazione applicativa implementa la logica dell’applicazione.
  + Strato di gestione dei dati esegue tutte le operazioni sul database.

Vantaggi:

* Distribuzione dei dati semplice.
* Essendo un’architettura distribuita, può sfruttare efficacemente sistemi collegati in rete con molti processori distribuiti.
* Fa un uso efficace dei sistemi di rete.
* Facile aggiungere nuovi server o aggiornare quelli esistenti.

Svantaggi:

* + Limita la flessibilità.
  + I client ricevono servizi dai server.
  + I client devono conoscere i servizi offerti da specifici server e devono sapere come contattarli.
  + Gestione ridondante di ogni server.
  + Nessun modello di dati condiviso: i sottosistemi utilizzano modelli di organizzazione dei dati differenti. Lo scambio di dati può essere inefficiente.

***Quali sono gli attributi di un buon software?***

Le norme ISO/IEC 9126 descrivono un modello di qualità del software; definiscono le caratteristiche che la determinano e propongono metriche per la misurazione.

Il modello definisce le caratteristiche di qualità cui sono associate delle sotto caratteristiche dette anche attributi.

* Funzionalità
  + Completezza
  + Accuratezza
  + Interoperabilità
  + Sicurezza
  + Aderenza alla funzionalità
* Affidabilità
  + Maturità
  + Tolleranza ai guasti
  + Recuperabilità
  + Aderenza all’affidabilità
* Usabilità
  + Comprensibilità
  + Apprendibilità
  + Operabilità
  + Attrattività
  + Aderenza all’usabilità
* Efficienza
  + Comportamento rispetto al tempo
  + Utilizzo delle risorse
  + Aderenza all’efficienza
* Manutenibilità
  + Analizzabilità
  + Modificabilità
  + Stabilità
  + Provabilità
  + Aderenza alla manutenibilità
* Portabilità
  + Adattabilità
  + Installabilità
  + Coesistenza
  + Sostituibilità
  + Aderenza alla portabilità
* Qualità in uso
  + Efficacia
  + Produttività
  + Sicurezza
  + Soddisfazione

Gli attributi interni del software, definiti in modo appropriato, sono un requisito per raggiungere il livello di qualità esterna del prodotto; le caratteristiche esterne, anch’esse definite in maniera opportuna, rappresentano un requisito per raggiungere la qualità d’uso desiderata.

***Dal piano di progetto al documento di specifica dei requisiti. Quali le differenze?***

Il piano di progetto è un documento versionato, redatto dal project manager per poter stimare realisticamente le risorse, i costi e i tempi necessari alla realizzazione del progetto.

Il piano di progetto si divide in due parti, pianificazione e consuntivazione, per rendere possibile la pianificazione delle attività future e il riscontro tangibile tra le attività effettuate e quelle previste. Il piano è un documento che varia nel tempo a seconda delle situazioni e dell’evolversi delle attività portate avanti dai membri. Una parte del documento, il diario delle modifiche, tiene traccia delle variazioni apportate al piano, registrando la data delle modifiche e la versione del piano. Il piano contiene la descrizione delle attività pianificate

* Definizione degli obiettivi
* Analisi dei rischi (piano gestione rischi, sintesi rischi individuati e strategie di prevenzione)
* Descrizione del modello di processo di sviluppo - Descrizione fasi (milestones) del processo
* Suddivisione in sottosistemi
* Attività di progetto (Work Breakdown Structure. Diagramma di Pert,
* Diagramma di Gantt)
* Stima dei costi
* Consuntivo attività

La documentazione inerente il controllo del progetto prevede almeno una tabella nella quale per ogni attività si evincano i partecipanti con i rispettivi monte ore e almeno una tabella nella quale per ogni membro del gruppo si evincano le attività a cui egli ha preso parte ed il totale numero di ore di lavoro.

Il piano di progetto viene dettagliato prima dell’inizio delle attività tecniche;

vengono dettagliati gli aspetti necessari per sviluppare e rilasciare la soluzione.

***Obiettivi di un sistema qualità?***

Fornire al cliente la garanzia della qualità di quanto acquisito attraverso la massima trasparenza dell’operatività e dei controlli nei processi produttivi.

