

Piano di Qualifica

# Informazioni sul documento

Versione 1.0

Redazione Mattia,maso, miotto, alessandro, andrea

Verifica boh

Responsabile boh

uso esterno

lista distribuzione Prof. Vardanega Tullio

Prof. Cardin Riccardo

CoffeeStrap

Descrizione

Questo documento ha lo scopo di presentare il piano di qualifica che il gruppo Aperture Software dovra’ seguire nel corso dello svolgimento del progetto “MaaP as an admin Platform”, presentato dal Proponente CoffeeStrap.

# 

# 

# 

# **Diario delle modifiche**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versione** | **Data** | **Autore** | **Descrizione modifica** |
| 1.2.0 | 2013-12-19 |  | Approvazione finale del documento |
| 1.1.1 | 2013-12-17 | secondo ver | Verifica del documento |
| 1.1.0 | 2013-12-16 | primo ver | Verifica del documento |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 1.0.0 | da guardare su gantt |  | Creazione struttura iniziale documento |

# 

# 

# Indice

Elenco tabelle

Elenco figure

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 1 Introduzione

## 1.1 Scopo del documento

Il piano di qualifica ha l’obiettivo di definire le strategie adottate dal gruppo Aperture Software per garantire la qualita’ del prodotto che verra’ sviluppato.

Il presente documento descrivera’ la qualita’ desiderate che il software dovra’ avere e i modi per raggiungerle. Verra’ inoltre definito cosa significa software “di alta qualita’”, in riferimento a questo specifico dominio, cosi’ da avere un ideale di riferimento per gli obiettivi del team.

## 1.2 Scopo del prodotto

Lo scopo del progetto è produrre un frameworkG per generare interfacce webG di amministrazione dei dati di business basati sullo stack Node.js e MongoDB.

L’obiettivo e’ quello di semplificare il lavoro allo sviluppatore che dovra’ rispondere in modo rapido e standard alla richieste degli esperti di business.

## 1.3 Glossario

Al fine di evitare ogni ambiguita’ nella comprensione del linguaggio utilizzato nel presente documento e, in generale, nella documentazione fornita dal gruppo Aperture Software, ogni termine tecnico, di difficile comprensione o di necessario approfondimento verra’ definito nel documento Glossario v1.2.0.

## 1.4 Riferimenti

### 1.4.1 Normativi

* **Norme di Progetto**: Norme di Progetto v.X.Y.Z.
* **Capitolato d’appalto C1:** MaaP as an admin Platform

<http://www.math.unipd.it/~tullio/IS-1/2013/Progetto/C1.pdf>

### 1.4.2 Informativi

* **Piano di Progetto:** Piano di Progetto v x.x.x
* Slides del corso di Ingegneria del software Modulo A, AA 2013/2014 del prof. Tullio Vardanega: <http://www.math.unipd.it/~tullio/IS-1/2013/>
* Wikipedia: [http://it.wikipedia.org](http://it.wikipedia.org/)
* Ian Sommerville, Software Engineering, 9 edizione (2011).
* **Glossario:** Glossario v x.x.x
* Standard ISO/IEC TR 15504 Software process assessment
* Standard ISO/IEC 9126:2001 Software engineering-product quality
* **Indice di Gulpease**
* **Complessita’ ciclomatica**

# 

# **2 Strategie di verifica**

**2.1 Qualita’ di processo**

Per garantire la qualita’ del prodotto finale e’ necessario migliorare la metodologia che porta alla qualita’ dei processi che compongono il prodotto. Per fare questo si e’ deciso di utilizzare lo standard ISO/IEC 15504 (vedi appendice) denominato SPICE (vedi appendice).

Per applicare il modello appena citato si deve utilizzare il ciclo di Deming (vedi appendice) che serve per il miglioramento continuo dei processi nel loro ciclo di vita.

## 2.2 Qualita’ di prodotto

Per cercare di realizzare e progettare un prodotto software, in accordo con specifiche e standard definiti, ed essere privo di non conformita’ o difetti, e’ necessario usare lo standard ISO/IEC 9126 (vedi appendice), il quale redige e descrive obiettivi qualitativi e specifica delle metriche per misurare il raggiungimento di tali obietttivi.

