Sintesi dei punti “da ricordare” di Ingegneria del software (1300 diapositive in 15 pagine) – Silvio Moioli

- Definizioni di base
  - Ingegneria del software: l'approccio sistematico allo sviluppo, all'operatività, alla manutenzione ad al ritiro del software.
    - Consiste:
      - nella divisione in fasi del processo produttivo e controllo dei risultati intermedi tramite la valutazione di modelli;
      - nella cura degli aspetti gestionali della produzione del software (tempi e costi preventivati).
    - Nasce dalla necessità di gestire i sistemi informatici moderni, i quali spesso:
      - devono integrarsi con le infrastrutture preesistenti;
      - sono distribuiti, con componenti eterogenee ed autonome;
      - devono essere flessibili ed evolutivi;
      - devono funzionare bene, oltre a funzionare;
      - devono integrare componenti prefabbricati (COTS);
    - Fonti della materia:
      - IEEE SWEBOK
      - Standard ISO (9126 sulla qualità del software, 12207 sui processi legati al ciclo di vita del software, 9000 sui sistemi qualità, ecc.);
      - Letteratura, certificazioni professionali, corsi universitari, ecc.
  - Processo: un insieme di attività correlate (fasi) che trasformano ingressi in uscite;
  - Organizzazione: un insieme di processi che produce prodotti o servizi;
  - Pattern: schema, forma ricorrente, astrazione di casi concreti che incorpora gli aspetti ricorrenti essenziali di un insieme di casi con proprietà definite.
- Modellazione, valutazione e miglioramento dei processi, sistemi qualità
  - Modellazione dei processi: necessaria alla misurazione e all'ottimizzazione dei processi;
  - Valutazione (assessment) dei processi:
    - Standard internazionali: ISO SPICE e DoD CMM;
    - Attributi di processo: caratteristiche misurabili del processo
    - Indicatore di maturità di un processo: livello di capacità (valutato sulla base degli attributi)
      0. Incomplete (non eseguito o eseguito in modo frammentario);
      1. Performed (eseguito);
      2. Managed (pianificato e con evidenze documentali);
      3. Estabilished (basato su modalità standard, eventualmente personalizzate);
      4. Predictable (con misure dei risultati del processo);
      5. Optimizing (con una sistematica attività di miglioramento continuativo, necessita delle misurazioni oggetto della norma ISO 9004).

- Sistemi Qualità: sistemi di gestione (ossia di governo dei processi) che guidano e tengono sotto controllo un'organizzazione con riferimento alla qualità
  - Oggetto della normativa ISO 9000, 9001 e 9004 (ente italiano per la traduzione ufficiale: UNI)
  - Principio base: formalizzazione, razionalizzazione, monitoraggio e miglioramento continuativo dei processi dell'organizzazione
    - Motivazioni:
      - migliorare la qualità dei processi;
      - migliorare la qualità dei prodotti/servizi (assumendo che dipenda dalla qualità dei processi);
      - ingegnerizzare l'organizzazione;
      - attivare un processo di miglioramento continuo;
      - ottenere la certificazione;
  - Certificazione: un'organizzazione che realizza un sistema qualità può richiedere la certificazione rispetto alla norma a un organismo di certificazione accreditato dal SINCERT (composto da UNI e CEI) per uno specifico insieme di prodotti e servizi;
    - Motivazioni:
      - iscrizione all'albo;
      - possibilità di partecipare a gare;
      - referenza;

