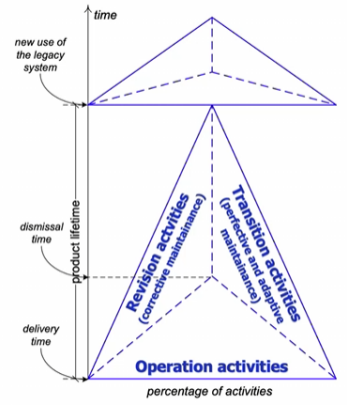
- ***Lezione 33***

  ➔ *Tredicesimo Blocco*

Passiamo ora all'argomento successivo direttamente correlato a quanto appena visto: la Qualità del Software.
Essa rappresenta per definizione il grado per cui il software possiede una serie di attributi "desiderabili".
La qualità del software può essere analizzata da vari punti di vista:

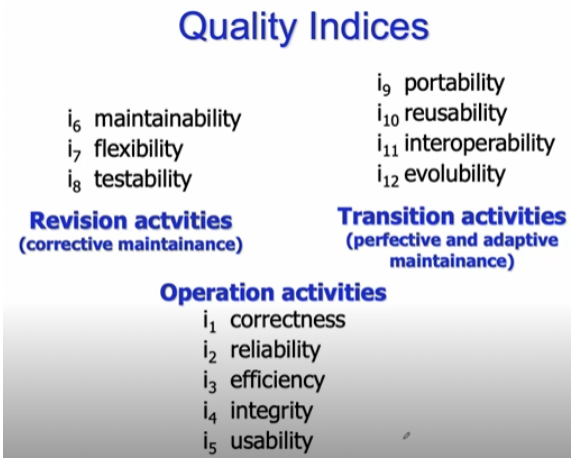
- Trascendentale: prodotto eccellente. Questo punto di vista l'abbiamo escluso fin dall'inizio, dal momento che l'obiettivo nello sviluppo software non deve essere tanto sviluppare un prodotto che sia eccellente quanto un prodotto che faccia correttamente quanto deve fare
- Utente: quanto il prodotto software contribuisca a raggiungere gli obiettivi dell'utente
- Prodotto: si valutano le caratteristiche del prodotto software (correttezza, affidabilità etc..) nonché la conformità con i requisiti, qui si fa riferimento alla qualità del software con la definizione introdotta poco fa dove la si vede come combinazione di queste caratteristiche "desiderabili", di qualità.
- Organizzazione: ci si focalizza sui benefici aziendali → costi e profitti (se il prodotto è facile da mettere sul mercato, quanto il prodotto permetta di incrementare i profitti/ridurre i costi etc…).

Vediamo ora come definire in modo quantitativo questa combinazione di attributi, nel tempo infatti per cercare di dare una valutazione oggettiva a questi attributi sono stati introdotti dei modelli di qualità che definiscono in modo preciso cosa si intende per qualità del software.
Importante introdurre questi modelli perché il concetto di qualità è un insieme di caratteristiche facilmente inquadrabili in modo soggettivo, ma si devono valutare in modo oggettivo per poter essere misurate e dare valutazione quantitativa.
In particolare ci soffermiamo sul Quality Model introdotto da McCall: il Quality Triangle.



Sulle ascisse la percentuale di attività che vengono svolte sul prodotto a partire dal momento in cui questo è immesso sul mercato. Sulle ordinate il tempo, che parte dal momento di

delivery (rilascio, momento in cui il software passa dallo stadio di sviluppo a quello di uso/manutenzione).
Dal momento del rilascio il software sarà usato da diversi attori in modi diversi: vi saranno gli utenti che svolgeranno le così dette Operation Activities. Mentre gli utenti usano il software gli sviluppatori avvieranno delle attività di monitoraggio e manutenzione pressoché continue per molti motivi (non solo correttivi!).
Infatti tra le attività di manutenzione le Revision Activities (manutenzione correttiva) e Transition Activities (manutenzione perfettiva e adattiva).
Ad un certo punto, quando gli utenti terminano di usare il software (Operation Activities), il software entra nello stadio di dismissione.
Come sappiamo questo stadio non è istantaneo e non rappresenta necessariamente la totale cestinazione del software, ma magari una sua rielaborazione (quindi ulteriori attività di sviluppo e manutenzione) per arrivare eventualmente a un nuovo utilizzo del sistema legacy (triangolo in alto).
Ciò che fa McCall, per ogni possibile attività di Revision, Transition o Operation, è definire degli attributi di qualità del software (detti nel caso di McCall Indici di Qualità).
Quindi qualità come combinazione di attributi desiderabili, nel caso di McCall come combinazione di 12 indici differenziati in base al tipo di attività.