Aumentare il livello di governabilità e di efficienza/efficacia dei processi produttivi.

Permeare nella cultura aziendale i concetti di responsabilità, soddisfacimento del cliente, e miglioramento del valore reso.

Raggiungere un livello crescente di uniformità dei comportamenti.

***Decomposizione del sistema, Stili di decomposizione – caratteristiche di accoppiamento e coesione.***

Non è pensabile progettare un sistema software come un unico modulo.

Per ridurre la complessità della soluzione, decomponiamo il sistema in parti più piccole chiamate sottosistemi.

Un sottosistema indipendente può essere sviluppato in maniera autonoma da un team di sviluppatori.

Un obiettivo importante della fase di System Design è definire la decomposizione del sistema in termini di sottosistemi:

* Definire l’elenco dei sottosistemi.
* Definire le interfacce.
* Definire le classi.

La qualità del design dipende da due fattori qualitativi fondamentali:

* **Accoppiamento.**
* **Coesione**.

L’obiettivo rimane l’indipendenza funzionale dei sottosistemi.

L’**accoppiamento** fra moduli esiste quando ci sono interdipendenze tra un modulo e l’altro.

L’accoppiamento è una misura su quanto un modulo è connesso, ha conoscenza, si basa su altri moduli.

Come linea guida, definiamo la responsabilità dei moduli in modo da ottenere un basso accoppiamento, migliorando la manutenibilità del software.

La **coesione** ci dice con quale criterio suddividere le parti.

La coesione è una misura di quanto siano relate e mirate le responsabilità di un modulo.

Alta coesione: ciascun’unità è responsabile di un singolo compito.

L’alta coesione è una proprietà desiderabile perché aiuta a comprendere meglio il ruolo di un modulo, aiuta a manutenerlo, permette di limitare e focalizzare i cambiamenti nel codice e aiuta il riutilizzo.

Esistono diversi tipi di coesione (dalla più alta alla più bassa):

1. Funzionale
2. Layer
3. Comunicazionale
4. Sequenziale
5. Procedurale
6. Temporale
7. Di utilità

Stile di scomposizione modulare:

* Scomposizione orientata agli oggetti
* Scomposizione orientata alle funzioni

*Scomposizione orientata agli oggetti*

Scompone il sistema in un insieme di oggetti comunicanti.

Si occupa delle classi di oggetti, dei loro attributi e delle loro operazioni.

Vantaggi:

* Gli oggetti sono accoppiati debolmente.
* Gli oggetti sono spesso rappresentazioni di entità del mondo reale.

Svantaggi:

* I cambiamenti dell’interfaccia oggetto possono causare problemi.
* Entità complesse possono essere difficili da rappresentare.

*Scomposizione orientata alle funzioni*

Le trasformazioni funzionali elaborano i propri input e producono output.

Vantaggi:

* Supporta riutilizzo delle trasformazioni.
* È intuitiva, poiché molte persone pensano in termini di input e output.
* Il sistema può evolversi aggiungendo nuove trasformazioni.
* Facilità di implementazione sia come sistema concorrente che sequenziale.

Svantaggi:

* Formato comune per il trasferimento dei dati
* I sistemi interattivi sono difficili da scrivere utilizzando il modello pipeling.
* Non adatti per interfacce grafiche utente.

**Altre domane compito:**

1. **RUP e UP**
2. **Client server nei sistemi inter e intra organizzativi**
3. **Differenza tra Corba e Soap**
4. **Sistemi distribuiti**
5. **Milestone e tempistiche**

***Rup***

Il Rational Unified Process (estensione dello Unified Process; versione commerciale dell’ UP), è un modello di processo software iterativo.

In RUP è presente:

* Un organizzazione del piano di progetto per fasi sequenziali.
* Indicazioni sulle attività da svolgere e sulla loro sequenza.
* Un insieme di ruoli predefiniti.
* Un insieme di documenti da produrre, con template ed esempi già realizzati.

I principi di RUP sono:

1. Affrontare i rischi fin dall’inizio, in modo sistematico.
2. Fornire risultati di valore ai propri clienti e ai propri utenti.
3. Produrre al più presto software funzionante, e farlo

evolvere in modo incrementale.