- Processi per lo sviluppo del software (cicli di vita)
  1. Non-modello: code and fix;
  2. Sequenziale: lo sviluppo (del software o del sistema) è suddiviso in fasi prestabilite in cui ogni fase ha bisogno dei documenti prodotti dalla precedente.
     - Fasi principali:
       1. Analisi dei requisiti: mira alla creazione di un documento che contiene informazioni riguardo allo scopo del progetto, alle funzionalità che dovrà avere e al contesto in cui verrà utilizzato;
       2. Progettazione: redazione dei documenti (architetturale e di dettaglio), che descrivono come sarà il sistema risultante e come deve essere costruito;
       3. Realizzazione/codifica: costruisce codice, configurazioni, documentazione, casi di test...
       4. Testing: acquisizione di una “sufficiente” fiducia che:
          - il sistema si comporti come specificato (verifica);
          - il sistema si comporti come desiderato dal committente (validazione).
       5. Messa in esercizio;
       6. Manutenzione;
       7. Ritiro.
     - Al termine di ogni fase vengono svolti controlli sui documenti prodotti, eventualmente vengono ripetute parzialmente le fasi precedenti per effettuare correzioni;
     - Vantaggi: molto utilizzato, obbligatorio in alcuni contratti (fasi controllate), si adatta bene ai casi in cui le caratteristiche del prodotto finito sono ben decidibili all'inizio del

processo;

- Svantaggi: troppo rigido nei casi in cui gli obiettivi siano variabili nel tempo o addirittura non completamente noti, nei casi in cui vi siano forti limitazioni temporali e nei casi in cui parti del sistema non siano costruite ad-hoc.

3. Sequenziale con analisi e progettazione interfogliati
   - Le fasi di analisi dei requisiti e progettazione sono divise in un numero di sotto-fasi eseguite alternatamente fino alla creazione dei documenti finali;
   - Vantaggi: si adatta ai casi in cui i processi di analisi e progettazione sono strettamente dipendenti l'uno dall'altro (separarli potrebbe non essere possibile o troppo costoso).
4. Incrementale: il sistema è costruito “a blocchi” da integrare in tempi successivi
   - Solo l'analisi dei requisiti e la progettazione architetturale sono completate all'inizio del processo;
   - Le altre fasi sono ripetute per il “nocciolo” del sistema e per ognuno dei “blocchi” non essenziali separatamente, il prodotto viene installato per integrazioni successive (prima il “nocciolo”, poi uno ad uno tutti gli altri “blocchi”);
   - Vantaggi: utile in caso di forti limitazioni temporali (scadenze che rendano impossibile la produzione dell'intero sistema in un unico ciclo di sviluppo).
5. Con prototipi
   - L'analisi dei requisiti è svolta tramite la costruzione di prototipi aiutino committenti e analisti a comprendere i requisiti. I prototipi hanno solo scopo dimostrativo, vengono utilizzati unicamente in questa fase;
   - Vantaggi: utile nel caso in cui l'utente non conosca esattamente le proprie esigenze oppure se una specifica completa sarebbe troppo costosa.
6. Con prototipi, iterativo
   - La realizzazione avviene per prototipi con funzionalità via via più complete che vengono costantemente testati e sottoposti al giudizio del committente;
   - E' necessario per quanto possibile minimizzare l'instabilità del processo;
   - Vantaggi: utile in casi in cui sia necessario uno stretto controllo del committente prima di arrivare al prodotto finito.
7. Per integrazione
   - Unisce alla realizzazione di software ad hoc realizzato con un qualunque processo di sviluppo l'integrazione con infrastrutture software e hardware preesistenti e/o con COTS;
   - Solamente le fasi di analisi dei requisiti e progettazione architetturale vengono svolte interamente dal produttore;
   - Vantaggi: permette di riutilizzare componenti già esistenti (o acquistati) anziché costruirli da zero.
8. Unified Process
   - Processo centrato sulla progettazione architetturale che prevede l'iterazione delle fasi di sviluppo (avviamento, elaborazione, realizzazione e transizione) con numerose verifiche al termine di ogni fase;
   - E' molto legato ai modelli, linguaggi, tecniche e strumenti della Rational (ora divisione di IBM) e va adattato ad ogni singolo caso.
9. Modelli agili

    - Processi che mirano alla minimizzazione della “burocratizzazione” del processo di sviluppo, rendendolo più adattabile alle esigenze in continuo cambiamento del committente. Si valorizzano a questo scopo la comunicazione nel team e con il committente, le competenze individuali degli sviluppatori ed il valore del codice rispetto a tutti gli altri documenti prodotti;
    - Vantaggi: permette di sviluppare sistemi anche in caso di continui cambiamenti dei requisiti o di requisiti non chiari fino alla realizzazione;
    - Svantaggi: richiede un team di sviluppo piccolo, collaborante e competente.