**2.3 Procedure di controllo di qualita’ di processo**

Per garantire la qualita’ dei processi si utilizza il principio PDCA (vedi appendice). Questo principio permette un continuo miglioramento della qualita’ di tutti i processi coinvolti nella realizzazione del prodotto finale.

Per controllare la qualita’ bisogna che i processi siano pianificati dettagliatamente, che le risorse siano individuate e ripartite in maniera quantificabile e ci deve essere un controllo sui processi.

La messa in opera di tali punti e’ descritta in maniera dettagliata nel Piano di Progetto v x.x.x.

Per rendere quantificabile la qualita’ dei processi si utilizzano le metriche descritte nella sezione x.x.x.

## La messa in opera di tali punti e’ descritta in maniera dettagliata nel Piano di Progetto v x.x.x.

## 

## 

## 

## 

## 2.4 Procedure di controllo di qualita’ di prodotto

Per garantire il controllo di qualita’ si utilizza:

* **Quality Assurance:** tradotta in “assicurazione di qualita’”, e’ la definizione di standard e processi che dovrebbero condurre a prodotti di alta qualita’ e all’introduzione di processi di qualita’ nei processi di stesura.

L’intenzione di un team di lavoro consiste nell’ottenere quella che si dice *correction by costruction*, ovvero “correttezza per costruzione”.

* **Verifica:** e’ la valutazione che un prodotto, servizio o sistema e’ conforme a regole, requisiti, specifiche o condizioni imposte. E’ spesso un processo interno e differisce dalla validazione

## Validazione: e’ l’assicurazione che un prodotto, servizio o sistema incontra nelle necessita’ che i clienti o gli stakeholders indentificano. Spesso comporta l’accettazione e l’idoneita’ con clienti esterni.

**2.5 Organizzazione**

Per ogni processo attuato si ci sono delle attivita’ di verifica e per ogni processo realizzato viene verificata la qualita’ del processo stesso e la qualita’ dell’ eventuale prodotto ottenuto da esso

Ogni fase descritta nel Piano di Progetto necessita di attivita’ diverifica:

* **Analisi**

si seguono i metodi di verifica descritti nelle Norme di Progetto sui documenti prodotti e i processi attuati. La messa in opera di tali tecniche e’ descritta in appendice.

La stesura dei documenti e’ l’attivita’ principale e costante nello svolgimento del progetto, il processo di verifica viene diviso in due attivita’.

In ogni documento e’ presente un diario delle modifiche per mantenere una cronologia delle attivita’ svolte e di chi le ha svolte.

**2.6 Pianificazione strategica e temporale**

Avendo scadenze prefissate nel Piano di Progetto, dobbiamo garantire che le attivita’ di verifica di tutti i documenti e prodotti deve essere sistematica, disciplinata e quantificabile. Procedendo in questo maniera si correggono gli errori il prima possibile.

La metodologia da seguire e’ descritta nelle Norme di Progetto.

**2.7 Responsabilita’**

Per garantire che il processo di verifica sia disciplinato, sistematico e quantificabile, bisogna attribuire responsabilita’ a specifici ruoli di progetto. I ruoli sono Responsabile di Progetto e Verificatore. I compiti di ciascun ruolo sono descritti nelle Norme di Progetto.

**2.8 Risorse**

Per raggiungere gli obiettivi di qualita’ prefissati sono necessari risorse umane e tecnologiche, suddivise rispettivamente in strumenti software e hardware utilizzati dai componenti del gruppo per effettuare verifica su processi e prodotti. I ruoli maggiormente coinvolti nella responsabilita’ delle attivita’ di verifica e validazione sono il Responsabile di Progetto e il Verificatore, e i rispettivi compiti sono descritti dettagliatamente nelle Norme di Progetto

# 4 Analisi

## 4.1 Analisi statica

Questa tipologia di analisi puo’ essere applicata sia al codice che alla documentazione, dato che prevede l’utilizzo di tecniche generali per ogni tipo di prodotto del team.

### 4.1.1 Walkthrough

Questa tecnica utilizza una scansione ampia e non mirata dell’oggetto in verifica, data la mancanza di esperienza e best practice del verificatore.

L’attuazione di questa tecnica di analisi e’ quindi molto onerosa, per questo sara’ nostro obiettivo renderla piu’ parallelizzabile possibile, cosi’ da ridurre i costi di verifica.

