CLIPS

Piano di Qualifica v1.0.0

Sommario

Questo documento ha lo scopo di fissare le norme necessarie ad assicurare i requisiti qualitativi del *progetto*$_g$ CLIPS, regolamentando le operazioni di pianificazione e di verifica attuate per rispettare tali norme.

Nome del documento	Piano di Qualifica
Versione	1.0.0
Data di redazione	2016-04-06
Redazione	Andrea Grendene
Verifica	Tommaso Panozzo
Approvazione	Viviana Alessio
Uso	Esterno
Lista di distribuzione	prof. Tullio Vardanega prof. Riccardo Cardin Miriade SpA

Diario delle modifiche

Versione	Riepilogo	Autore	Ruolo	Data
1.0.5	Aggiornate le sezioni A e B	Andrea Grendene	Progettista	2016-05-03
1.0.4	Aggiunte le sezioni 3.8.1.2, 3.8.1.3, 3.8.3.7, 3.8.3.8 e l'Appendice B. Modificate la sezione 3.8.3.5 e la tabella della sezione 3.8.3.9	Andrea Grendene	Progettista	2016-04-28
1.0.3	Eliminata la sezione 3.7 Strumenti. Spostata la sezione 3.7.6 nella sezione 5.2. Modificati i riferimenti alla sezione 3.8 e alle relative sottosezioni.	Andrea Grendene	Progettista	2016-04-27
1.0.2	Modificate le sezioni 3.8.2.1, 3.8.2.2, 3.8.2.3, 3.8.2.4, 3.8.2.5 e aggiunte le sottosezioni 3.8.2.6 e 3.8.2.7. Stesa la prima parte della Sezione 5	Andrea Grendene	Progettista	2016-04-26
1.0.1	Piccole modifiche e correzioni grammaticali nelle sezioni 2.6, 3.1, 3.3, 3.7, 3.8.2, 3.9.2.1 e 4.1	Andrea Grendene	Progettista	2016-04-24
1.0.0	Approvazione documento	Viviana Alessio	Responsabile	2016-04-06
0.8.0	Terminata la verifica delle modifiche	Tommaso Panozzo	Verificatore	2016-04-05
0.7.1	Aggiunti i valori dell'indice $Gulpease_{\mathrm{g}}$ dei documenti e il loro esito	Andrea Grendene	Progettista	2016-04-05
0.7.0	Applicate le modifiche proposte dal Verificatore	Andrea Grendene	Progettista	2016-04-05
0.6.0	Terminata la verifica del documento	Tommaso Panozzo	Verificatore	2016-04-04
0.5.1	Terminata stesura della struttura del Resoconto dell'attività di verifica (senza i valori dell'indice $Gulpease_{\mathrm{g}}$ nella tabella)	Andrea Grendene	Progettista	2016-03-25
0.5.0	Terminata stesura della Gestione amministrativa della revisione	Andrea Grendene	Progettista	2016-03-25
0.4.0	Terminata stesura della Visione generale della strategia	Andrea Grendene	Progettista	2016-03-25

Versione	Riepilogo	Autore	Ruolo	Data
0.3.2	Terminata stesura della sezione 3.8	Andrea Grendene	Progettista	2016-03-24
0.3.1	Terminata stesura delle sezioni 3.6 e 3.7	Andrea Grendene	Progettista	2016-03-23
0.3.0	Terminata stesura delle sezioni da 3.1 a 3.5	Andrea Grendene	Progettista	2016-03-22
0.2.2	Terminata stesura della Definizione degli obiettivi di qualità	Andrea Grendene	Progettista	2016-03-21
0.2.1	Terminata stesura dell'Introduzione	Andrea Grendene	Progettista	2016-03-20
0.2.0	Stesura della prima parte dei Riferimenti	Andrea Grendene	Progettista	2016-03-19
0.1.1	Stesura dell'introduzione fino ai Riferimenti	Andrea Grendene	Progettista	2016-03-18
0.1.0	Impostazione della struttura e dei dettagli del documento	Andrea Grendene	Progettista	2016-03-15

Indice

1 Introduzione **7**
1.1 Scopo del documento . . . 7
1.2 Scopo del *prodotto*$_g$ . . . 7
1.3 Glossario . . . 7
1.4 Riferimenti . . . 7
1.4.1 Normativi . . . 7
1.4.2 Informativi . . . 7

2 Definizione obiettivi di qualità **8**
2.1 Funzionalità . . . 8
2.2 Affidabilità . . . 8
2.3 Usabilità . . . 8
2.4 Efficienza . . . 8
2.5 Manutenibilità . . . 9
2.6 Portabilità . . . 9
2.7 Altre qualità . . . 9

3 Visione generale della strategia **10**
3.1 Procedure di controllo di qualità di processo . . . 10
3.2 Procedure di controllo di qualità di *prodotto*$_g$ . . . 10
3.3 Organizzazione . . . 10
3.4 Pianificazione strategica e temporale . . . 11
3.5 Responsabilità . . . 11
3.6 Risorse . . . 11
3.6.1 Necessarie . . . 11
3.6.2 Disponibili . . . 12
3.7 Tecniche di analisi . . . 12
3.7.1 Analisi statica . . . 12
3.7.1.1 Walkthrough . . . 12
3.7.1.2 Inspection . . . 13
3.7.2 Analisi dinamica . . . 13
3.7.2.1 Test di unità . . . 13
3.7.2.2 Test di integrazione . . . 13
3.7.2.3 Test di sistema . . . 14
3.7.2.4 Test di regressione . . . 14
3.7.2.5 Test di validazione . . . 14
3.7.2.6 Tracciamento . . . 14
3.8 Misure e metriche . . . 14
3.8.1 Metriche per i processi . . . 14
3.8.1.1 Valutazione *CMM*$_g$ . . . 14
3.8.1.2 Schedule Variance . . . 15
3.8.1.3 Budget Variance . . . 15
3.8.2 Metriche per i documenti . . . 15
3.8.2.1 Indice *Gulpease*$_g$ . . . 15
3.8.3 Metriche per il codice . . . 16
3.8.3.1 Numero di parametri . . . 16
3.8.3.2 Complessità ciclomatica . . . 16
3.8.3.3 Numero di campi dati per classe . . . 16
3.8.3.4 Livello d'annidamento . . . 17
3.8.3.5 Grado di accoppiamento e Grado di instabilità . . . 17
3.8.3.6 Chiamate innestate di metodi . . . 17
3.8.3.7 Copertura del codice . . . 17
3.8.3.8 Numero di linee per metodo . . . 18