10. Meta-modello: a spirale
    - Prevede quattro fasi (definizione degli obiettivi, analisi dei rischi, sviluppo/validazione e pianificazione) da ripetere iterativamente;
    - Ognuna delle fasi ha obiettivi sia tecnici che gestionali.

- ISO 12207: “catalogo” di processi standard per lo sviluppo del software.

- Analisi dei requisiti (“perché si costruisce?”)
  - E' la branca dell'ingegneria del software che si occupa:
    - dell'identificazione degli scopi di un sistema da realizzare;
    - dei bisogni degli utenti che esso dovrà soddisfare;
    - del contesto in cui verrà utilizzato;
    - delle sue funzionalità;
    - delle caratteristiche non funzionali che dovrà avere;
    - dei vincoli a cui lo sviluppo è sottoposto;
  - E' spesso fonte di problemi poiché:
    - interessa più soggetti con esigenze e visioni diverse, in alcuni casi in conflitto e potenzialmente variabili nel tempo;
    - il committente potrebbe fornire indicazioni incomplete;
    - è oggettivamente difficile stabilire o esplicitare alcuni requisiti;
    - richiede la conoscenza del campo applicativo dove il sistema verrà utilizzato;
    - richiede anche attitudini umane e di comunicazione;
    - gli errori sono tanto più costosi quanto meno avanzata è la fase dello sviluppo del progetto in cui vengono commessi, quindi è sbagliare in questa fase è critico;
  - Si suddivide in:
    - identificazione dei requisiti del committente;
    - specificazione: formalizzazione di funzioni e caratteristiche del prodotto.
  - Documento di specifica: un modello del sistema da realizzare e del suo utilizzo
    - Comprende (in ordine di importanza e completezza):
      - una lista delle funzionalità che il sistema dovrà mettere a disposizione e le caratteristiche non funzionali;
      - l'identificazione dei profili di utente che ne faranno uso;
      - come il sistema si integrerà col resto del sistema informativo preesistente (il bordo);
      - il contesto in cui il sistema verrà utilizzato;

        - gli obiettivi del committente;
        - il problema che ha generato l'esigenza della creazione del sistema.
    - Può presentare i requisiti suddivisi in classi (priorità);
    - Può presentare i requisiti con stile operazionale (comportamento atteso del prodotto) o dichiarativo (proprietà attese del prodotto);
    - Tecniche:
        - Interviste con il committente;
        - Interviste registrate;
        - Questionari scritti;
        - Osservazione dei futuri utenti al lavoro;
        - Studio di documenti, procedure, sistemi preesistenti;
        - Incontri tra sviluppatori e committenti (modelli agili);
        - Pattern;
        - COTS e framework;
    - Linguaggi e modelli:
        1. Linguaggio naturale
            - Rischia di essere ambiguo e di difficile comprensione: da evitare;
        2. Linguaggio naturale strutturato
            - Meno ambiguo, più facilmente interpretabile da persone diverse;
        3. Linguaggi semiformali
            - Solitamente grafici, più precisi ed espressivi del linguaggio naturale anche se non completamente formalizzati da un punto di vista teorico;
            - Alcuni di essi, i modelli per l'analisi, sono modelli specificatamente creati per l'analisi di sistemi software o di loro parti;
            - Esempi: casi d'uso UML, Reti di Petri interpretate, diagrammi di flusso, diagrammi E/R, ecc.
        4. Linguaggi formali
            - Solitamente più difficili da comprendere dei precedenti ma rigorosamente controllabili tramite risultati teorici noti e con proprietà ben definite;
            - Esempi: specifiche algebriche, specifiche logiche, automi a stati finiti, Reti di Petri, ecc.
        5. Prototipi
            - Esempi o porzioni del sistema realizzato, evolutivi o “usa e getta”.
    - Validazione:
        - Coerenza interna del documento;
        - Verifica delle proprietà dei modelli;
        - Comprensibilità;
        - Critica dei prototipi;
        - Corrispondenza con i bisogni delle parti interessate;