Operation:

- i_1 Correttezza: rappresenta il grado con cui il prodotto soddisfa le specifiche e gli obiettivi degli utenti (quanto il software fa quello che l'utente vuole?)
- i_2 Affidabilità: grado con cui il software esegue le sue funzioni con la precisione richiesta (diverso dall'affidabilità classica che avevamo definito come la probabilità che il software funzioni correttamente in un certo intervallo di tempo)
- i_3 Efficienza: fa riferimento alle risorse di calcolo, quanto me serve potente il pc per eseguire il software
- i_4 Integrità: il software è sicuro? Si fa riferimento alla sicurezza del software in termini di accesso esterno indesiderato
- i_5 Usabilita: impegno richiesto per riuscire ad utilizzare il software da parte dell'utente

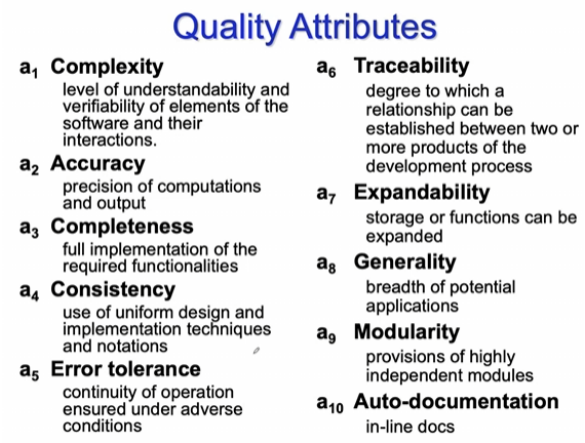
Revision:

- i_6 Manutenibilità: impegno necessario per individuare e correggere un difetto del programma
- i_7 Testabilità: impegno necessario per testare il prodotto e verificare che funzioni come desidero
- i_8 Flessibilità: impegno richiesto per modificare il prodotto (legato alla complessità del software stesso).

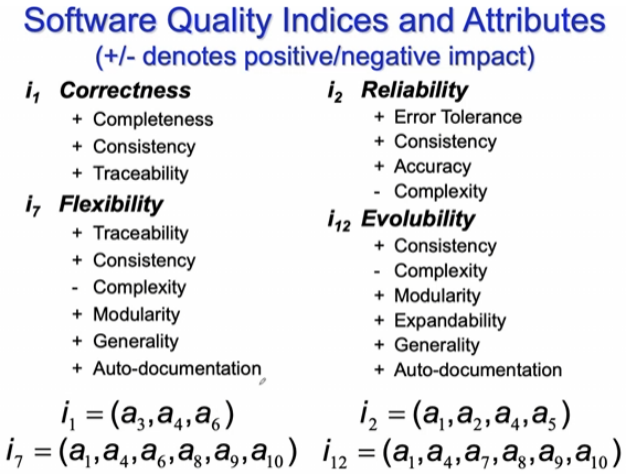
Transition:

- i_9 Portabilità: impegno richiesto per trasportare il prodotto da un ambiente operativo a un altro
- i_{10} Riusabilità: il grado con cui il prodotto o sue parti possono essere riusate in applicazioni differenti
- i_{11} Interoperabilità: quanto il software può essere accoppiato con altri prodotti
- i_{12} Evolubilità: effort richiesto per aggiornare il prodotto al fine di aggiornare eventuali nuovi requisiti (manutenzione perfettiva anche detta evolutiva)