1. Predisporsi fin dall’inizio a gestire i cambiamenti.
2. Consolidare il prima possibile un’architettura eseguibile.
3. Costruire il sistema con un approccio che massimizzi la

manutenibilità e la riusabilità dei componenti.

1. Lavorare insieme, come un unico gruppo di lavoro.
2. La qualità è nel modo concreto di lavorare, non nella

compilazione di documenti a parte.

Il ciclo di vita di RUP è diviso in una serie di iterazioni.

Ogni iterazione è divisa in una serie di fasi:

* Avvio (Inception) – riduzione rischi su obiettivi - definizione dei

limiti, della fattibilità e della giustificazione economica dell’intervento.

* Approfondimento (Elaboration) – riduzione rischi architetturali -

approfondimento dei requisiti e produzione di un primo incremento del

sistema.

* Realizzazione (Construction) – riduzione rischi di non finire in tempo

utile - produzione incrementale del sistema (completamente testato e

integrato).

* Completamento (Transition) – riduzione rischi di rilasciare un prodotto non accettabile - rimozione ultimi errori, e produzione della versione rilasciabile del sistema.

***Calcolo distribuito inter-organizzativo***

* Per ragioni di protezione e di interoperabilità, il calcolo distribuito è stato implementato innanzitutto a livello organizzativo su una serie di server sui quali distribuisce il carico.
* Solitamente collocati nella stessa organizzazione, possono essere applicati standard e processi operativi interni.
* I modelli di calcolo distribuito, che permettono la distribuzione inter- organizzativa, invece che intra-organizzativa, in cui i nodi diversi si trovano in diverse organizzazioni.

Due approcci possibili

1. peer-to-peer che si basa sull'esecuzione del calcolo da parte di nodi di rete  individuali;
2. Orientato ai servizi che si basa sui servizi distribuiti piuttosto che sugli oggetti  distribuiti, e si affida a standard basati su XML per lo scambio di dati.

***Corba e Soap***

Gli standar CORBA sono una serie di standard per il middleware che supporta le architetture ad oggetti distribuiti: comprendono definizioni di modelli di oggetti, di mediatori di richieste e di servizi comuni.

L’architettura ad oggetti distribuiti richiede in middleware (ORB) per gestire la comunicazione tra oggetti distribuiti.

Gli oggetti possono essere implementati con linguaggi diversi, per questo il middleware deve lavorare affinché la comunicazione sia il più trasparente possibile.

SOAP definisce un organizzazione per lo scambio di dati strutturati per i webservice.

SOAP può operare su differenti protocolli di rete ed è basato sul meta-linguaggio XML.

Gli strati di SOAP sono:

* Soap rpc
* Soap envelop
* Soap encoding

Tutti definiti sopra XML framework.

**Tempistiche/Milestone**

Quando si pianifica un progetto è necessario stabilire una serie di milestone a cui corrisponde un output formale come un report da presentare al responsabile.

Le milestone dovrebbero rappresentare la fine di una fase separata e razionale del progetto.

Una consegna è un risultato del progetto, che viene consegnato al cliente alla fine di alcune fasi principali.

In genere le consegne sono delle milestone ma le milestone non sono necessariamente delle consegne.

La tempistica richiede la suddivisione del progetto in attività separate, e le stime di tempo richiesto lo completano.

Oltre al tempo bisogna stimare le risorse necessarie per portare a termine il compito.

Le tempistiche del progetto sono rappresentate con un insieme di grafici.

**Ciclo di vita software**

Il ciclo di vita del software inizia con l'analisi dei requisiti.

Lo scopo è:

* Specificare i servizi che saranno forniti dal sistema proposto.
* Identificare le condizioni (vincoli di tempo, sicurezza e così via) di tali servizi.
* Definire in che modo il mondo esterno interagirà con il sistema.
* Prevede un significativo input delle parti interessate.
* Studio di Fattibilità: nei casi cui l'utente ultimo sia un’entità quale, un’agenzia governativa o un'azienda.