3.8.3.9 Riepilogo 18

4 Gestione amministrativa della revisione **19**
4.1 Comunicazione e risoluzione delle anomalie 19

5 Pianificazione dei test **20**
5.1 Descrizione dei test 20
5.2 Modello a V 20
5.3 Test di validazione 21
5.4 Test di sistema 21
5.5 Test di integrazione 21

A Resoconto dell'attività di verifica **22**
A.1 Fase A 22
A.1.1 Processi 22
A.1.2 Documenti 22
A.2 Fase B 22
A.2.1 Processi 22
A.2.2 Documenti 22
A.3 Fase C 23
A.3.1 Processi 23
A.3.2 Documenti 23

B Esito delle revisioni **24**
B.1 Revisione dei Requisiti 24

Elenco delle tabelle

1 Riepilogo delle metriche e dei $range_g$ di accettazione e ottimali 18
2 Esiti verifica documenti 22
3 Esiti verifica documenti 22
4 Esiti verifica documenti 23

Elenco delle figure

1 Modello a V 20

1 Introduzione

1.1 Scopo del documento

Questo documento fissa le regole decise dal *team*$_g$ per garantire una buona qualità del *prodotto*$_g$ e dei processi attuati durante l'intera durata del *progetto*$_g$. Per assicurarne il rispetto verranno svolte costantemente delle attività di verifica dei processi. In questo modo verrà garantita la qualità del *prodotto*$_g$ e si minimizzano le risorse impiegate.

1.2 Scopo del *prodotto*$_g$

Il prodotto finale consisterà di un'applicazione mobile che, interagendo con dei beacons sparsi nell'area designata, guiderà l'utente attraverso un percorso. L'utente potrà completare il percorso superando tutte le prove che gli si presenteranno nelle diverse tappe. Le prove potranno essere degli indovinelli o dei semplici giochi inerenti all'area in cui si svolge il percorso.

1.3 Glossario

Al fine di evitare ogni ambiguità nel linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, gli acronimi e le abbreviazioni che necessitano di definizione sono riportati nel documento "*Glossario v1.0.0*".

Inoltre ogni occorrenza di un vocabolo presente nel Glossario sarà posta in corsivo e seguita da una 'g' minuscola a pedice (p.es. *Glossario*$_g$).

1.4 Riferimenti

1.4.1 Normativi

- **Norme di Progetto:** "*Norme di Progetto v1.0.0*";
- **Capitolato d'appalto C2:** CLIPS: Communication & Localisation with Indoor Positioning Systems, reperibile all'indirizzo http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C2.pdf.

1.4.2 Informativi

- **Analisi dei Requisiti:** "*Analisi dei Requisiti v1.0.0*";
- **Piano di *progetto*$_g$:** "*Piano di Progetto v1.0.0*";
- **Slide dell'insegnamento Ingegneria del Software modulo A:** http://www.math.unipd.it/~tullio/IS-1/2015/;
- **Capacity Maturity Model (*CMM*$_g$):** http://en.wikipedia.org/wiki/Capability_Maturity_Model;
- **Capacity Maturity Model Integration (CMMI):** http://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration;
- ***ISO*$_g$ 9001:** http://en.wikipedia.org/wiki/ISO_9001;
- ***ISO*$_g$/IEC 9126:2001:** http://en.wikipedia.org/wiki/ISO/IEC_9126;
- ***ISO*$_g$/IEC 15504:** http://en.wikipedia.org/wiki/ISO/IEC_15504;
- **Indice *Gulpease*$_g$:** http://it.wikipedia.org/wiki/Indice_Gulpease.

2 Definizione obiettivi di qualità

Basandosi sullo standard [ISO_g/IEC 9126] il $team_g$ si impegna a garantire al $prodotto_g$ CLIPS le seguenti qualità:

2.1 Funzionalità

Il $prodotto_g$ deve garantire tutti i requisiti stabiliti nel documento "*Analisi dei Requisiti v1.0.0*" e implementarli nel modo più completo ed economico possibile.

- **Misura:** l'unità di misura adottata sarà la quantità di requisiti presenti e funzionanti nel $prodotto_g$.
- **Metrica:** la sufficienza è stabilita nel soddisfacimento dei requisiti obbligatori.
- **Strumenti:** ogni requisito dovrà superare tutti i test previsti in modo da garantire il loro funzionamento. Per avere informazioni dettagliate sugli strumenti si veda il documento "*Norme di Progetto v1.0.0*".

2.2 Affidabilità

Il $prodotto_g$ deve essere il più robusto possibile e facilmente ripristinabile in caso di errori.

- **Misura:** l'unità di misura adottata sarà il numero di esecuzioni che hanno successo.
- **Metrica:** le esecuzioni dovranno coinvolgere tutte le parti possibili del $prodotto_g$ ed esaminare il maggior numero possibile di casi. Non si può definire una soglia di sufficienza perché è impossibile determinare ogni situazione d'utilizzo possibile.
- **Strumenti:** da definire.

2.3 Usabilità

Il $prodotto_g$ deve essere di facile utilizzo per la classe di utenti designata. Inoltre deve soddisfare ogni necessità dell'utilizzatore.

- **Misura:** verrà usata come unità di misura la valutazione soggettiva del $prodotto_g$. Questo perché non esiste uno strumento adatto ad eseguire una misurazione oggettiva dell'usabilità.
- **Metrica:** purtroppo non esiste una metrica adeguata che possa determinare una soglia di sufficienza. Il $team_g$ si impegna comunque a fornire la miglior qualità d'uso possibile. Per ottenere un risultato più soddisfacente verranno consultate delle persone esterne al gruppo per verificare l'usabilità del $prodotto_g$. Per garantire una misura affidabile (ma comunque non precisa) verranno consultati almeno 15 tester. Tra questi ci saranno almeno 5 persone con una buona confidenza con la tecnologia mobile, 5 con una media confidenza e 5 con una confidenza basilare, cioè persone che possiedono uno $smartphone_g$ ma in genere lo usano solo per chiamate, messaggi e email.
- **Strumenti:** si vedano le "*Norme di Progetto v1.0.0*".