Dopo ogni attivita’ di verifica tramite Walkthrough, sperabilmente avremo trovato la maggior parte degli errori, fornendoci una visione delle erroneita’ commesse, di conseguenza potremo raffinare l’analisi e avvicinarci alla metodologia di Inspection.

### 4.1.2 Inspection

Questa tecnica e’ un’evoluzione del walkthrough e applica una ricerca piu’ mirata e specifica. E’ possibile utilizzare questa tecnica dopo aver acquisito dimestichezza con l’attivita’ di verifica, quindi non sara’ possibile utilizzarla fin da subito.

E’ obiettivo di una fase di Inspection la ricerca mirata di errori, aumentando l’efficienza della verifica e riducendo i costi in termini di tempo e risorse.

## 

## 

## 

## 

## 4.2 Analisi dinamica

Si applica solo al prodotto software e viene svolta durante l’esecuzione del codice. Per approfondimenti vedi appendice.

Si possono definire 5 tipi diversi di test:

* Test di unita’: vedi appendice per spiegazione, per applicazione vedi norme di progetto
* Test di integrazione: vedi appendice per spiegazione, per applicazione vedi norme di progetto: vedi appendice per spiegazione, per applicazione vedi norme di progetto
* Test di sistema: vedi appendice per spiegazione, per applicazione vedi norme di progetto
* Test di regressione: vedi appendice per spiegazione, per applicazione vedi norme di progetto
* Testi di accettazione: vedi appendice per spiegazione, per applicazione vedi norme di progetto

# 5 Metrica

## 5.1 Metrica documenti

### 5.1.1 Indice Gulpease

Obiettivo ottimale: [50-100]

Obiettivo sufficiente: [40-100]

5.1.4 Indice di leggibilita’ per varieta’ testuali(??)

## 5.2 Metrica software

Per la qualita’ del software si cerchera’ di utilizzare le seguenti metriche:

5.2.1-Complessita’ ciclomatica: vedi appendice per spiegazione,

5.2.2-Numero di livelli di annidamento: vedi appendice per spiegazione,

5.2.3-Attributi per classe: vedi appendice per spiegazione,

5.2.4-Numero di parametri per metodo: vedi appendice per spiegazione,

5.2.5-Accoppiamento: vedi appendice per spiegazione,

5.2.6-Copertura del codice: vedi appendice per spiegazione,

5.2.7-Lunghezza del codice: vedi appendice per spiegazione,

## 5.3 Metrica processo

* 5.3.1 Schedule Variance: vedi appendice per spiegazione,per applicazione vedi norme di progetto
* 5.3.2 Budget Variance: vedi appendice per spiegazione,per applicazione vedi norme di progetto

# 

# 

# 

# **7 Resoconto attivita’ di verifica**

7.1 Processi

Di seguito vengono riportati i valori degli indici di SV e BV calcolati per i documenti durante la fase di Analisi

|  |  |  |
| --- | --- | --- |
| **Nome documento** | **SV** | **BV** |
| Glossario |  |  |
| Norme di Progetto |  |  |
| Studio di Fattibilita’ |  |  |
| Piano di Qualita’ |  |  |
| Piano di Progetto |  |  |
| Analisi dei Requisiti |  |  |

7.2 Documenti

Di seguito vengono riportati gli indici di Gulpease calcolati per i vari documenti durante la fase di Analisi. L’esito di ciascun documento sara’ superato o bocciato in base all’indice di Gulpease stabilito nel paragrafo ….

|  |  |  |
| --- | --- | --- |
| **Nome documento** | **Indice Gulpease** | **Esito** |
| Glossario |  |  |
| Norme di Progetto |  |  |
| Studio di fattibilita’ |  |  |
| Piano di Qualita’ |  |  |
| Piano di Progetto |  |  |
| Analisi dei Requisiti |  |  |