        - Adeguatezza rispetto ai possibili cambiamenti futuri;
        - Verifica del livello di astrazione;
        - Verifica dell'identificazione delle parti stabili e meno stabili del sistema;
        - Verifica che l'analisi del dominio applicativo sia sufficiente;
- Progettazione (“cosa si costruisce?”, “come?”)
    - In ingegneria è l'adattamento intenzionale di mezzi per raggiungere un fine prefissato.
    - In ingegneria del software è la specifica della struttura e del funzionamento di un sistema informatico che soddisfa le specifiche ed è costruibile nelle risorse assegnate (tecnologie, risorse umane, tempi, costi).
    - Il documento di progettazione dettaglia:
        - Le associazioni tra componenti del sistema e funzioni da essi svolte;
        - l'interazione tra componenti;
        - l'ambiente di implementazione;
        - i dettagli esecutivi.
    - Criteri:
        1. Semplicità: mantenere il progetto il più semplice possibile;
        2. Riuso: spesso è conveniente riutilizzare componenti ed idee con proprietà note dello stesso progetto o di altri progetti;
        3. Astrazione: i componenti non devono essere descritti né in modo eccessivamente astratto né troppo dettagliato. Perché il progetto sia di facile comprensione è bene che ogni componente non sia descritto da più di 10 sotto-componenti;
        4. Coesione: è bene che gli aspetti di un sistema simili o correlati siano allocati nello stesso componente per favorire riuso e comprensibilità;
        5. Incapsulamento (information hiding): è bene che l'interfaccia fra ogni componente ed il resto del sistema sia nota e minimale, per minimizzare gli effetti collaterali in caso di modifiche; il componente dovrebbe nascondere il più possibile dentro di sé la sua logica di funzionamento;
        6. Disaccoppiamento: è bene che ogni componente dipenda dal minor numero possibile di altri componenti, per evitare effetti collaterali “a cascata” in caso di modifiche e favorire riuso e comprensibilità;
        7. Criticità: concentrare gli sforzi sui componenti più critici e problematici.
    - Processo di progettazione:
        1. Il progettista si immagina una prima soluzione possibile (“la forma” del sistema);
        2. Si aggiungono progressivamente proprietà di costruibilità al progetto (sintesi, “la struttura”)
            - Processi di sintesi
                1. Top-down (per decomposizione)
                    - Partendo da una visione molto astratta del sistema, si scende nei dettagli per passi successivi;
                2. Bottom-up (per composizione)
                    - Partendo da alcuni dettagli noti si astrae il sistema completo;
                3. Per aggiunte

        - Si progetta in prima battuta solo il nocciolo del sistema con le funzionalità di base, e a seguire vari altri moduli con funzionalità aggiuntive;
    4. Misto
        - Si utilizzano per porzioni diverse del sistema diversi processi.
3. Si aggiungono i dettagli (“le decorazioni”);
4. Il progetto viene verificato
    - Cosa verificare?
        - Specifiche funzionali: devono essere tutte implementabili;
        - Specifiche non funzionali: l'implementazione del progetto deve soddisfare il modello ISO 9126.
    - Metodi
        - Ispezioni;
        - Analisi (“ragionare” sul progetto: casi limite, scenari, alternative possibili, documentazione...);
        - Verifica dei modelli nel documento di progetto;
        - Prototipi/simulazioni.
    - Tecniche
        - Verifiche per livelli: ogni verifica può essere approfondita con vari gradi di dettaglio;
        - Verifiche per scenari: ogni caratteristica è valutata in base al comportamento del progetto in un possibile scenario d'uso, manutenzione, guasto, ecc.;
        - Ispezioni per checklist.