2.4 Efficienza

Il $prodotto_g$ deve fornire tutte le funzionalità nel minore tempo possibile e minimizzando l'utilizzo di risorse.

- **Misura:** il tempo di latenza per ottenere una risposta in ogni pagina del $prodotto_g$ e quello per ottenere una risposta dal server.

- **Metrica:** la sufficienza è raggiunta nel primo caso con un tempo di latenza inferiore a 0.5 secondi; nel secondo invece è raggiunta con un tempo di latenza minore di 5 secondi, ponendo che non ci siano problemi di connessione.
- **Strumenti:** si vedano le "*Norme di Progetto v1.0.0*".

2.5 Manutenibilità

Il *prodotto*$_g$ dev'essere comprensibile ed estensibile in modo facile e verificabile.

- **Misura:** l'unità di misura utilizzata saranno le metriche sul codice stabilite nella sezione 3.7.3.
- **Metrica:** il *prodotto*$_g$ deve raggiungere la sufficienza in tutte le metriche descritte nella sezione 3.7.3.
- **Strumenti:** si vedano le "*Norme di Progetto v1.0.0*".

2.6 Portabilità

Il *prodotto*$_g$ deve essere il più portabile possibile. Il gruppo Beacon Strips dovrà scegliere un sistema operativo si cui creare l'app tra *Android*$_g$ e *iOS*$_g$, mentre il supporto per l'altro sarà opzionale. Il *front end*$_g$ dev'essere utilizzabile da più dispositivi possibili. Il *back end*$_g$ deve poter girare su ogni versione dei sistemi operativi supportati che permette la lettura affidabile dei *beacon*$_g$, ovvero dalla 5.0 in poi per *Android*$_g$ e dalla 9.0 in poi per *iOS*$_g$.

- **Misura:** il *back end*$_g$ dev'essere affidabile per ogni versione del sistema operativo che supporta la lettura dei *beacon*$_g$. Ogni dispositivo con una versione dei sistemi operativi supportati uguale o maggiore deve avere un *front end*$_g$ usabile, a prescindere dalle specifiche *hardware*$_g$ o dalla risoluzione dello schermo.
- **Metrica:** il *prodotto*$_g$ dovrà raggiungere la sufficienza in tutte le metriche della sezione 3.7.3. Il *back end*$_g$ dovrà raggiungere la sufficienza in affidabilità ed efficienza in ogni dispositivo testato. Il *front end*$_g$ dovrà raggiungere la sufficienza in usabilità in ogni dispositivo testato.
- **Strumenti:** si vedano le "*Norme di Progetto v1.0.0*".

2.7 Altre qualità

Saranno inoltre garantite le seguenti caratteristiche:

- **incapsulamento:** per aumentare la manutenibilità e il riuso di codice verrà applicata la tecnica dell'incapsulamento. Questo implica che dove sarà possibile verrà favorito l'uso delle interfacce.
- **coesione:** per rendere il *prodotto*$_g$ più manutenibile, più semplice e con un indice di dipendenze minore verrà usata la tecnica della coesione. Questo significa che le funzionalità con il medesimo scopo risiederanno nello stesso componente.

3 Visione generale della strategia

3.1 Procedure di controllo di qualità di processo

Per garantire la qualità dei processi e quindi un loro miglioramento continuo verrà usato il principio *PDCA*$_g$. Di conseguenza migliorerà la qualità del *prodotto*$_g$.
Per avere il controllo dei processi, e di conseguenza qualità, è necessario che:

- i processi siano pianificati dettagliatamente;
- vi sia un controllo sul lavoro di ogni membro del *team*$_g$;
- nella pianificazione siano ripartite chiaramente le risorse.

L'attuazione di questi punti è approfondita nel "*Piano di Progetto v1.0.0*".
Analizzando la qualità del *prodotto*$_g$ si controlla anche la qualità dei processi. Un *prodotto*$_g$ scadente indica che i processi sono migliorabili.
Inoltre si possono usare delle metriche per quantificare la qualità dei processi, tali metriche sono descritte nella sezione 3.7 Misure e metriche.

3.2 Procedure di controllo di qualità di *prodotto*$_g$

Il controllo di qualità del *prodotto*$_g$ verrà garantito da:

- **quality assurance:** è l'insieme di attività realizzate per raggiungere gli obiettivi di qualità. Queste attività prevedono l'attuazione di tecniche di analisi statica e dinamica, descritte nel documento "*Norme di Progetto v1.0.0*".
- **verifica:** è un processo che determina se l'output di una fase è consistente, corretto e completo. Per tutta la durata del *progetto*$_g$ verranno svolte attività di verifica, i cui risultati sono e saranno riportati nell'Appendice A.
- **validazione:** è il processo di conferma oggettiva del soddisfacimento dei requisiti, attuato per ogni fase del *progetto*$_g$.

3.3 Organizzazione

L'organizzazione della strategia di verifica si basa sull'attuazione di verifiche per ogni processo completato. Queste verifiche controllano sia la qualità del processo stesso sia la qualità del *prodotto*$_g$ ottenuto. Grazie al diario delle modifiche sarà possibile eseguire una verifica solo sui cambiamenti effettuati.
A causa della diversa natura dei risultati ottenuti da ogni fase del processo ognuno di essi richiederà l'attuazione di specifiche procedure di verifica. Il *team*$_g$ ha deciso di adottare un ciclo di vita *incrementale*$_g$ per lo sviluppo del *progetto*$_g$. Di conseguenza il processo di verifica adottato per ogni fase del *progetto*$_g$ opererà nel modo seguente:

- **A e B:** in questa fase verranno redatti i documenti che riporteranno i requisiti individuati, le strategie e le norme adottate.
  - Verrà controllata la correttezza ortografica con il correttore automatico di *TeXstudio*$_g$.
  - Verrà controllata la correttezza lessicale con un'attenta ed accurata rilettura.
  - Verrà controllata la correttezza dei contenuti rispetto alle aspettative del documento attraverso una rilettura accurata.
  - Verrà verificato che ogni requisito abbia una corrispondenza in un caso d'uso; per farlo si controlleranno le apposite tabelle di tracciamento, con l'ausilio di *Trender*$_g$.