# **Appendice**

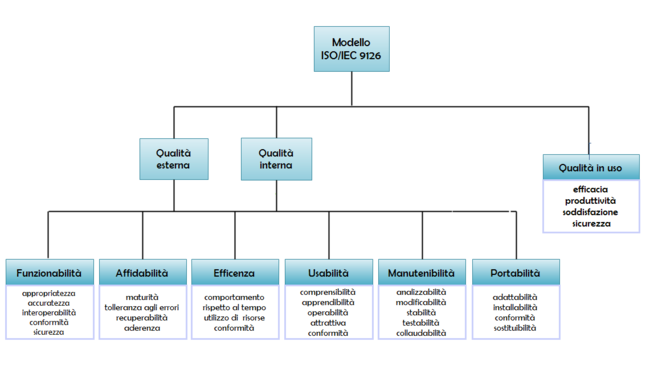
# A - Standard di Qualita’

## A.1 ISO/IEC 15504

ISO/IEC 15504, anche conosciuto come SPICE(Software Process Improvement and Capability Determination, ovvero miglioramento di processi software e determinazione di capacita’) e’ un insieme di documenti di standard tecnici per lo sviluppo software.

Questo documento viene utilizzato per il miglioramento dei processi in quanto stabilisce una struttura per la definizione BLA BLA BLA. CONTINUA QUI.

**A.2 ISO/IEC 9126**

Con la sigla ISO/IEC 9126 si individua una serie di normative e linee guida, sviluppate dall’ISO (Organizzazione internazionale per la normazione) in collaborazione con l'IEC (Commissione Elettrotecnica Internazionale), preposte a descrivere un modello di qualità del software. Il modello propone un approccio alla qualità in modo tale che le società di software possano migliorare l'organizzazione e i processi e, quindi come conseguenza concreta, la qualità del prodotto sviluppato.Il presente standard definisce 6 caratteristiche di qualita’ che ogni prodotto software deve perseguire, al fine di garantire la conformita’ agli standard con efficienza ed efficacia. Le caratteristiche delle qualita’ in uso esulano dal presente progetto didattico, in quanto non e’ prevista l’attivita’ di manutenzione conseguente al rilascio del prodotto. Quindi ci soffermiamo all’analisi della qualita’ interna ed esternza dello standard ISO/IEC 9126.

Le caratteristiche che un prodotto deve avere sono le seguenti:

* Funzionalita’: capacita’ di un prodotto software di fornire funzioni che soddisfano esigenze stabilite, necessarie per operare sotto condizioni specifiche.
  + Appropriatezza: rappresenta la capacita’ del prodotto software di fornire un appropriato insieme di funzioni per gli specificati compiti ed obiettivi prefissati all’utente;
  + Accuratezza: la capacita’ del prodotto software di fornire i risultati concordati o precisi effetti richiesti;
  + Interoperabilita’: la capacita’ del prodotto software di interagire ed operare con uno o piu’ sistemi specificati;
  + Conformita’: la capacita’ del prodotto software di aderire agli standard, convenzioni e regolamentazioni rilevanti al settore operativo a cui vengono applicati;
  + Sicurezza: la capacita’ del prodotto software di proteggere informazioni e i dati negando in ogni modo che persone o sistemi non autorizzati possano accedervi o modificarli, e che a persone o sistemi effettivamente autorizzati non sia negato l’accesso ad essi.
* Affidabilita’: capacità del prodotto software di mantenere uno specificato livello di prestazioni quando usato in date condizioni per un dato periodo.
  + Maturita’: capacità di un prodotto software di evitare che si verificano errori, malfunzionamenti o siano prodotti risultati non corretti;
  + Tolleranza degli errori: capacità di mantenere livelli predeterminati di prestazioni anche in presenza di malfunzionamenti o usi scorretti del prodotto;
  + Recuperabilita’: capacità di un prodotto di ripristinare il livello appropriato di prestazioni e di recupero delle informazioni rilevanti, in seguito a un malfunzionamento. A seguito di un errore, il software può risultare non accessibile per un determinato periodo di tempo, questo archo di tempo è valutato proprio dalla caratteristica di recuperabilità;
  + Aderenza: capacità di aderire a standard, regole e convenzioni inerenti all'affidabilità.
* Efficienza: capacità di fornire appropriate prestazioni relativamente alla quantità di risorse usate.
  + Comportamento rispetto al tempo: capacità di fornire adeguati tempi di risposta, elaborazione e velocità di attraversamento, sotto condizioni determinate;
  + Utilizzo delle risorse: capacità di utilizzo di quantità e tipo di risorse in maniera adeguata;
  + Conformita’: capacità di aderire a standard e specifiche sull'efficienza.
* Usabilita’: capacità del prodotto software di essere capito, appreso, usato e benaccetto dall'utente, quando usato sotto condizioni specificate.
  + Comprensibilita’: esprime la facilità di comprensione dei concetti del prodotto, mettendo in grado l'utente di comprendere se il software è appropriato;
  + Apprendibilita’: capacità di ridurre l’impegno richiesto agli utenti per imparare ad usare la sua applicazione;
  + Operabilita’: capacità di mettere in condizione gli utenti di farne uso per i propri scopi e controllarne l’uso;
  + Attrattiva: capacità del software di essere piacevole per l'utente che ne fa uso;
  + Conformita’: capacità del software di aderire a standard o convenzioni relativi all'usabilità.
* Manutenibilita’: capacità del software di essere modificato, includendo correzioni, miglioramenti o adattamenti.
  + Analizzabilita’: rappresenta la facilità con la quale è possibile analizzare il codice per localizzare un errore nello stesso;
  + Modificabilita’: capacità del prodotto software di permettere l'implementazione di una specificata modifica (sostituzioni componenti);
  + Stabilita’: capacità del software di evitare effetti inaspettati derivanti da modifiche errate;
  + Testabilita’: capacità di essere facilmente testato per validare le modifiche apportate al software.
* Portabilita’: capacità del software di essere trasportato da un ambiente di lavoro ad un altro. (Ambiente che può variare dall'hardwar[e](http://it.wikipedia.org/wiki/Hardware) al sistema operativ[o](http://it.wikipedia.org/wiki/Sistema_operativo)).
  + Adattabilita’: capacità del software di essere adattato per differenti ambienti operativi senza dover applicare modifiche diverse da quelle fornite per il software considerato;
  + Installabilita’: capacità del software di essere installato in uno specificato ambiente;
  + Conformita’: capacità del prodotto software di aderire a standard e convenzioni relative alla portabilità;
  + Sostituibilita’: capacità di essere utilizzato al posto di un altro software per svolgere gli stessi compiti nello stesso ambiente.