- Varianti al processo:
    1. Progettazione interfogliata con le specifiche;
    2. Progettazione per famiglie di prodotti;
    3. Progettazione per integrazione (di COTS, sistemi esistenti o componenti riutilizzati da altri progetti);
- Metodi:
    - Progettazione basata sui dati: il nucleo del progetto è la specifica della base di dati, componente fondamentale del sistema;
    - Progettazione basata sulle funzioni: il nucleo del progetto è la specifica di una lista di funzioni da implementare e la loro struttura gerarchica;
    - Progettazione orientata agli oggetti (OOD): il nucleo del progetto è la specifica di un insieme di classi e di modalità secondo cui cui i rispettivi oggetti interagiscono;
    - Progettazione orientata agli eventi: il sistema è pilotato da una serie di eventi asincroni che ne variano lo stato;
    - Progettazione orientata agli aspetti (AOD): suddivide il sistema in oggetti ed aspetti (che modellano i bisogni che logicamente apparterrebbero a molti oggetti, i cross-cutting concerns);
    - Progettazione basata su processi concorrenti: il nucleo del progetto è la specifica di un insieme di processi cooperanti;
    - Progettazione basata sulle interfacce utente: inizialmente si disegnano le interfacce

utente basandosi sui requisiti, si procede poi aggiungendo le funzionalità che emergono dalla loro analisi;

- Progettazione di sistemi real-time: alle metodologie precedenti si aggiungono dettagli riguardanti i vincoli temporali.
- Pattern: ad esempio Factory Method, Facade, Visitor, Iterator, Observable/observer, ecc.
- Progettazione basata sulle architetture
  - La progettazione può essere divisa in una prima fase più generale (architetturale, “cosa si produce?”) ed una seconda (di dettaglio, “come si produce?”);
  - La progettazione architetturale mira a scomporre il sistema in componenti distribuiti, autonomi e cooperanti attraverso meccanismi di comunicazione e sincronizzazione definiti, che possono essere sviluppati o acquistati in blocco;
  - Documentazione di un'architettura
    - In genere un'architettura è documentata tramite modelli che ne descrivono aspetti diversi, quali:
      - Vista statica: insieme di componenti del sistema e loro funzioni;
      - Vista dinamica: scambio dei messaggi fra i componenti;
      - Vista di implementazione software: allocazione dei componenti software a processi concorrenti e tecnologie di implementazione;
      - Vista di implementazione hardware: allocazione dei componenti software nell'hardware;
  - Stili (pattern architetturali):
    - Layer
      - Il sistema (o una sua parte) è modellato come un insieme di macchine virtuali strutturato su più livelli, in cui ogni livello offre servizi a quello superiore utilizzando solamente i servizi dei livelli inferiori;
      - Se esistono più componenti su uno stesso livello lo stile è detto segmented layer;
      - Se per accedere ai servizi di un livello basso A si usano quelli di un servizio a livello più alto B che fanno da tramite, B è detto bridge layer.
    - Pipes and filters
      - Il sistema (o una sua parte) è modellato come una catena di montaggio: i dati passano attraverso diversi componenti ognuno dei quali ha un solo input, un output e può funzionare anche autonomamente;
    - Repository
      - Il sistema (o una sua parte) è modellato come un componente che mantiene i dati e diversi altri componenti “client” fra loro indipendenti che sono in grado di leggere e scrivere da esso;
  - Stili di controllo (pattern architetturali del flusso di controllo):
    - Call-Return: un chiamante invoca una routine che gli ritorna il controllo al suo termine (la routine, a sua volta, può invocarne altre);
    - Manager: un componente decide l'avviamento e la terminazione dei processi;
    - Selective broadcast (Observer/Observable): un gestore invoca le routine degli altri processi ma a differenza di Manager, che lo fa arbitrariamente, segue una

logica di “prenotazione” sulla base di alcuni eventi che può osservare;

- Interrupt-driven: il flusso di controllo è pilotato da eventi: ognuno di essi genera un codice che è univocamente associato ad un processo da eseguire.