Si ricorda che questi indici sono di alto livello, ciò che poi farà il modello di qualità è scomporli ulteriormente per arrivare a un livello tale da poter quantificare il valore dell'attributo. Per ora li stiamo definendo solo a livello astratto.
Il modello di qualità funziona come segue. Misuro questi 12 indici, che rappresentano le possibili caratteristiche che permetterebbero di definire oggettivamente la qualità del mio software. Per misurarli è necessario introdurre dei sottoindici, che mi permettano di arrivare ad un livello di dettaglio tale da poter quantificare nella pratica queste caratteristiche.
Secondo McCall in particolare la qualità del software è definita da un vettore contenente i 12 indici di qualità appena visti $q = (i_1, \dots , i_2)$.
Ciascun indice può essere misurato facendo riferimento ad un insieme di attributi di qualità (attento alla terminologia e quindi alla differenza tra Indice di Qualità e Attributo di Qualità per McCall). → $i_j = (a_1, a_2, \dots, a_n)$.
Un attributo può impattare anche più indici di qualità.
Gli attributi sono 10 :
Complessità come livello di comprensibilità, Accuratezza come precisione nel calcolo, Completezza come quanto il software ha implementato in modo completo le funzionalità richieste, Consistenza come utilizzo di approcci di design uniforme, Errror Tolerance come quanto il software riesce a "sopravvivere" ad eventuali malfunzionamenti e continuare a funzionare, Tracciabilità grado di relazione tra due o più prodotti nel processo di sviluppo (es. tra codice e documento requisiti etc..), Espandibilità come quanto il software può essere esteso in termini di storage e funzioni, Generalità come ampiezza di applicazioni potenziali, Modularità come grado di indipendenza tra moduli e Auto-documentation come quanto il software è in grado di aiutare l'utente attraverso un help in linea.



Quindi McCall definisce alcuni di questi attributi per ogni indice e misura tutti questi attributi secondo specifiche metriche. Vediamo un esempio per 4 indici



Si noti come gli attributi possono avere impatto positivo o negativo per gli indici. Ad es. per la flessibilità maggiore è la complessità, peggiore è l'indice di qualità!
In generale + se l'attributo influisce positivamente sull'indice, - altrimenti.
Per misurare gli attributi a loro volta sono necessari tipicamente dei sottoattributi per arrivare ad un'effettiva misura: es. per la modularità a_9 per misurarla si utilizzano tipicamente i sottoattributi come abbiamo visto nelle lezioni precedenti di coesione e coupling, morfologia e information flow (poi esistono anche altri modi, ogni modello usa il proprio e può far quindi riferimento a diverse metriche).
Per questi calcoli, che quindi possono avvenire usando diverse metriche, si introduce il Checklist Method, che mette a disposizione a chi deve valutare la qualità del software delle checklist che tipicamente pongono delle domande alle quali semplicemente l'utilizzatore deve porre una risposta.
Sarà a carico del metodo checklist, una volta fornite le risposte, calcolare il valore complessivo dell'attributo.
Una volta calcolati i valori dei vari attributi questi vengono raggruppati per ottenere il livello del corrispondente indice (tenendo ovviamente conto dell'impatto, se positivo o negativo).

- ***Lezione 34***

La valutazione delle checklist.
Una checklist rappresenta un insieme di domande, e ad ogni domanda sono associate quattro possibili risposte con un valore numerico. Alcune domande potranno essere identificate come *Non Applicabili* se non si vuole prendere in considerazione la domanda per il calcolo del valore di un attributo, *Non Valutabili* se invece si vuole scegliere lo score più basso possibile tra quelli elencati nelle quattro risposte alla domanda.
Colui che si prende carico di rispondere a queste domande è il Checklist Evaluation Team, composto da almeno quattro persone che svolgono ruoli differenti e sono competenti in termini di qualità del software.
Si raccomanda che il team sia costituito da: un Quality Assurance Specialist, un Project Leader, un System Engineer, un Software Analyst.
Nella prima parte del corso, quando abbiamo parlato di ingegneria dei requisiti e in particolare delle modalità di valutazione della documentazione, abbiamo detto che esistono due tecniche principali: Walkthrough (meno formale, il documento viene letto dai valutatori per rispondere in modo accurato alle domande della relativa checklist) e Ispezione (più formale).
In generale per ognuno dei 10 attributi definiti da McCall esiste una checklist corrispondente che permette di ottenerne il valore. Dopo averli valutati, secondo metodi specifici, è possibile risalire al valore dell'indice desiderato.
In particolare il metodo per calcolare il valore numerico di ciascun attributo (ECCETTO CHE PER LA MODULARITÀ) è basato sulla seguente formula:

$$V_{attribute} = \frac{\sum_{i=1}^{\#\,questions} V_{answer_i}}{\sum_{i=1}^{\#\,questions} max(V_{answer_i})}$$

(si ricorda infatti che a ciascuna risposta è associato un valore numerico).
Calcolati i valori per ogni attributo, si procede al calcolo del valore dell'indice.