In altri casi potrebbe occuparsi della creazione di software commerciale off- the-shelf (COTS, in vendita al pubblico) per il mercato di massa, magari in distribuzione presso punti vendita o scaricabile via Internet.

In questo scenario l’utente è l’entità definita con meno precisione, e l'analisi dei requisiti potrebbe partire da un'indagine di mercato condotta dallo sviluppatore stesso.

L'ingegneria dei requisiti fornisce il meccanismo appropriato per:

* Comprendere ciò che il cliente desidera.
* Analizzare i suoi bisogni.
* Valutare la fattibilità.
* Negoziare una soluzione ragionevole.
* Specificare la soluzione in modo non ambiguo.
* Validare le specifiche.
* Gestire i requisiti mentre questi vengono trasformati in un sistema funzionante.

Il processo d'ingegneria dei requisiti viene ottenuto tramite l'esecuzione di sette diverse funzioni:  avvio, raccolta, elaborazione, negoziazione, specifica, validazione e gestione.

Studio di Fattibilità: nei casi cui l'utente ultimo sia un’entità quale, un’agenzia governativa o un'azienda.

**Obiettivi studio di fattibilità.**

Per tutti i nuovi sistemi il processo d’ingegneria dei requisiti dovrebbe iniziare con uno studio di fattibilità.

Input:

* Insieme di requisiti aziendali preliminari.
* Descrizione generale del sistema e come si vuole che supporti i  processi aziendali.

Risultato dovrebbe essere:

* Un rapporto che indica se vale o no la pena di continuare con l'ingegneria dei requisiti e lo sviluppo del sistema.

Uno studio di fattibilità è un breve studio che mira a rispondere a una serie di domande (le 3 domande d’oro):

1. Il sistema contribuirà agli obiettivi generali dell'organizzazione?
2. Il sistema può essere implementato usando la tecnologia attuale,  secondo costi prefissati e con vincoli sulla tempistica?
3. Il sistema può essere integrato con altri sistemi che sono già  installati?

Sorgenti di informazione:

* Manager dei dipartimenti dove il sistema sarà usato.
* Ingegneri del software che hanno familiarità con il tipo di sistema proposto.
* Esperti della tecnologia.
* Utenti finali del sistema.

In genere, bisognerebbe provare a completare uno studio di fattibilità in due o tre settimane.  Una volta ottenute tali informazioni si scrive il rapporto dello studio di fattibilità, indicando se lo sviluppo del sistema debba continuare o no. Nel rapporto si possono proporre cambiamenti allo scopo, al budget o alla tempistica del sistema e suggerire altri requisiti di alto livello.

**(Studio di fattibilità in pratica) Obiettivi dello studio di fattibilità:**

* Analizzare le esigenze informative connesse allo sviluppo di un nuovo progetto.
* Individuare una o più soluzioni relative alle applicazioni ed alle soluzioni organizzative e tecnologiche.
* **per**
* Fornire gli elementi di valutazione necessari per prendere una decisione sulla realizzazione operativa del progetto stesso.
* Proporre la soluzione con valutazione dei costi, benefici e rischi connessi con la realizzazione.
* Esporre le conseguenze del mancato raggiungimento degli obiettivi progettuali.

Stima, pianificazione, controllo:

* Definire le operazioni che devono essere svolte.
* I rischi.
* Le risorse.
* I prodotti da realizzare

Valutazione preliminare dei costi e dei benefici per stabilire se si debba avviar lo sviluppo dell’applicazione:

* Alternative possibili.
* Scelte più ragionevoli.
* Risorse finanziarie e umane necessarie per l’attuazione del progetto.

l risultato è un documento che dovrebbe contenere le seguenti informazioni:

1. Definizione preliminare del problema.
2. Possibili scenari che illustrino eventuali diverse strategie di soluzione.
3. Costi, tempi e modalità di sviluppo per le diverse alternative.

La fattibilità può essere:

* Tecnica - Verifica se gli aspetti tecnici della proposta sono effettivamente realizzabili.
* Organizzativa - Verifica se la proposta è realizzabile nell’ambito della organizzazione esistente.
* Economica - Verifica se le risorse necessarie per la realizzazione del sistema (costi) sono giustificate dai ritorni prevedibili, espressi in termini di benefici.
* Temporale - Verifica se il sistema è realizzabile nei termini in cui continua ad essere utile alla organizzazione.
* Motivazionale - Verifica l’effettivo grado di accettabilità che gli utenti potranno esprimere rispetto al nuovo sistema, una volta realizzato.