  - Ogni documento dovrà rispettare le "*Norme di Progetto v1.0.0*"; per verificarlo verranno adoperati gli strumenti più appropriati.
  - Verrà verificato che sia presente una didascalia per ogni rappresentazione grafica e il contenuto di ogni figura e tabella.

- **C:** verrà garantito che ogni requisito possa essere rintracciabile, attraverso il processo di verifica. Ogni requisito sarà rintracciabile nei componenti individuati in questa fase, si veda la Sezione 2 per ulteriori approfondimenti.
- **D, E e F:** i Programmatori svolgeranno le attività di codifica e di esecuzione dei test di unità per la verifica del codice. Queste attività avverranno nel modo più automatizzato possibile, rispettando anche i vincoli statici. I Verificatori controlleranno parallelamente la presenza di eventuali anomalie, definite nella Sezione 4.1.
- **G:** alla Revisione di Accettazione (RA) il gruppo Beacon Strips garantisce il funzionamento corretto del *prodotto*$_{\text{g}}$ realizzato. Le eventuali modifiche per eliminare le possibili diversità rispetto al *prodotto*$_{\text{g}}$ atteso saranno a carico del *proponente*$_{\text{g}}$.

In ogni documento viene inoltre incluso il diario delle modifiche, in modo da mantenere uno storico delle attività svolte e delle relative responsabilità.

3.4 Pianificazione strategica e temporale

Per impedire una rapida diffusione degli errori è necessario che la verifica della documentazione sia sistematica ed organizzata. In questo modo inoltre l'individuazione e la correzione degli errori avverrà il prima possibile.
Nel "*Piano di Progetto v1.0.0*"verranno pianificate le attività svolte per migliorare la qualità dei processi, le quali stabiliranno delle nuove norme di *progetto*$_{\text{g}}$.
Per ridurre la possibilità di commettere errori e/o imprecisioni di natura tecnica/concettuale ogni attività di redazione o di codifica dovrà essere preceduta da uno studio preliminare. In questo modo viene alleggerita l'attività di verifica perché richiederà a posteriori meno interventi correttivi.
Il *team*$_{\text{g}}$ ha come obiettivo il rispetto delle scadenze fissate nel "*Piano di Progetto v1.0.0*", riportate di seguito:

- revisioni formali:
  - Revisione dei Requisiti: 2016/04/18.
  - Revisione di Accettazione: 2016/09/12.
- revisioni di progresso:
  - Revisione di Progettazione: 2016/06/17.
  - Revisione di Qualifica: 2016/08/24.

3.5 Responsabilità

La responsabilità dell'assegnazione degli incarichi è a carico del Responsabile. L'Amministratore avrà come responsabilità l'adeguamento dell'ambiente di lavoro per lo svolgimento di tutti i compiti necessari alla realizzazione del *progetto*$_{\text{g}}$. Ogni componente del *team*$_{\text{g}}$ è responsabile del proprio materiale *prodotto*$_{\text{g}}$.

3.6 Risorse

3.6.1 Necessarie

Per la realizzazione del *prodotto*$_{\text{g}}$ sono necessarie risorse sia tecnologiche che umane:

- **risorse umane:** vengono descritte dettagliatamente nel "*Piano di Progetto v1.0.0*".
  - Amministratore;
  - Responsabile;
  - Analista;
  - Progettista;
  - Programmatore;
  - Verificatore.
- **risorse *software*$_g$:** sono necessari strumenti *software*$_g$ utili:
  - alla stesura di documentazione in *LaTeX*$_g$;
  - alla creazione di diagrammi in *UML*$_g$;
  - allo sviluppo nei linguaggi di programmazione scelti;
  - a semplificare ed automatizzare la verifica;
  - all'analisi statica del codice;
  - alla gestione dei test sul codice.
- **risorse *hardware*$_g$:** sono necessari computer con tutti gli strumenti *software*$_g$ descritti nelle "*Norme di Progetto v1.0.0*". È necessario avere a disposizione uno o più luoghi dove poter effettuare le riunioni interne del gruppo Beacon Strips. Sono necessari i *beacon*$_g$ in un numero sufficiente per i test del *progetto*$_g$. È necessario avere un server su cui installare il database necessario per il funzionamento del *prodotto*$_g$. Sono necessari degli *smartphone*$_g$ adatti ad eseguire i test del *prodotto*$_g$.

3.6.2 Disponibili

Ogni membro di Beacon Strips ha a disposizione almeno un computer personale dotato di tutti gli strumenti necessari con cui poter svolgere tutti i propri compiti.
A scopo di test, di supporto degli strumenti scelti per l'ambiente di sviluppo, e per il funzionamento dell'app vengono messi a disposizione uno spazio web e un server privato da Miriade SpA.
Per lo svolgimento delle riunioni interne il *team*$_g$ usa le aule del Dipartimento di Matematica dell'Università degli Studi di Padova; inoltre usa *Google Hangouts*$_g$ come strumento di discussione tramite videochiamate.

3.7 Tecniche di analisi

3.7.1 Analisi statica

L'analisi statica è una tecnica di verifica applicabile sia ai documenti che al codice. Verrà effettuata per tutta la fase di sviluppo del *progetto*$_g$ e serve per trovare anomalie, che verranno trattate come descritto nelle "*Norme di Progetto v1.0.0*". Può essere applicata nei due modi seguenti.

3.7.1.1 Walkthrough

Consiste in una lettura del documento/codice cercando anomalie ed errori ad ampio spettro, cioè senza avere un'idea precisa di quali possibili errori si potranno trovare. Per evitare incomprensioni ogni modifica verrà discussa con gli autori e concordata tra le due parti.
Il *walkthrough*$_g$ è una tecnica indispensabile nelle prime fasi di sviluppo, in cui non si ha un'idea precisa dei possibili errori. Usando questa tecnica sarà poi possibile stilare una "lista di controllo",

dove verranno archiviati gli errori più frequenti. Tale lista sarà salvata nell'Appendice delle "*Norme di Progetto v1.0.0*".
Inizialmente le fasi di verifica prevederanno perlopiù l'uso della tecnica *walkthrough*$_g$. Appena la lista di controllo sarà sufficientemente ampia si userà sempre di più la tecnica di *Inspection*$_g$.

