Piano di Qualifica

Gruppo MINT — Progetto MaaS

Informazioni sul documento

Versione	1.0.0
Redazione	Fabiano Tavallini, Tommaso Zagni
Verifica	Enrico Canova
Approvazione	Michael Ogbuachi
Uso	Interno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo MINT

Descrizione

Questo documento descrive le operazioni di verifica e validazione seguite dal gruppo MINT durante la realizzazione del progetto MaaS.

Registro delle modifiche

Versione	Data	Collaboratori	Descrizione
1.0.0	27-03-2016	Michael Ogbuachi	Approvazione del documento.
0.2.0	27-03-2016	Enrico Canova	Verifica del documento.
0.1.0	26-03-2016	Enrico Canova	Verifica del documento.
0.0.5	25-03-2016	Tommaso Zagni	Resoconto delle attività di verifica.
0.0.4	24-03-2016	Fabiano Tavallini	Stesura gestione amministrativa della revisione.
0.0.3	22-03-2016	Tommaso Zagni, Fabiano Tavallini	Completamento stesura visione generale.
0.0.2	19-03-2016	Fabiano Tavallini	Stesura definizione obiettivi di qualità.
0.0.1	18-03-2016	Tommaso Zagni, Fabiano Tavallini	Inizio stesura documento.

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

L'obiettivo primario è la *qualità*$_G$ del prodotto e dei suoi processi, ottenibile mediante una serie di controlli stabiliti inizialmente. L'assenza di queste verifiche, combinata ad un *team di sviluppo*$_G$ con più componenti senza particolari accortezze e competenze, porta al progressivo deterioramento del materiale prodotto, sia esso codice sorgente o documentazione.

Bisogna pertanto prevenire l'inserimento, all'interno del *repository*$_G$, di materiale non congruo alle *Norme di Progetto v1.0.0* poiché si avvierebbe un graduale degrado della sua qualità.

1.1 Scopo del documento

Il Piano di Qualifica illustra la strategia di *verifica*$_G$ e *validazione*$_G$ che il gruppo MINT ha deciso di adottare per lo svolgimento del progetto. È necessario dimensionare la qualità dei prodotti e dei processi, operazione che non rientra nei normali ruoli di progettazione, bensì rappresenta una *funzione aziendale*$_G$. Secondo le strategie riportate in questo documento il *Committente*$_G$ sarà in grado di valutare oggettivamente quanto è stato prodotto e disporrà di una solida base di verifica.

1.2 Scopo del prodotto

L'obiettivo preposto dal Committente per il progetto *MaaS*$_G$ (*MongoDB*$_G$ as an admin Service), è quello di costruire un servizio web che incorpora *MaaP*$_G$ (MongoDB as an admin Platform) e lo rende direttamente disponibile, attraverso il Web, per molteplici società. L'architettura di MaaS servirà gruppi di utenti rendendogli disponibile una condivisione dedicata di una sua istanza, includendo tutti i dati e le funzionalità in loro possesso.

1.3 Ambiguità

Ogni occorrenza di termini tecnici, di dominio e gli acronimi sono evidenziati in corsivo e marcati con la lettera G in pedice. I relativi significati sono riportati nel documento *Glossario v1.0.0*.

1.4 Riferimenti

Vengono elencati qui di seguito i riferimenti sui quali si basano il presente documento stesso e l'organizzazione delle attività di verifica e validazione.

1.4.1 Normativi

- **Norme di progetto**: *Norme di Progetto v1.0.0*;
- **Capitolato d'appalto C4**: RedBabel, MaaS http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C4.pdf;
- **Standard ISO/IEC 15504**: http://en.wikipedia.org/wiki/ISO/IEC_15504;

- **Standard ISO/IEC 9126**: http://en.wikipedia.org/wiki/ISO/IEC_9126;
- **Standard IEEE 610.12-90**: https://cow.ceng.metu.edu.tr/Courses/download_courseFile.php?id=2677.