Conclusioni di studi di fattibilità

* Il progetto non si può realizzare.
* Il progetto si può realizzare, ma basta un intervento organizzativo.
* Il progetto si può realizzare con questa soluzione, con questi costi, con questi rischi.
* Il progetto si può realizzare, con queste soluzioni comparate.

**Parti dello studio di fattibilità:**

* Parte prima - La situazione attuale.
* Parte seconda - Progetto di massima della soluzione.
* Parte terza - Analisi del rischio.
* Parte quarta - Analisi costi-benefici.
* Parte quinta - Il progetto proposto.

Parte prima - La situazione attuale:

* Consiste nell’analisi del sistema esistente sulla base delle interviste ai principali “attori” del processo di produzione e di trasformazione delle informazioni
* Produce una “fotografia” dei processi esistenti che deve poi essere “modellata” sulla base delle tecniche.

Parte seconda - Progetto di massima della soluzione:

* Definizione delle soluzioni alternative.
* Analisi di impatto aziendale.
* Definizione della qualità attesa dal progetto.

Parte terza - Analisi del rischio

La valutazione delle alternative.

Lo SF definisce in maniera univoca:

* i requisiti informativi e funzionali della soluzione
* i requisiti architetturali derivanti dalla visione tecnologica assunta e dalle necessità di integrazione con S.I. esistenti interni o esterni
* Sono possibili alternative nella definizione della soluzione essenzialmente in  termini di:
  + architettura dati e architettura funzionale.
  + architettura tecnologica.
  + make or buy.
  + riutilizzo esistente.

Tali alternative possono essere esaminate e valutate nello SF e la scelta preferenziale può costituire:

* elemento vincolante (diventa requisito della richiesta di fornitura).
* elemento di preferenza nella scelta della fornitura.
* Le offerte rappresentano comunque diverse modalità di realizzazione della soluzione scelta.

**Analisi dei requisiti:**

L‘analisi dei requisiti rappresenta l’analisi completa dei bisogni dell’utente e dominio del problema.

Coinvolgimento di committente e ingegneri del SW.

*Obiettivo:* descrivere le caratteristiche di qualità che l’applicazione deve soddisfare.

*Output:* documento di specifica dei requisiti. Standard IEEE 830. Costituisce un accordo scritto tra tutte le parti in causa; va inteso come guida allo sviluppo del software; è il mezzo per dirimere eventuali controversie che possono verificarsi nel seguito del processo sviluppo.

Cosa sono i requisiti?

I requisiti di un sistema sono la descrizione dei servizi forniti e dei suoi vincoli operativi.

In alcuni casi, un requisito è semplicemente una formulazione astratta e di  alto livello di un servizio che il sistema dovrebbe fornire, oppure un vincolo  del sistema.

Altre volte, può essere una definizione formale e dettagliata di una  funzione del sistema.

Un requisito svolge una duplice funzione:

1. Può essere la base per una gara fra potenziali sviluppatori: quindi deve essere aperto a interpretazioni diverse;
2. Può essere la base del contratto stesso: quindi deve essere definito in dettaglio;

**I requisiti utente** indicano i requisiti astratti di alto livello:

* Dovrebbero descrivere requisiti funzionali e non-funzionali in modo da essere comprensibili a utenti del sistema privi di dettagliate conoscenze tecniche.
* Non devono descrivere caratteristiche di progettazione, ma solo il comportamento esterno.
* Sono definiti usando il Linguaggio Naturale, tabelle e diagrammi comprensibili a tutti gli utenti.

**I requisiti di sistema** danno la descrizione dettagliata di quello che  il sistema dovrebbe fare.

Dovrebbero descrivere semplicemente il comportamento esterno e i vincoli operativi: non come il sistema dovrebbe essere progettato o implementato; in realtà, il livello di dettaglio richiesto per specificare in modo completo un sistema software complesso, rende impossibile escludere tutte le informazioni sulla progettazione.