$$V_{index} = \frac{\sum V_{attribute(+)} + \sum \left(1 - V_{attribute(-)}\right)}{number\ of\ attr.\ associated\ to\ index}$$

Si noti come si prende in considerazione l'impatto positivo o negativo degli attributi sull'indice in questione, in particolare se impatto negativo allora metto 1 - valore attr.
Chiaramente, essendo valori normalizzati, i valori degli attributi sono compresi tra 0 e 1 e lo stesso vale per i valori associati agli indici.
Una volta ottenuti i valori degli indici, vengono definiti i così detti "livelli di accettazione" degli indici di qualità (Acceptance Level Scale)

Acceptance Level Scale

- The output of the formula yielding V_{index} is a value between 0 and 1
- The following scheme can be used in order to identify threshold values for quality acceptance:

V_{index}	*Quality Level*
$0.66 < V \leq 1$	High
$0.33 < V \leq 0.66$	Medium
$0 < V \leq 0.33$	Low

Vediamo ora degli esempi di checklist per capire meglio il funzionamento di questa valutazione.
Immaginiamo di trovarci nella fase di progettazione architetturale (preliminare), dopo la quale abbiamo prodotto l'architettura del nostro sistema software.
Tra i 10 attributi consideriamo da calcolare a1 Complessità, a8 Generalità e a9 Modularità (per la quale vedremo un metodo di calcolo diverso da quello visto).
Iniziamo dalla Complessità, ecco la checklist:

General aspects

1) Are modules, procedures and structure names significant or conform to a standard, if any.
   - 0 : Yes, almost always
   - 1 : Quite often
   - 2 : Rarely
   - 3 : No
2) Is the formalism utilised to describe the system architecture only one, or multiple ones are present.
   - 0 : Unique formalism, standard and properly chosen
   - 1 : Few formalisms, standard and properly chosen
   - 2 : Few formalisms and someone out of standard
   - 3 : A lot of formalisms, someone out of standard
3) Is the global system design properly structured and easy understandable by people without specific knowledge about the system.
   - 0 : Yes
   - 1 : Almost positive
   - 2 : Not properly structured
   - 3 : Bad structured, and hardly understandable

Chiaro come complessità abbia impatto negativo sugli indici → caso migliore quello per cui si scelgono tutte risposte con 0 (si vuole ridurre la complessità, infatti nella successiva sommatoria per calcolare l'indice si dovrà fare il calcolo con 1 – valore!).
Queste erano domande generali, la checklist prosegue con domande più specifiche direttamente misurabili sull'architettura software nel nostro caso (metrics application).
Chiaramente le checklist cambiano in base anche al tipo di documentazione che si utilizza (ogni documentazione fornisce informazioni diverse, un conto è lavorare con il documento di specifica requisiti un conto con l'architettura software).

4) Is it possible to deduce the class of information of each single data present into the logical model. Are all the utilisation of this data declared in the documentation and realised with the purpose to read or write that class of information.
   - 0 : No
   - 1 : Rarely
   - 2 : Quite often
   - 3 : Approximately always.

Metrics applications

5) If is possible to perform measurements about the program complexity, based on the program calling graph, then use the metrics below defined.
   - 5.1) *Hierarchical* complexity: is the average number of modules per calling-tree level, that is the total number of modules divided by the number of tree levels
     - 0 : 0 up to 4
     - 1 : 4 up to 8
     - 2 : 8 up to 12
     - 3 : Less than 2 or greater than 12
   - 5.2) *Structural* complexity: is the average number of call for each module, that is the number of inter-module calls divided by the number modules.
     - 0 : Less than 2
     - 1 : 2 up to 4
     - 2 : 4 up to 8
     - 3 : Greater than 8

Per quanto riguarda invece l'attributo generalità questo impatta positivamente -> si vogliono valori più alti

Generality checklist

1) Are all functions offered by modules (and their interfaces) properly planned so that is possible to reuse them in some implementations that wasn't planned at the start time of the project. (i.e. : a specific percentage [18%], or any percentage of any value)
   - 0 : No
   - 1 : Rarely
   - 2 : Quite often
   - 3 : Always
2) Concerning modules that haven't enough generality, is it possible to make them more generic with low effort (near 10% of global effort to obtain each of them).
   - 0 : No
   - 1 : Rarely
   - 2 : Quite often
   - 3 : Always
3) Are all the data structure manipulations combined into modules specifically dedicated to this.
   - 0 : No
   - 1 : Rarely
   - 2 : Quite often
   - 3 : Always