- Architetture distribuite (stili dei sistemi distribuiti)
  - Client-Server
  - Three-tiered layer structure: presentazione sul client, logica applicativa e gestione dei dati su due server dedicati;
  - Middleware: una piattaforma software mette a disposizione dei programmatori servizi complessi in aggiunta ai servizi di base del Sistema Operativo;
  - Architetture ad oggetti distribuiti: gli oggetti che compongono il sistema sono variamente dislocati in una rete di elaboratori e si scambiano messaggi tramite opportuni sottosistemi quali, ad esempio:
    - RPC/RMI
    - CORBA
    - MOM
    - DCOM

- Linguaggi: progetti ed architetture possono essere documentati tramite qualunque linguaggio o modello si ritenga opportuno
  - Esempi: diagrammi E/R, diagrammi UML, reti di Petri interpretate, automi a stati finiti, ecc.;
- Tracciabilità dei componenti
  - Ogni componente realizzato deve avere un corrispondente nel progetto e viceversa, è necessario tenere traccia di versioni e cambiamenti.

- La qualità del software
  - Cos'è? Non esistono definizioni precise o univoche, ma esiste una norma (ISO 9126) che delinea cosa si intenda per qualità del software:
    1. Funzionalità: adeguatezza, accuratezza interoperabilità, sicurezza, conformità;
    2. Affidabilità: maturità, tolleranza ai difetti, recuperabilità, conformità;
    3. Usabilità: comprensibilità; apprendibilità; operatività; piacevolezza; conformità;
    4. Efficienza: prestazioni temporali; utilizzo di risorse; conformità;
    5. Manutenibilità: analizzabilità; modificabilità; stabilità; testabilità; conformità;
    6. Portabilità: adattabilità; installabilità; coesistenza; sostituibilità; conformità;
  - Come si ottiene?
    - Qualità di processo
      - Assunzione: un processo di qualità probabilmente produce risultati di qualità;
    - Qualità di prodotto
      - Attività di controllo e retroazione durante e al termine del processo;
  - Processo di valutazione della qualità di prodotto
    1. Definizione del profilo di qualità (“cosa si vuole?”): per ogni componente del sistema e per ogni attributo di qualità si stabilisce un livello qualitativo (matrice di qualità). Il profilo deve tenere conto dello scopo del sistema, delle aspettative

degli utenti e degli aspetti di mercato;

2. Definizione del piano di qualità (“cosa si controlla?”, “Quanto?”): coerentemente con quanto deciso al punto precedente, si stende una strategia di applicazione per ognuna delle tecniche disponibili (ispezioni, metriche, testing) e per ognuna delle caratteristiche di qualità;
3. Definizione del piano di testing, delle ispezioni e delle metriche (“come si controlla?”): si stende l'elenco dei casi di test, le checklist e il piano per le metriche. Inoltre si specifica un insieme di regole di interpretazione dei risultati;
4. Specifica di ogni caso di test;
5. Esecuzione del piano;
6. Raccolta dei risultati;
7. Interpretazione dei risultati (applicazione di regole di interpretazione);
8. Spiegazione: ripartizione dei giudizi derivanti dall'interpretazione sulle possibili cause.

- Metodi di valutazione della qualità di prodotto
  - Testing: esecuzione del prodotto software
    - Premessa: è praticamente (in molti casi anche teoricamente) impossibile assicurarsi della correttezza di un programma, si ricorre al concetto di fiducia nel programma per un certo uso;
    - Definizione di caso di test: documento che descrive una prova del sistema. Composto da:
      1. Dati in ingresso;
      2. Modalità di esecuzione;
      3. Dati in uscita;
      4. Criterio di successo (Oracolo);
    - Processo di testing:
      1. Definizione della strategia di test: valutazione della criticità dei componenti del sistema, dei tempi e dei costi e conseguente redazione di un documento che spieghi dove gli sforzi sono concentrati e perchè;
      2. Redazione del piano di test: elenco strutturato dei casi di test;
      3. Specifica del test: creazione di ogni singolo caso di test;
      4. Esecuzione dei test;
      5. Rapporto di test: quali e quanti test sono passati? Quali sono le cause (errori)?
      6. Debugging: identificazione, comprensione e correzione del difetto che ha causato il malfunzionamento.
    - Processo di testing di accettazione/collaudo:
      1. Test/Debug: di ogni programmatore durante la scrittura del codice;
      2. α-test: a prodotto finito, fatto dal produttore;
      3. β-test: presso il committente in modo limitato;
      4. Collaudo: presso il committente in condizioni di esercizio, ma con l'affiancamento del produttore, l'esito positivo comporta l'accettazione del prodotto;