3.7.1.2 Inspection

L'*inspection*$_g$ si basa sulla lettura mirata dei documenti/codice, cercando gli errori segnalati nella lista di controllo. Man mano che le verifiche procedono la lista verrà estesa e l'*inspection*$_g$ sarà più efficace.

3.7.2 Analisi dinamica

L'analisi dinamica verifica il funzionamento delle sole componenti *software*$_g$ tramite dei test e aiuta ad identificarne le anomalie.
Per garantire un risultato attendibile il test dev'essere ripetibile, ovvero deve ottenere sempre lo stesso output dallo stesso input quando viene eseguito nello stesso ambiente. Solo così il test garantisce di trovare eventuali problemi e quindi verificare la correttezza del *prodotto*$_g$ *software*$_g$.
Per ogni test dev'essere definito:

- **ambiente:** è l'insieme dei sistemi *hardware*$_g$ e *software*$_g$ sui quali viene pianificata l'esecuzione del test sul *prodotto*$_g$. Va inoltre specificato uno stato iniziale da cui il test deve partire;
- **specifica:** consiste nella specifica degli input e degli output attesi;
- **procedure:** è la specifica di ulteriori istruzioni sull'esecuzione del test e sull'interpretazione dei risultati.

Nelle prossime sezioni sono definiti i tipi di test effettuati sul *prodotto*$_g$ *software*$_g$ in sviluppo dal *team*$_g$.

3.7.2.1 Test di unità

Il test di unità verifica che ogni singola unità di *prodotto*$_g$ funzioni correttamente. Per unità si intende una porzione di *software*$_g$ abbastanza piccola da poterne assegnare lo sviluppo ad un singolo Programmatore.
Attraverso questo test verrà verificata la correttezza di ogni modulo base che compone il *software*$_g$, limitando gli errori di implementazione.
I test di unità sono identificati dalla seguente sintassi:

TU[Codice test]

3.7.2.2 Test di integrazione

Il test di integrazione verifica che due o più moduli già verificati funzionino come previsto quando vengono assemblati insieme. Inoltre aiuta a scovare i difetti residui sfuggiti dalla fase di test precedente.
In più con questo test viene verificata la collaborazione tra i moduli prodotti e le componenti esterne usate come *framework*$_g$ o librerie.
È possibile che vengano usate delle componenti "fittizie" al posto dei moduli non ancora completati per poter eseguire comunque dei test.
I test di integrazione sono identificati dalla seguente sintassi:

TI[identificativo del componente]

Tale identificativo corrisponde al componente i cui elementi sono integrati.

3.7.2.3 Test di sistema

I test di sistema validano l'intero *prodotto*$_{\text{g}}$ *software*$_{\text{g}}$ quando si ritiene che esso sia giunto ad una versione definitiva. Viene quindi controllato se tutti i requisiti sono stati soddisfatti.
I test di sistema sono identificati dalla seguente sintassi:

TS[Tipo Requisito][Codice Requisito]

Il tipo e il codice si riferiscono al requisito di cui verrà testato il soddisfacimento.

3.7.2.4 Test di regressione

Questo test viene eseguito dopo la modifica di una componente, e consiste nel rieseguire tutti i test. In questo modo si controlla che dopo la modifica tutti i moduli continuino a funzionare correttamente.
Il tracciamento aiuta a capire quali sono i test da ripetere (di ogni tipo) poiché potenzialmente a rischio in caso di modifica.

3.7.2.5 Test di validazione

Coincide con il collaudo del *software*$_{\text{g}}$ in presenza del *proponente*$_{\text{g}}$. Se il test risulta positivo il *prodotto*$_{\text{g}}$ sarà considerato sufficientemente maturo da permetterne il rilascio.
I test di validazione sono identificati dalla seguente sintassi:

TV[Tipo Requisito][Codice Requisito]

Il tipo e il codice si riferiscono al requisito di cui verrà testato il soddisfacimento.

3.7.2.6 Tracciamento

Sarà compito dei Verificatori controllare la corrispondenza di ogni requisito con una o più fonti. In caso di anomalie bisogna comportarsi come descritto [NOTA PER IL VERIFICATORE: bisogna controllare se da qualche parte c'è una procedura per informare della presenza dell'anomalia e scrivere quindi qui il riferimento].

3.8 Misure e metriche

Per essere utile ed informativo il processo di verifica dev'essere quantificabile. Vanno quindi stabilite a priori delle metriche, sulle quali saranno basate le misure rilevate dal processo di verifica. Nel caso in cui vi fossero delle metriche approssimate ed incerte, esse miglioreranno in modo *incrementale*$_{\text{g}}$. Questo grazie al ciclo di vita adottato descritto nel "*Piano di Progetto v1.0.0*". Possono esserci due tipi di *range*$_{\text{g}}$:

- ***range*$_{\text{g}}$ di accettazione:** insieme di valori richiesti affinché il *prodotto*$_{\text{g}}$ sia accettato.
- ***range*$_{\text{g}}$ ottimale:** insieme di valori entro i quali dovrebbe collocarsi la misurazione. Esso non è vincolante, ma fortemente consigliato. Scostamenti da tali valori necessitano una verifica approfondita.

3.8.1 Metriche per i processi

3.8.1.1 Valutazione *CMM*$_{\text{g}}$

I processi verranno valutati secondo il modello *CMM*$_{\text{g}}$. Tale modello prevede che ogni processo verrà valutato secondo le proprie "Capability" e "Maturity". Per entrambe verrà assegnato un

voto da 1 (stato iniziale) a 5 (stato controllato), che ne indicherà la valutazione complessiva.
La scelta di CMM_g rispetto ai più recenti CMMI e [ISO_g/IEC 15504] è dovuta alla maggior snellezza della valutazione. Il primo valuta usando solo due parametri, i secondi invece dividono il processo in più attributi.
È stata fatta questa scelta a causa del $progetto_g$ non molto grande. Questo renderebbe alcuni attributi superflui o sopravvalutati. Altri due elementi che hanno portato a questa scelta sono l'inesperienza del gruppo e la rotazione dei ruoli. Questi porterebbero a facili fraintendimenti e disaccordi sulle singole valutazioni.