### A.2.1 Qualita’ esterne

Le **metriche esterne**, specificate nella norma **ISO/IEC 9126-2**, misurano i comportamenti del software sulla base dei test, dall'operatività e dall'osservazione durante la sua esecuzione, in funzione degli obiettivi stabiliti in un contesto tecnico rilevante o di business.

### A.2.2 Qualita’ interne

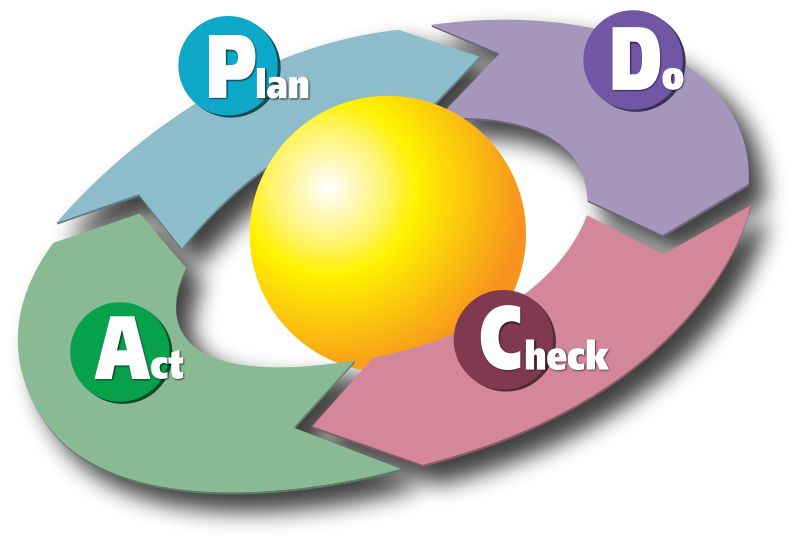
La **qualità interna**, più precisamente le metriche interne, è specificata nella norma **ISO/IEC 9126-3** e si applica al software non eseguibile (ad esempio il codice sorgente) durante le fasi di progettazione e codifica. Le misure effettuate permettono di prevedere il livello di qualità esterna ed in uso del prodotto finale, poiché gli attributi interni influiscono su quelli esterni e quelli in uso. Le metriche interne permettono di individuare eventuali problemi che potrebbero influire sulla qualità finale del prodotto prima che sia realizzato il software eseguibile. Esistono metriche che possono simulare il comportamento del prodotto finale tramite simulazioni.