1.4.2 Informativi

- **Piano di Progetto**: *Piano di Progetto v1.0.0*;
- **SWEBOK v3**: capitolo 10;
- **Slides del corso di Ingegneria del Software mod. A**: Qualità del software http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/L08.pdf;
- **Slides del corso di Ingegneria del Software mod. A**: Qualità del $processo_G$ http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/L09.pdf;
- **Software Engineering 9th - I. Sommerville (Pearson, 2011)**: capitoli 24 e 26;
- **Metriche del software G - Ercole F. Colonese**: http://www.colonese.it/00-Manuali_Pubblicatii/08-Metriche%20del%20software_v1.0.pdf;
- **Ciclo di Deming**: https://it.wikipedia.org/wiki/Ciclo_di_Deming;
- **Indice Gulpease**: https://it.wikipedia.org/wiki/Indice_Gulpease.

2 Definizione obiettivi di qualità

La qualità perseguita nel presente documento si basa sugli standard ISO/IEC 15504 e ISO/IEC 9126 con l'obiettivo di approfondirne incrementalmente la copertura.

2.1 Qualità di processo

La qualità del processo è un fattore determinante per la qualità del prodotto. Si è deciso di perseguirla servendosi dei modelli $SPICE_G$ e $PDCA_G$.

2.1.1 Standard ISO/IEC 15504

La **qualità di processo** definita in questo standard come SPICE, specifica come la qualità è collegata alla maturazione dei processi. Il $team_G$ ha scelto questo standard ai fini di una valutazione oggettiva dei processi, per darne un giudizio di maturità e per individuare azioni migliorative. Vengono individuati dei livelli di maturità al quale il fornitore può fare riferimento per determinare le proprie capacità organizzative. Vengono definiti:

- **Modelli di riferimento**:
  - Dimensione del processo;
  - Livelli di maturità dei processi:
    - **5**: Ottimizzato

    - **4**: Predicibile
    - **3**: Stabilito
    - **2**: Gestito
    - **1**: Eseguito
    - **0**: Incompleto

    La capacità di un processo viene misurata tramite degli attributi che sono assimilabili alle metriche dei processi individuate in 3.7.2, in particolare la *Schedule Variance*$_G$ permette di capire se un processo è incompleto o gestito; il gruppo raggiungerà uno stato accettabile quando i processi diventeranno predicibili ossia quando la Schedule Variance subirà al più lievi oscillazioni;

- **Stime**: si concretizzano in una struttura per la misurazione composta da:
  - I processi di misurazione, indicati nel *Piano di Progetto v1.0.0*;
  - Un modello per la misurazione identificabile in questo documento;
  - Gli strumenti utilizzati, specificati nelle *Norme di Progetto v1.0.0*.
- **Competenze e Qualifiche di chi controlla**: lo standard redige in modo rigoroso una serie di attività volte a formare chi opera l'attività di verifica e di stesura del Piano di Qualifica. Tali competenze sono assenti all'interno del gruppo e, considerato che effettuare una formazione in linea con quanto specificato dallo standard sarebbe impossibile, tutti i membri si impegnano a studiare ed applicare al meglio quanto descritto in questo documento.

2.1.2 Ciclo di Deming

La qualità va ricercata non sul prodotto bensì sui processi alla base del prodotto, per questo il team ha scelto il metodo PDCA per il controllo delle attività di processo ripetibili e misurabili e per la manutenibilità dei processi stessi. Esso prevede l'iterazione ripetuta tra i quattro stadi definiti di seguito, assicurando un incremento della qualità ad ogni ciclo.