I requisiti di sistema sono versioni espanse dei requisiti utente e sono utilizzati dagli ingegneri del software come base di partenza per la progettazione; aggiungono dettagli e spiegano come i requisiti utente dovrebbero essere forniti dal sistema, possono essere usati come parte del contratto per l'implementazione e dovrebbero quindi essere una specifica completa e coerente dell'intero sistema.

*Architettura:*

* Potrebbe essere necessario progettare un'architettura iniziale del sistema per aiutare a strutturare le specifiche dei requisiti.
* I requisiti di sistema sono organizzati a seconda dei diversi sotto- sistemi che lo compongono. Questa definizione dell'architettura è essenziale se si vogliono riutilizzare dei componenti software quando si implementa il sistema.

*Interazione:*

* In molti casi i sistemi devono interagire con altri sistemi esistenti.
* Questo vincola la progettazione e impone requisiti al nuovo sistema. Potrebbe essere necessario l'uso di una specifica architettura per soddisfare requisiti non funzionali (come la programmazione a N-versioni per raggiungere l'affidabilità).

*Certificazione:*

* Un ente certificatore esterno che deve attestare la sicurezza del sistema, potrebbe indicare l’uso di una determinata architettura che è già stata verificata.

**Requisiti Funzionali**

* Descrivono i servizi, o funzioni, offerti dal sistema (normalmente attivati da user-input): “Quando l’utente richiede la visualizzazione dell’estratto conto... allora il sistema deve ...”.
* Frasi che descrivono ciò che il sistema dovrà fare, come reagirà agli input e si comporterà in varie situazioni. In alcuni casi possono affermare esplicitamente cosa il sistema non dovrebbe fare.

**Requisiti Non-funzionali**

* Descrivono gli aspetti del sistema che non sono direttamente legati al comportamento (funzionalità) del sistema.
* Descrivono vincoli sui servizi offerti dal sistema, e sullo stesso processo di sviluppo. “La visualizzazione dell’estratto conto deve avvenire entro 4 secondi dalla sua richiesta”.
* Vincoli sui servizi offerti dal sistema (come vincoli sui tempi di risposta...) a cui sottostare durante lo sviluppo → Requisiti di qualità.
* Includono vincoli temporali, sul processo di sviluppo e sugli standard.
* I requisiti non funzionali solitamente si applicano al sistema completo, non a singole funzioni o servizi.
* Possono riferirsi (Definire/ limitare) a proprietà del sistema come:
  + Affidabilità.
  + Tempi di risposta.
  + Occupazione di spazio.
* Possono definire vincoli come:
  + Capacità dei dispositivi di I/O.
  + Rappresentazione dei dati utilizzata nelle interfacce del sistema.

**Requisiti di dominio**

* funzionali e non-funzionali - riflettono caratteristiche generali del dominio applicativo. “L’accesso alla cassa continua da parte dell’addetto bancario al rifornimento deve avvenire secondo le consuete procedure di sicurezza a doppia-chiave”.
* Requisiti che derivano dal dominio di applicazione del sistema e ne riflettono le caratteristiche e i limiti.

**Tipi di requisiti non-funzionali:**

**Requisiti del prodotto**

* Specificano il comportamento del prodotto:
  + Prestazionali (velocità di esecuzione, quantità di memoria richiesta)
  + Usabilità
  + Efficienza
  + Affidabilità (tasso di fallimento accettabile)
  + Portabilità

**Requisiti organizzativi**

* Derivano dalle politiche e procedure dell’organizzazione del cliente e dello sviluppatore (es. Standard di processo da usare, piattaforme, requisiti di consegna che specificano quando il prodotto e la relativa documentazione devono essere consegnati. etc.).

**Requisiti esterni**

* Derivano da fattori esterni al sistema e al suo processo di sviluppo
  + Interoperabilità (come il sistema deve interagire con i sistemi di altre organizzazioni).
  + Legislativi (affinché il sistema operi legalmente).
  + etici perché questo possa essere accettato dai suoi utenti e dal grande pubblico.