Parametri utilizzati:

- **Range accettazione:** 3 - 5;
- **Range ottimale:** 4 - 5.

3.8.1.2 Schedule Variance

L'avanzamento dei processi verrà valutato tramite la schedule variance. Questo indice di efficacia mette in relazione il lavoro pianificato con quello svolto.
Se l'indice è maggiore di zero allora si è svolto più lavoro rispetto a quello pianificato, viceversa se è negativo. L'utilizzo di periodi di $slack_g$ aumenta la possibilità che la schedule variance assuma valori positivi.

Parametri utilizzati:

- **Range accettazione:** $\geq (preventivo\ fase * 10\%)$;
- **Range ottimale:** $\geq (preventivo\ fase * 5\%)$.

3.8.1.3 Budget Variance

L'avanzamento dei processi verrà valutato anche tramite la budget variance. Questo indice do efficienza mette in relazione il preventivo pianificato con il consuntivo.
Se l'indice è maggiore di zero allora si è riusciti ad essere più efficienti di quello pianificato, viceversa se negativo.

Parametri utilizzati:

- **Range accettazione:** $\geq (preventivo\ fase * 10\%)$;
- **Range ottimale:** $\geq (preventivo\ fase * 5\%)$.

3.8.2 Metriche per i documenti

3.8.2.1 Indice $Gulpease_g$

L'indice $Gulpease_g$ è un indice di leggibilità del testo creato appositamente per valutare la lingua italiana. Dato che non valuta la lunghezza delle parole mediante il numero di sillabe tale indice semplifica il calcolo rispetto ad altri indici di leggibilità. Esso infatti è basato sul numero di caratteri contenuti in una parola e ad altri fattori come il numero di parole e di frasi. Come tutti gli indici di leggibilità esso permette di indicare la complessità di un documento. Il calcolo da effettuare è il seguente:

$$89 + \frac{300(numero\ delle\ frasi) - 10(numero\ delle\ lettere)}{numero\ delle\ parole}$$

I risultati sono compresi tra 0 e 100, dove 100 è il valore di leggibilità più alto e 0 quello più basso. Sono state stabilite delle soglie per rapportare il livello di istruzione di un individuo con i vari gradi dell'indice:

- **inferiore a 80:** documento difficile da leggere per chi ha la licenza elementare;
- **inferiore a 60:** documento difficile da leggere per chi ha licenza media;
- **inferiore a 40:** documento difficile da leggere per chi ha la licenza superiore.http://52.58.6.246

In realtà tale indice non certifica se il testo sia comprensibile o meno. Per lo scopo dei documenti e per la formalità richiesti da essi verranno usati spesso termini tecnici che non si possono sostituire. Di conseguenza un documento potrebbe avere un ottimo indice di $Gulpease_{g}$ ma essere difficile da comprendere per i termini usati. Anche spezzare una frase può migliorare l'indice, ma interromperebbe il ragionamento fatto in quella frase. Infine usare frasi troppo dirette potrebbe essere poco professionale ai fini del documento. Per queste ragioni i documenti verranno valutati da un essere umano, così potrà stabilire se il testo è semplificabile. I limiti imposti da tale indice saranno sufficientemente rilassati per accettare frasi un po' più articolate.

Parametri utilizzati:

- **Range accettazione:** 40 - 100;
- **Range ottimale:** 50 - 100.

3.8.3 Metriche per il codice

3.8.3.1 Numero di parametri

Indica il numero di parametri formali in input di un metodo. Se il numero di parametri è elevato, lo $stack_{g}$ del programma può essere riempito rapidamente in caso di chiamate multiple innestate. I costruttori potranno superare questo limite se il numero elevato di parametri potrà facilitare la verifica tramite test e rendere più robusto agli errori l'oggetto definito.

Parametri utilizzati:

- **Range accettazione:** 0 - 8;
- **Range ottimale:** 0 - 5.

3.8.3.2 Complessità ciclomatica

Indica il numero di cammini linearmente indipendenti che attraversano il grafo di flusso di controllo del metodo. In tale grafo i nodi rappresentano unità atomiche di istruzioni. Gli archi indicano che le istruzioni collegate dai nodi collegati possono essere eseguite consecutivamente.
Un alto valore di complessità si può ridurre con la suddivisione in più metodi. È accettata anche una misurazione più lasca se questo dovesse influire in modo positivo sulla velocità di esecuzione. Oltretutto un valore alto potrebbe essere causato da strutture che in realtà aiutano ad ordinare il codice, come gli switch di condizioni.

Parametri utilizzati:

- **Range accettazione:** 0 - 10;
- **Range ottimale:** 0 - 6.

3.8.3.3 Numero di campi dati per classe

Un elevato numero di attributi interni rende la classe troppo poco specializzata, ed è indice di cattiva progettazione. Dato che la classe ricoprirà più ruoli, rende anche più difficile il mantenimento del codice.
La riduzione del numero dei campi dati si può ottenere con l'incapsulamento in nuove classi.

Parametri utilizzati:

- **Range accettazione:** 0 - 16;
- **Range ottimale:** 0 - 10.

3.8.3.4 Livello d'annidamento

Indica quante volte le strutture di controllo sono state inserite l'una all'interno dell'altra. Un alto valore può portare a difficoltà nella verifica e nell'astrazione del codice.

Parametri utilizzati:

- **Range accettazione:** 0 - 6;
- **Range ottimale:** 0 -4.

3.8.3.5 Grado di accoppiamento e Grado di instabilità

Il grado di accoppiamento è derivato da due singoli indici.