Vediamo ora come si calcola invece l'attributo Modularità.
Sappiamo che un modo per misurarla riguarda l'utilizzo dei concetti e le misurazioni di Cohesion e Coupling, ma anche di Morfologia e Information Flow.
In questo caso si fa riferimento solo alla coesione e al coupling.
Alle risposte non sono associati valori numerici, ma delle etichette A, B C, o D.
Viene richiesto di fornire la percentuale di moduli in base al tipo (coesione coincidentale, logica o temporale, procedurale o comunicativa, informational o functional). Invece di separare i 7 li si raggruppa quindi in gruppi per un totale di 4 risposte.
Lo stesso vale per il coupling, dove viene richiesto di definire la distribuzione percentuale di coppie di moduli in base al tipo di coupling (content, common, control, stamp o data).

Cohesion

1) Which is the percentage distribution of system modules according the following four types:
   - 1A modules with coincidental cohesion
   - 1B modules with logical or temporal cohesion
   - 1C modules with procedural or communicational cohesion
   - 1D modules with informational or functional cohesion

Coupling

2) Which is the percentage distribution of communication modes (coupling) between modules according to the following four types:
   - 2A pairs of modules with content coupling
   - 2B pairs of modules with common coupling
   - 2C pairs of modules with control coupling
   - 2D pairs of modules with stamp or data coupling

Successivamente si passa sugli attributi generali di modularità e si torna quindi alle classiche checklist. Anche in questo caso, essendo modularità un attributo con impatto positivo sull'indice, le risposte migliori sono quelle tali per cui il valore è più alto:

General

3) Is the system decomposition organised in a hierarchical way
   - 0 : No
   - 1 : Rarely
   - 2 : Yes, enough
   - 3 : Yes

4) Which is the percentage of hierarchical structures (each program contains a comparable number of modules)
   - 0 : less then 70%
   - 1 : within 70% and 80%
   - 2 : within 80% and 90%
   - 3 : more than 90%

Vediamo ora come calcolare il valore complessivo di modularità.

$$V_{answer_1} = \frac{(0 \times \%1A) + (1 \times \%1B) + (2 \times \%1C) + (3 \times \%1D)}{50}$$

$$V_{answer_2} = \frac{(0 \times \%2A) + (1 \times \%2B) + (2 \times \%2C) + (3 \times \%2D)}{50}$$

Le prime due domande sono calcolate a parte come si vede sopra: in particolare si pesa zero la coesione e il coupling peggiore (coincidentale e content) mentre si pesa a 3 quelle migliori. Dividendo tutto per 50 si normalizza ottenendo quindi ancora un valore tra 0 e 1.

$$V_{Modularity} = \frac{V_{answer_1} + V_{answer_2} + V_{answer_3} + V_{answer4}}{\max(V_{answer_1}) + \max(V_{answer_2}) + \max(V_{answer_3}) + \max(V_{answer_4})}$$

Torniamo a vedere come si effettua la valutazione di questi attributi.
La documentazione come anticipato deve essere analizzata in dettaglio dal Checklist Evaluation Team per rispondere a ciascuna domanda, e per farlo come anticipato e già

spiegato nel primo modulo esistono tecniche come Walkthrough o Ispezioni più formali.
Ogni membro del team deve rispondere alle domande in modo indipendente senza confrontarsi subito con gli altri. Talvolta vi possono essere domande della checklist non applicabili al caso specifico che si sta valutando, in tal caso quelle domande si escludono dal calcolo definendole Not Applicable.
Quando invece la domanda risulta Applicabile ma Not Valutable (non valutabile magari in quanto la documentazione non risulta conforme per riuscire a rispondere alla domanda) → si assegna il punteggio più basso tra quelli disponibili.
Durante le riunioni di Walkthrough o i meeting di ispezione i membri del team finalmente si confrontano per verificare di aver dato le stesse risposte alle domande e il perché. Se sono diverse allora bisogna discuterne per arrivare a un'unica risposta comune, per poter finalmente calcolare il valore dell'attributo.
Si da ad ogni attributo un template contenente la lista di domande della checklist, se la domanda non è valutabile o non applicabile, e il punteggio. Si fa lo stesso per gli indici con la lista degli attributi, se hanno peso negativo o positivo, il punteggio di ogni attributo, il calcolo dell'index e il quality level (low, high o avg).