## A.3 Ciclo di Deming (ciclo PDCA)

Il **ciclo di Deming** o **Deming Cycle** (**ciclo di PDCA** - plan–do–check–act) è un modello studiato per il miglioramento continuo della qualità in un'ottica a lungo raggio. Serve per promuovere una cultura della qualità che è tesa al miglioramento continuo dei processi e all'utilizzo ottimale delle risorse. Questo strumento parte dall'assunto che per il raggiungimento del massimo della qualità sia necessaria la costante interazione tra ricerca, progettazione, test, produzione e vendita. Per migliorare la qualità e soddisfare il cliente, le quattro fasi devono ruotare costantemente, tenendo come criterio principale la qualità.

La sequenza logica dei quattro punti ripetuti per un miglioramento continuo è la seguente:

* **P** - Plan. Pianificazione.
* **D** - Do. Esecuzione del programma, dapprima in contesti circoscritti.
* **C** - Check. Test e controllo, studio e raccolta dei risultati e dei riscontri.
* **A** - Act. Azione per rendere definitivo e/o migliorare il processo.



## 

# B - Descrizione

## B.1 Indice Gulpease

L'**Indice Gulpease** è un indice di leggibilità di un testo tarato sulla lingua italiana. Rispetto ad altri ha il vantaggio di utilizzare la lunghezza delle parole in lettere anziché in sillabe, semplificandone il calcolo automatico.

Definito nel 1988 nell'ambito delle ricerche del GULP (Gruppo Universitario Linguistico Pedagogico) presso il Seminario di Scienze dell'Educazione dell'Università degli studi di Roma "La Sapienza", si basa su rilevazioni raccolte tra il 1986 e il 1987 dalle cattedre di Filosofia del linguaggio e di [Ped](http://it.wikipedia.org/wiki/Pedagogia)a[gogia](http://it.wikipedia.org/wiki/Pedagogia) dell'Istituto di Filosofia.

L'indice di Gulpease considera due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere.

La formula per il suo calcolo è la seguente:



I risultati sono compresi tra 0 e 100, dove il valore "100" indica la leggibilità più alta e "0" la leggibilità più bassa. In generale risulta che testi con un indice

* inferiore a 80 sono difficili da leggere per chi ha la licenza elementare;
* inferiore a 60 sono difficili da leggere per chi ha la licenza media;
* inferiore a 40 sono difficili da leggere per chi ha un diploma superiore.

**B.3 Metriche processo**

**B.3.1 Schedule Variance**:Indica se si è in linea, in anticipo o in ritardo rispetto alla schedulazione delle attività di progetto pianificate nella baseline. E’ un indicatore di efficacia e se il suo valore e’ >0 allora il progetto sta avanzando con maggiore velocita’ rispetto a quanto pianificato. Viceversa se negativo.

**B.3.2 Budget Variance**: indica se alla data corrente si e’ speso di piu’ o di meno rispetto a quanto si era pianificato alla data corrente. Se tale valore e’ >0 allora il progetto sta consumando il proprio budget con minor velocita’ rispetto a quanto pianificato. Viceversa se negativo.

**B.4 Metriche software**

**Complessita’ ciclomatica:** E’ utilizzata per misurare la complessità di funzioni, metodi o classi di un programma. La complessità ciclomatica è calcolata utilizzando il grafo di controllo di flusso del programma: i nodi del grafo corrispondono a gruppi indivisibili di istruzioni, mentre gli

archi connettono due nodi se il secondo gruppo di istruzioni può essere eseguito immediatamente dopo il primo gruppo.

Alti valori di complessità ciclomatica implicano una ridotta manutenibilità del codice.

Valori bassi di complessità ciclomatica potrebbero però determinare scarsa efficienza dei

metodi (ad esempio: per ridurre il valore di tale metrica si eliminano blocchi condizionali

che garantirebbero scorciatoie in casi di esecuzione comuni).

Questo paramento inoltre è un indice del carico di lavoro richiesto per il testing.