- **Accoppiamento afferente:** numero di classi esterne al $package_{\mathrm{g}}$ che dipendono da classi al suo interno. Se il numero è basso il $package_{\mathrm{g}}$ non fornirà molte funzionalità e sarà poco utile. Se invece è alto troppe classi saranno dipendenti da tale $package_{\mathrm{g}}$, rischiando di dover effettuare troppi cambiamenti ad ogni sua modifica.
- **Accoppiamento efferente:** numero di classi interne al $package_{\mathrm{g}}$ dipendenti da classi esterne. Un alto numero può indicare una scarsa progettazione.

Il grado di instabilità invece misura l'instabilità delle componenti del sistema utilizzando i due indici precedenti.
La stabilità di una componente indica la possibilità di effettuare modifiche a quella componente senza influenzarne altre all'interno dell'applicazione. Questo indice viene calcolato attraverso la formula:

$$I = \frac{Ce}{Ca + Ce}$$

dove Ce rappresenta l'accoppiamento efferente e Ca quello afferente.

Parametri utilizzati:

- **Range accettazione:** 0 - 0.8;
- **Range ottimale:** 0.3 - 0.7.

3.8.3.6 Chiamate innestate di metodi

Un grande numero di chiamate innestate di metodi può portare alla saturazione dello $stack_{\mathrm{g}}$, soprattutto in caso di un grande numero di parametri, quindi è necessario limitarne il numero.

Parametri utilizzati:

- **Range accettazione:** 0 - 8;
- **Range ottimale:** 0 - 5.

3.8.3.7 Copertura del codice

Rappresenta la percentuale di istruzioni eseguite durante i test. Maggiore è questo valore più significativi saranno i test eseguiti, quindi saranno minori le probabilità di rilevare errori residui al termine dei test.
Si puo ridurre l'indice tramite l'utilizzo di metodi semplici che non richiedono test, come i metodi getter e setter.

Parametri utilizzati:

- **Range accettazione:** $80\% - 100\%$;
- **Range ottimale:** $90\% - 100\%$.

3.8.3.8 Numero di linee per metodo

Indica il numero di $statement_g$ che compongono un metodo. Maggiore è la quantità, più lungo è il metodo, più difficile risulta comprenderne il funzionamento.
Se un metodo risulta troppo lungo allora può servire dividerlo in piu sottofunzioni. Può anche essere indice di cattiva progettazione della classe, per cui risulta essere necessaria anche una riprogettazione per intero.

Parametri utilizzati:

- **Range accettazione:** al massimo 50;
- **Range ottimale:** al massimo 40.

3.8.3.9 Riepilogo

Metriche	Range accettazione	Range ottimale
3.7.1 Metriche per i processi		
3.7.1.1 Valutazione CMM_g	3 - 5	4 - 5
3.7.1.2 Schedule Variance	$\geq -10\%$	$\geq -5\%$
3.7.1.3 Budget Variance	$\geq -10\%$	$\geq -5\%$
3.7.2 Metriche per i documenti		
3.7.2.1 Indice $Gulpease_g$	40 - 100	50 - 100
3.7.3 Metriche per il codice		
3.7.3.1 Numero di parametri	0 - 8	0 - 5
3.7.3.2 Complessità ciclomatica	0 - 10	0 - 6
3.7.3.3 Numero campi dati per classe	0 -16	0 - 10
3.7.3.4 Livello d'annidamento	0 - 6	0 - 4
3.7.3.5 Grado di instabilità	0 - 0.8	0.3 - 0.7
3.7.3.6 Chiamate innestate di metodi	0 - 8	0 - 5
3.7.3.7 Copertura del codice	80% - 100%	90% - 100%
3.7.3.8 Numero linee per metodo	≤ 60	≤ 40

Tabella 1: Riepilogo delle metriche e dei $range_g$ di accettazione e ottimali

4 Gestione amministrativa della revisione

4.1 Comunicazione e risoluzione delle anomalie

Un'anomalia corrisponde a:

- un errore ortografico;
- la violazione delle norme tipografiche del documento;
- l'uscita dal *range*$_g$ di accettazione degli indici di misurazione, descritti nella sottosezione 3.7;
- un'incongruenza del *prodotto*$_g$ rispetto a determinate funzionalità. Tali funzionalità sono state indicate nel documento "*Analisi dei Requisiti v1.0.0*";
- un'incongruenza del codice con il design del *prodotto*$_g$.

Nel caso in cui un Verificatore individui un'anomalia, dovrà aprire un *ticket*$_g$ seguendo la procedura indicata nelle "*Norme di Progetto v1.0.0*".

5 Pianificazione dei test

5.1 Descrizione dei test

Vengono ora indicati i test di validazione, di sistema e di integrazione previsti. I test di unità saranno inseriti in un momento successivo.
Poiché i test saranno applicati in uno stadio di lavoro successivo a quello attuale, lo stato dei singoli è indicato come **N.I.**: non implementati.
Di ogni test verranno indicati la tipologia ed altri parametri come specificato dalla seguente sintassi:

- per i test di unità: **TU[Codice Test]**;
- per i test di integrazione: **TI[Identificativo del componente]**;
- per i test di sistema: **TS[Tipo Requisito][Codice Requisito]**;
- per i test di validazione: **TV[Tipo Requisito][Codice Requisito]**.

In particolare:

- **Codice Requisito**: è il codice gerarchico univoco di ogni vincolo espresso in numero (esempio: 1.3.2);
- **Identificativo del componente**: corrisponde al componente i cui elementi sono integrati;
- **Tipo Requisito**: può assumere solo uno fra i seguenti valori:
  - F: funzionale;
  - Q: di qualità;
  - P: prestazionale;
  - V: di vincolo.

5.2 Modello a V

Per la pianificazione dei test si utilizzerà il Modello a V. Secondo questo modello il testing del software viene suddiviso in livelli differenti, i quali si concretizzano in un'esecuzione bottom-up che avanza sequenzialmente alle attività di codifica e di validazione. Ad ogni livello corrisponde un ciclo di uno specifico tipo di test, ed ogni test viene creato in base al relativo livello di progettazione.