**Documento di Specifica dei Requisiti (DdR or SRS)**

***SRS: Software Requirements Specification***

* È una dichiarazione ufficiale di ciò che gli sviluppatori dovrebbero implementare (attenzione: il COSA ma non il COME).
* Deve includere sia i requisiti utente (RU), sia una specifica dettagliata dei requisiti di sistema (RS).
* Talora RU e RS sono contenuti nello stesso documento, talora in documenti separati.

**Fase di progettazione/organizzazione**

L’architettura Software stabilisce l’organizzazione globale di un sistema software, definendo:

* La divisione del software in sottosistemi.
* La decisione di come queste parti interagiranno.
* Le interfacce delle varie parti.

Definire l’architettura è il cuore della fase di progettazione: è fondamentale per ogni Ing. Software.

* L’architettura spesso vincolerà l’efficienza complessiva, la riusabilità, e la manutenibilità del sistema.
* La progettazione architetturale.
* è un processo di progettazione finalizzato ad individuare i sottosistemi che compongono il sistema da realizzare, e le loro interrelazioni di controllo e di comunicazione.

L’output di questo processo di progettazione sarà una descrizione dell’architettura software.

Un Progettazione deve esibire una struttura di architettura che:

* sia stata creata utilizzando stili architetturali e Progettazione pattern riconoscibili.
* sia costituita da componenti che esibiscono buone caratteristiche di
* progettazione.
* possa essere implementata in modo evolutivo facilitando implementazione e testing.
* essere modulare; il prodotto software deve cioè essere decomposto  logicamente in elementi o sottosistemi.
* contenere delle rappresentazioni distinte dei dati, dell’architettura,  delle interfacce e dei componenti.
* condurre a strutture dati che siano appropriate per gli oggetti da  implementare.
* condurre a componenti con caratteristiche funzionali indipendenti.
* condurre ad interfacce che riducano la complessità delle comunicazioni fra  i componenti e con l’ambiente esterno.
* essere ricavata mediante un metodo ripetibile, pilotato dalle informazioni ottenute dall’analisi dei requisiti.
* essere rappresentato utilizzando una notazione che comunichi in modo efficace il suo significato.

Progettare significa eseguire un processo di risoluzione di un problema il cui obiettivo è trovare e descrivere un modo:

* Per implementare i requisiti funzionali.
* Rispettando i vincoli imposti dai requisiti non funzionali inclusi i vincoli sul budget.
* In conformità con una serie di principi di buona qualità.

***La progettazione*** di un sistema software (o System Design) è l’insieme dei task svolti dall’ingegnere del software per trasformare il modello di analisi (dai requisiti) nel modello di design del sistema.

***Obiettivo*** ultimo è definire un nuovo documento, scritto in linguaggio tecnico/formale, da dare ai programmatori, per guidarli nell’implementazione del software descritto nel Documento dei Requisiti Software.

***Progettazione top-down design***

Si parte col progettare la struttura di altissimo livello del sistema. Es: definire l’architettura del software e il tipo di database che sarà impiegato.

Quindi si procede gradualmente con decisioni più dettagliate riguardanti aspetti di più basso livello, quali:

* Il formato di particolari dati.
* Gli algoritmi che saranno usati.

***Progettazione bottom-up design***

Si parte progettando i componenti basilari di basso livello e si decide poi come collegarli insieme per ottenere i componenti di più alto livello.

Si procede dunque per livelli di astrazione (layers of abstraction).

**Domande Mattone B**

***Qual è la differenza tra ingegneria del software e ingegneria dei sistemi?***

* L’ingegneria dei Sistemi ha come oggetto tutti gli aspetti dello sviluppo di sistemi informatici, inclusi quelli HW, SW e di processo.

L’ingegneria del Software è una parte di questo processo: riguarda lo sviluppo dell'infrastruttura software,il controllo,le applicazioni,le basi di dati nel sistema.

***Che cosa è il CASE (Computer-Aided-Software-Engineering)?***

* In [informatica](https://it.wikipedia.org/wiki/Informatica) gli strumenti C.A.S.E. sono quelli che supportano lo [sviluppo](https://it.wikipedia.org/wiki/Sviluppo_%28informatica%29) del [software](https://it.wikipedia.org/wiki/Software) attraverso [interfacce](https://it.wikipedia.org/wiki/Interfaccia_utente) grafiche ([shell](https://it.wikipedia.org/wiki/Shell_%28informatica%29)) e visuali e [librerie](https://it.wikipedia.org/wiki/Libreria_%28informatica%29) di funzionalità.