Template for attribute scoring

Attribute:			
Question #	N/A	Not Valuable	Score
			0–1
		Total Score (see formula on slide 12)	

Template for index evaluation

Index:		
Attribute	Weight (+/-)	Score
Evaluation (see formula on slide 13)		
Quality level (see acceptance scale on slide 14)		

L'utilizzo delle checklist (ognuna diversa per ogni attributo e in base al documento di specifica) e la valutazione di qualità fa parte di un'attività più generale molto importante nel progetto software: la Software Quality Assurance.
Questa fase del progetto rappresenta l'approccio sistematico per assicurare che sia il processo software (sviluppo) che il prodotto software in sé siano conformi agli standard, processi e procedure stabilite.
Gli obiettivi della fase SQA sono in particolare migliorare la qualità del software monitorando sia il software che il processo di sviluppo assicurandosi di star seguendo gli standard prestabiliti.
Fare SQA è un'attività molto costosa che richiede personale esperto, tempo effort… quindi l'attività incide sui costi del progetto.
Per introdurre un programma SQA è quindi necessario anzitutto che i top manager che gestiscono il progetto siano d'accordo e supportino SQA con budget adeguato.
A quel punto si dovrà identificare un piano di SQA stabilendo quali sono gli standard di riferimento, per poi realizzarlo al fine di valutare la qualità del software prodotto.
Il ruolo del team di SQA è garantire ai manager responsabili che quanto si sta facendo appunto è conforme agli standard e alle procedure.
Si garantisce l'uso di un'appropriata metodologia per lo sviluppo, che si proceda secondo

standard e procedure, che vengano realizzate opportune review per valutare quanto viene fatto e se viene fatto correttamente, che si producano documenti per supportare la manutenzione e il miglioramento del software, che si utilizzi un sistema di software configuration management per garantire la tracciabilità di quanto prodotto, che venga effettuato il testing e che questo venga superato correttamente, che si identifichino deviazioni ed eventuali problemi (se rilevati in modo tempestivo si possono affrontare in modo più semplice e meno costoso).
Gli obiettivi principali di SQA sono quindi ridurre rischi monitorando in modo appropriato il software e il processo di sviluppo, assicurandosi la conformità con standard e procedure e assicurandosi anche che le inadeguatezze e gli eventuali problemi del prodotto software, processo software o standard siano portati immediatamente all'attenzione del management affinché si risolvano velocemente.
È importante dire che SQA non è responsabile della produzione di software di qualità, ma di verificare che chi lavora al prodotto lo faccia correttamente seguendo gli standard e le procedure cercando di capire se vi sono problemi nel mentre (auditing, monitoraggio su chi produce il software per accorgersi se c'è qualcosa che non va).
Parlando di SQA abbiamo detto spesso i termini standard e procedure. Ma cosa rappresentano nel concreto?
Gli standard in questo contesto rappresentano delle linee guida alle quali il progetto software dovrebbe esser comparato per assicurarsi di procedere bene (gli standard definiscono QUELLO che in generale dovrebbe essere fatto).
Il minimo standard indispensabile da seguire include:

- Documentation Standard (es. che si utilizzi un template per il documento di specifica dei requisiti piuttosto che inventarsene la struttura di sana pianta)
- Design Standard (standard di progettazione, si vuole cioè trasformare i requisiti software in progettazione software utilizzando un'adeguata documentazione progettuale come linguaggi di modellazione, specifiche di definizione degli algoritmi…)
- Code Standard (standard di codifica per ottenere codice che sia quanto più possibile uniforme)

Quando si utilizza il termine "Procedure" invece si fa riferimento alle linee guida che mi suggeriscono COME procedere effettivamente nello sviluppo (nella specifica, progettazione etc…) anche suggerendo chi deve farla, quando e cosa fare con i risultati.
Tutta l'attività svolta dal team di SQA deve essere definita e pianificata con precisione affinché il management sappia quello che il team farà.
Tutto ciò fa parte dell'SQA Planning, che è esso stesso un documento che segue uno standard ben preciso. Uno di questi standard per l'SQA Plan è l'IEEE che specifica gli obiettivi, le attività da svolgere, gli standard e procedure da seguire etc…
Abbiamo quindi capito come il tema della qualità software sia molto importante e preveda costi dedicati anche talvolta significativi. Ciò permette un lavoro più agevole a chi realizza il prodotto essendo guidato da standard e procedure riconosciute.

→ Manca la fase di Testing