**Numero di livelli di annidamento:** Rappresenta il numero di livelli di annidamento dei metodi, cioè l’inserimento di una struttura di controllo all’interno di un’altra.

Un valore elevato di tale indice implica un’alta complessità ed un basso livello di astrazione

del codice.

**Numero di attributi per classe:** Un numero elevato di attributi interni ad una classe potrebbe evidenziare la necessità di suddividere la classe in più classi da mettere in relazione tra di loro.

**Numero di parametri per metodo:** Un numero elevato di parametri per un metodo potrebbe evidenziare la necessità di ridurre le funzionalità associate a tale metodo.

**Accoppiamento:** L’accoppiamento tra due componenti indica che una modifica ad una delle componenti implica una modifica anche dell’altra componente. L’accoppiamento si divide in due parti:

Accoppiamento Afferente: Questo valore indica quanto il resto del software dipende da un package; alti valori di questo indice non sono collegati ad una errata progettazione, ma rappresenta soltanto la criticita’ del package in esame.

Accoppiamento Efferente: Questo valore indica quanto un package dipende dal resto del software, bisogna tenerlo basso ma senza complicare troppo la progettazione. I possibili valori per questo indice verranno scelti durante la fase di progettazione.

**Copertura del codice:** Questo valore indica la percentuale di istruzioni che sono eseguite durante i test. Piu’ la percentuale di istruzioni coperte dai test e’ grande, maggiore sara’ la probabilita’ che le componenti testate avranno una ridotta quantita’ di errori. La presenza di metodi semplici contribuisce ad abbassare il valore di tale indice.

**Lunghezza del codice:** Questa metrica misura la dimensione del codice di una componente. Piu’ grande e complessa e’ questa componente piu’ sara’ soggetta ad errori. La lunghezza del codice ha dimostrato di essere una delle metriche piu’ affidabili per predire errori di questo tipo.

11.2 Analisi dinamica

Le tecniche di analisi dinamica si applicano al prodotto software attraverso l’uso di test progettati prima della componente, o insieme di componenti, che essi verificano e della quale rileveranno possibili difetti di implementazione.

Una carattestistica fondamentale di questi test e’ la ripetibilita’ in quanto il riscontro di problemi e la correttezza del prodotto software possono essere garantiti solo se questo, dato un certo input, produce sempre lo stesso output su uno speciﬁco ambiente. Esistono vari livelli in cui si possono applicare queste tecniche; dal basso (singoli componenti) verso l'alto (intero sistema):

* Test di unita’: Il test di unita’ consiste nella verica di ogni singolo modulo all'interno di un'unica unita’, incluse le interazioni tra i moduli, eventualmente attraverso l'uso di stub e driver per sopperire alla mancanza di eventuali componenti collegati. La verica consiste nell'assicurare il soddisfacimento di tutti e soli i requisiti associati all'unita. I test di unita’ vengono progettati in fase di Progettazione di Dettaglio e Codifica.
* Test di integrazione: In seguito ai test sulle unita’, vengono eseguiti dei test che vericano la corretta integrazione tra le unita che fanno parte dello stesso componente e che sono gia’ state vericate attraverso i test di unita. In questo modo il fallimento del test indica un problema nell'interazione e non nelle singole unita. I test di integrazione vengono progettati in fase di Progettazione Architetturale.
* Test di sistema:Consiste nella validazione del prodotto software nel momento in cui vengono aggiunti tutti i componenti e lo si ritiene giunto ad una versione deﬁnitiva. Tale test deve veriﬁcare che la copertura dei requisiti software stabiliti in fase di analisi dei requisiti sia totale.
* Test di regressione: Ogni volta che viene apportata una modica al prodotto a seguito dell'individuazione e la correzione di un'anomalia, e necessario applicare un test di regressione per verificare che la modica apportata sia e  
  effettivamente correttiva e che non introduca nuove anomalie. Per effettuare questo test e’ necessario sottoporre le unita’ coinvolte ai rispettivi test di unita ed ai test di livello superiore che le coinvolgono fino al livello piu alto disponibile nella fase di sviluppo corrente.
* Test di accettazione: Test finale che consiste nel collaudo da parte del Committente. Se questo risulta positivo indica la fine della fase di sviluppo del software.