5. Esercizio.

- Tecniche
  - Test funzionale (black box, basato su scenari);
  - Test strutturale (white box, qualità interna);
  - Casi di test da modelli: esistono sistemi automatici per la verifica di programmi che usano modelli formali o semiformali;
  - Casi di test da grafi causa-effetto: se l'output dipende da un insieme di input ed è possibile costruire una rete combinatoria che modella il sistema, si può costruire una tavola della verità per tutte le combinazioni ingresso-uscita, determinare le celle di interesse e svolgere dei test su di esse;
  - Cicli (o catene) di test: sequenze di test ottimizzate per una specifica area funzionale;
  - Criterio delle classi di equivalenza: generalmente un programma si comporta in modo analogo per certi insiemi di possibili input (classi di equivalenza)
    - Si può limitare il numero di casi di test a uno per classe di equivalenza;
    - E' opportuno verificare i valori limite delle classi (che in molti casi sono programmati in modo apposito).
  - Test negativo: prova del sistema in caso di input fuori specifica;
  - Test per integrazione: prova del sistema per parti ed integrazioni successive
    - Bottom-up: con uso di skeleton (chiamanti “fantoccio”);
    - Top-down: con uso di stub (chiamati “fantoccio”);
    - Per interfacce.
  - Test di sistemi ad oggetti (unit testing);
  - Test di non regressione: test rieseguito dopo la modifica del software per assicurarsi di non aver introdotto nuovi errori;
  - Criteri di copertura
    - Copertura funzionale;
    - Copertura topologica
      - istruzioni;
      - decisioni;
      - condizioni;
      - cammini (con N-copertura dei cicli).
  - Test di sistemi ad oggetti
    - Test dei metodi
    - Test sui valori degli attributi
    - Test di gruppi di oggetti
- Aspetti organizzativi/gestionali
  - E' utile che il testing sia svolto da persone diverse dagli sviluppatori;
  - E' utile che esista un processo sistematico di analisi e raccolta dei malfunzionamenti;

- Ispezioni: verifiche statiche (manuali o automatiche) dei documenti prodotti
  - Tecniche
    - Review: interna al gruppo di sviluppo;
    - Audit: svolta da professionisti di terze parti;
    - Walkthrough: l'ispezione avviene sotto la guida di un coordinatore esperto del progetto dopo che il gruppo di revisione ha potuto analizzare la documentazione separatamente;
    - Inspection: l'ispezione avviene in modo completamente separato, al termine riunione congiunta;
    - Per checklist: l'ispezione viene svolta attraverso una lista strutturata di aspetti da esaminare;
  - Le ispezioni vengono normalmente verbalizzate e vi è sempre una discussione su come risolvere i difetti trovati;
- Misure/metriche: misure di attributi del software da cui si possono stimare i valori delle caratteristiche di qualità
  - Classificazione
    - Misure: risultato della misurazione di un attributo;
    - Metriche: insieme di regole per definire un attributo e stabilire una sua misura, qualitativa o quantitativa;
    - Interne: valutate sui documenti interni al processo di sviluppo;
    - Esterne: valutate sul prodotto in funzione nel suo ambiente di utilizzo;
  - Tecniche
    - Numero ciclomatico (di Mc Cabe): numero di decisioni binarie di un programma;
    - LOC;
    - Densità dei commenti;
    - Fan in/out;
    - Profondità dell'albero di ereditarietà;
    - Modelli di affidabilità con l'analisi dei malfunzionamenti;
    - Valutazioni comparate;
  - Processo di una stima di qualità
    1. Pianificare un insieme di misure in base alla matrice di qualità;
    2. Eseguire le misure durante il processo di sviluppo;
    3. Interpretare i singoli risultati;
    4. Integrare i risultati e stimare l'attributo di qualità in maniera globale.
  - Note
    - E' praticamente impossibile stabilire una relazione tra gli indicatori e una caratteristica di qualità di un prodotto software;
    - E' però possibile utilizzare metriche e misure per
      - Gestione dei rischi;