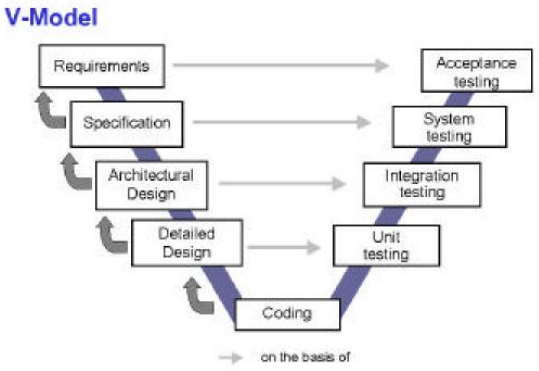

Figura 1: Modello a V

5.3 Test di validazione

I test di validazione servono per accertarsi che il prodotto realizzato sia conforme alle attese del Miriade SpA.
Per ognuno vengono indicati i passi necessari all'utente per testare i requisiti associati. Il tracciamento tra i test di validazione e i requisiti correlati viene riportato nel documento "*Analisi dei Requisiti v1.0.0*".

5.4 Test di sistema

I test di sistema servono per accertarsi che il comportamento dinamico del sistema rispetti i requisiti software individuati e descritti nel documento "*Analisi dei Requisiti v1.0.0*".

5.5 Test di integrazione

I test di integrazione servono per verificare che tutti i diversi componenti del sistema comunichino correttamente tra loro, e che vi sia all'interno del software il flusso di dati atteso.
Verrà utilizzata una strategia di integrazione incrementale per poter sviluppare e verificare più componenti in parallelo. Questo metodo permette di dare priorità ai test relativi alle componenti che vengono ritenute più importanti e quindi sarà possibile partire dalle componenti che soddisfano i requisiti obbligatori fino ad integrarle con quelle che soddisfano i requisiti opzionali. Inoltre permette di restringere la ricerca dell'errore in caso di test fallito, perché molto probabilmente l'errore si trova nel nuovo componente o dalla sua interazione con il sistema corrente. Non si dovrà escludere il caso in cui il test fallisce perché la nuova istanza di test utilizza un campione di input non trattato in precedenza, portando così il sistema a generare l'errore.
L'integrazione delle parti è bottom-up: innanzitutto verranno inserite le componenti con meno dipendenze funzionali e più funzionalità, cioè che corrispondono ai requisiti obbligatori. Di conseguenza queste componenti saranno testate molte volte in modo da ridurre la possibilità che il prodotto finale contenga difetti, e così si otterrà una versione funzionante nel minor tempo possibile. In seguito si risalirà l'albero delle dipendenze fino all'inserimento delle componenti di alto livello.
Questo metodo è più oneroso rispetto ad altri in quanto richiede che venga generato del codice di supporto, sotto forma di driver e stub, che simuli le componenti mancanti, però permette una maggiore copertura perché testa ripetutamente le componenti più importanti.

A Resoconto dell'attività di verifica

A.1 Fase A

A.1.1 Processi

A.1.2 Documenti

Sono riportati qui i valori dell'indice Gulpease per ogni documento presente durante l'attività di analisi nella fase A. Un documento è considerato valido soltanto se rispetta le metriche descritte secondo la sezione 3.7.2.1 Indice Gulpease.

Documento	Valore	Esito
"*Analisi dei Requisiti v1.0.0*"	66	Superato
"*Glossario v1.0.0*"	53	Superato
"*Norme di Progetto v1.0.0*"	56	Superato
"*Piano di Progetto v1.0.0*"	53	Superato
"*Piano di Qualifica v1.0.0*"	60	Superato
"*Studio di Fattibilità v1.0.0*"	61	Superato

Tabella 2: Esiti verifica documenti

A.2 Fase B

A.2.1 Processi

A.2.2 Documenti

Sono riportati qui i valori dell'indice Gulpease per ogni documento presente durante l'attività di analisi nella fase B. Un documento è considerato valido soltanto se rispetta le metriche descritte secondo la sezione 3.7.2.1 Indice Gulpease.

Documento	Valore	Esito
"*Analisi dei Requisiti v1.0.0*"		Superato
"*Glossario v1.0.0*"		Superato
"*Norme di Progetto v1.0.0*"		Superato
"*Piano di Progetto v1.0.0*"		Superato
"*Piano di Qualifica v1.0.0*"		Superato
"*Studio di Fattibilità v1.0.0*"		Superato

Tabella 3: Esiti verifica documenti

A.3 Fase C

A.3.1 Processi

A.3.2 Documenti

Sono riportati qui i valori dell'indice Gulpease per ogni documento presente durante l'attività di analisi nella fase C. Un documento è considerato valido soltanto se rispetta le metriche descritte secondo la sezione 3.7.2.1 Indice Gulpease.

Documento	Valore	Esito
"*Analisi dei Requisiti v1.0.0*"		Superato
"*Glossario v1.0.0*"		Superato
"*Norme di Progetto v1.0.0*"		Superato
"*Piano di Progetto v1.0.0*"		Superato
"*Piano di Qualifica v1.0.0*"		Superato
"*Studio di Fattibilità v1.0.0*"		Superato

Tabella 4: Esiti verifica documenti

B Esito delle revisioni

Durante lo sviluppo del progetto ci saranno quattro revisioni a cui sottoporsi. Il *committente*$_g$ segnalerà gli errori riscontrati fornendo una valutazione generica dell'andamento del progetto ed una dettagliata per ogni documento. Si elencano di seguito le modifiche apportate in seguito alle revisioni.

B.1 Revisione dei Requisiti

- **Studio di fattibilità:** sono stati corretti gli acronimi scritti in maniera errata.
- **Norme di progetto:** nel documento sono state aggiunte le sezioni che erano state impropriamente inserite nel documento "*Piano di Qualifica v1.0.0*".
- **Analisi dei Requisiti:** sono stati modificati numerosi casi d'uso, cambiando ad esempio la descrizione, lo scenario principale e il padre. Inoltre alcuni casi d'uso sono stati aggiunti, mentre altri sono stati eliminati. Sono stati modificati anche alcuni requisiti sia come conseguenza delle modifiche dei casi d'uso sia per le segnalazioni ricevute.
- **Piano di Progetto:**
- **Piano di Qualifica:** il documento è stato completamente rivisto a seguito delle segnalazioni. Sono state aggiunte altre metriche, soprattutto per la verifica dei processi, ed è stata ampliata l'appendice con i risultati della verifica.