***Il progetto software open-source: il sistema di licenze.***

* Il software open-source è distribuito con una licenza che permette le 4 libertà fondamentali:
  + Libertà 0: Libertà di eseguire il programma per qualsiasi scopo.
  + Libertà 1: Libertà di studiare il programma e modificarlo.
  + Libertà 2: Libertà di copiare il programma in modo da aiutare il prossimo.
  + Libertà 3: Libertà di migliorare il programma e di distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio.

Sistema di licenze:

* + GNU GPL (GNU General Public License): impone che ogni prodotto software derivato - ovvero, che modifica o usa codice sotto GPL - venga a sua volta distribuito con la stessa licenza.
  + GNU LGPL (Lesser GPL): impone di ridistribuire sotto LGPL solo il sw. originariamente LGPL. Utile per librerie sw.
  + BSD (Berkeley Software Distribution): garantisce le 4 libertà del software ma prevede che un programma protetto da licenze BSD possa essere modificato e ridistribuito usando la stessa o qualunque altra licenza
  + MPL (Mozilla Pubblic License): il codice sorgente copiato o modificato sotto la licenza MPL deve rimanere MPL ma è possibile combinare codice MPL e codice proprietario, che resta tale.
  + Apache, MIT, CPL, EUPL (European Union Public License).

***Quali sono le principali sfide dell’ingegneria del software?***

* L'eterogeneità, la consegna e la fiducia.
* Eterogeneità: lo sviluppo di tecniche per la compilazione del software in grado di far fronte a piattaforme eterogenee e agli ambienti di esecuzione;
* Consegna: lo sviluppo di tecniche che portano a una più rapida consegna di software;
* Fidati: lo sviluppo di tecniche che dimostrino che il software può essere considerato attendibile dai suoi utenti.

***Le leggi dell’evoluzione del software: la legge della conservazione della familiarità.***

* A mano a mano che un PS si evolve, tutti gli attori associati (sviluppatori, personale di vendita, utenti) devono mantenere la padronanza del suo contenuto e del suo comportamento, per garantire un'evoluzione soddisfacente. Una crescita eccessiva riduce tale padronanza e induce un temporaneo rallentamento nel tasso di crescita o addirittura una decrescita. Pertanto la crescita incrementale media rimane al più costante durante l'evoluzione del sistema.

***Spazio fisico e sicurezza fisica.***

* La sicurezza fisica è il complesso di misure che prevengono o dissuadono gli attaccanti dall'accedere a un locale, a una risorsa o a informazioni e delle linee guida su come progettare strutture in grado di resistere ad atti ostili.

Lo spazio fisico ai tempi di internet, non è più legato strettamente alla fisicità; si aggiunge un spazio virtuale costituito da un insieme di calcolatori interconnessi. Un attacco dal mondo virtuale può colpire oggetti del mondo fisico.

***Che cosa è l’ingegneria?***

* L’ingegneria è una scienza applicata alla risoluzione di problematiche che concorrono alla soddisfazione dei bisogni umani.

L’ingegneria, applicando le norme tecniche, fornisce metodi, progetti e specifiche per la produzione (costruzione) di un bene fisico, di un prodotto o di un servizio, e più in generale per lo sviluppo e il controllo di un processo industriale.

L’ingegneria propone metodologie di sviluppo, che riassumono e formalizzano esperienze e conoscenze pregresse.

***Che cosa è un processo software?***

* Un insieme di attività il cui obiettivo è lo sviluppo o la modifica (o evoluzione) del prodotto software.

Attività generiche in tutti i processi software:

* Specifiche - ciò che il sistema deve fare e dei suoi vincoli di sviluppo.
* Sviluppo - produzione del sistema software.
* Validazione - verifica che il software è ciò che il cliente vuole.
* Evoluzione - cambiare il software in risposta alle mutevoli esigenze.