        - Monitoraggio;
        - Fini di accettazione;
        - Valutazioni quantitative (es. per reingegnerizzazione).
- La gestione (“come si stabiliscono e rispettano i vincoli di tempi e costi?”)
    - WBS: una scomposizione del progetto in moduli di cui è possibile stimare tempi e costi di produzione
        - Per attività
        - Per componenti
        - Mista
    - Studio di fattibilità: stimare tempi e costi
        - Mira ad analizzare i rischi principali e fare una stima di tempi, costi e prezzo finale del prodotto;
        - Necessita almeno di una prima versione dell'analisi dei requisiti e della progettazione architetturale;
        - Permette di valutare le alternative architetturali (es. make or buy);
        - Tecniche
            - Stima analitica da WBS;
            - Stima Delphi: almeno due persone (il gruppo di stima) stilano separatamente una WBS a testa, successivamente in una riunione congiunta si esaminano le stime e si cerca di raggiungere un consenso;
            - Stima COCOMO: basata su modelli matematici che stimano lo sforzo di programmazione, di difficile applicazione perché richiede la conoscenza del numero di righe di codice del prodotto finito in anticipo.
    - Il gruppo di progetto
        - Il capo progetto: è l'”amministratore delegato” del progetto: normalmente scrive lo studio di fattibilità e stima tempi e costi a cui è vincolato dopo l'accettazione del contratto. E' inoltre responsabile della qualità tecnica del prodotto, del personale a sua disposizione e solitamente è il referente del cliente;
        - E' bene che esistano diversi ruoli standardizzati per l'organizzazione (capo progetto, programmatore, analista, esperto di database, sistemista, ecc.) eventualmente ricoperti dalla stessa persona;
        - Le competenze delle persone sono fondamentali e non sostituibili nella scelta dei componenti del gruppo.
    - Processo di gestione
        1. Studio di fattibilità
        2. Avvio del progetto
            1. Creazione del gruppo di progetto
            2. Allocazione delle risorse
            3. Revisione del piano
            4. Riunione/i di avvio
        3. Controllo di avanzamento (tempi e costi, solitamente in momenti particolari detti milestone)
        4. Controllo finanziario

5. Terminazione del progetto
    1. Archiviazione documenti
    2. Valutazioni a consuntivo, possibili evoluzioni
    3. Deallocazione delle risorse

- Gestione delle versioni e delle configurazioni
    - Risponde alle necessità di:
        - Tener traccia di tutte le versioni di tutti i documenti di uno o più progetti durante lo sviluppo;
        - Tener traccia dello storico di tutte le versioni dei prodotti rilasciati;
        - Tener traccia delle modifiche apportate ad ogni documento, degli autori e delle date;
        - Tener traccia di eventuali versioni diverse dello stesso prodotto (es. famiglie di prodotti o prodotti multipiattaforma);
        - Tener traccia delle configurazioni (procedimenti per passare dal codice sorgente all'eseguibile);
        - Gestire i diritti di accesso ai documenti;
        - Gestire transazioni lunghe ed accessi concorrenti.
    - Eccettuati i casi più semplici, è implementato tramite un software apposito che mantiene un database dei documenti.
- Gestione delle richieste di cambiamento
    - Tiene traccia dei bug report e delle richieste dei clienti
- Gestione della conoscenza
    - La seconda risorsa più importante per un'organizzazione ICT (dopo le risorse umane), è la sua conoscenza accumulata;
    - E' necessario che il know-how non venga perso ed è auspicabile che venga condiviso il più possibile nell'organizzazione, che sia accessibile e fruibile;
    - Knowledge management: l'approccio integrato alla creazione, organizzazione, gestione accesso e utilizzo delle conoscenze di un'organizzazione.