1. **PLAN**: vengono stabiliti gli obiettivi e i processi necessari per raggiungere la qualità attesa, nel dettaglio:
   - Identificare il problema, o i processi da migliorare; per descrivere il problema è necessario raccogliere i dati tramite misurazioni;
   - Analizzare il problema e individuare gli effetti negativi, definendo la loro importanza e le priorità di intervento;
   - Definire gli obiettivi di massima in modo chiaro e quantitativo, indicando i benefici ottenibili con il suo raggiungimento. Devono essere definiti anche i tempi, gli indicatori e gli strumenti di controllo.
2. **DO**: viene implementato il punto precedente, applicando le soluzioni individuate al problema;

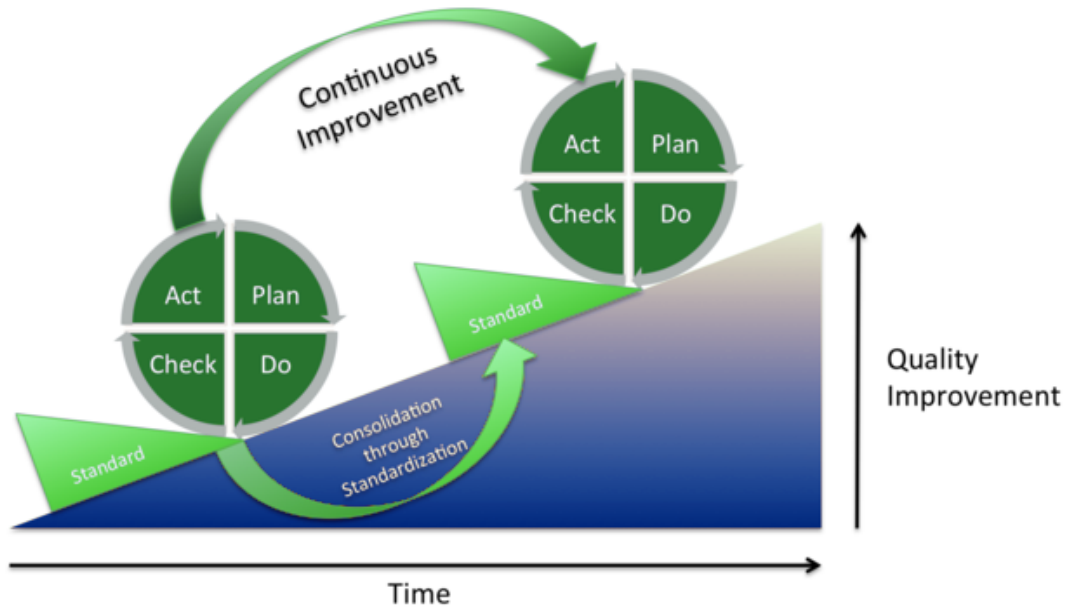

Figura 1: Continuous quality improvement with PDCA

3. **CHECK**: verificare i risultati delle azioni intraprese; un confronto con i risultati attesi sarà il riscontro se quanto operato va nella direzione giusta. Vanno considerate metriche come la Schedule Variance (vedi 3.7.2.1) e la completezza dei risultati attesi soddisfatti, vanno elaborati grafici e tabelle per avere una visione chiara di quanto rilevato. Se si è raggiunto l'obiettivo definito nello stadio di Plan, si può passare allo stadio di Act, altrimenti è necessario ripetere un nuovo ciclo PDCA sullo stesso problema, analizzando i vari stadi del ciclo precedente individuandone le cause del non raggiungimento dell'obiettivo stabilito;
4. **ACT**: La soluzione individuata viene standardizzata e tutti i membri del gruppo vengono informati e formati. Si potrà eseguire tramite riunioni o strumenti di messaggistica interna al gruppo. Terminato questo stadio si procederà con una nuova iterazione a partire dal punto 1.

2.2 Qualità di prodotto

Sono necessari degli obiettivi rivolti direttamente alla qualità del prodotto per massimizzare l'efficacia. Lo standard ISO/IEC 9126 classifica la qualità del software e definisce delle metriche per la sua misurazione.

2.2.1 Standard ISO/IEC 9126

Prendendo come riferimento questo standard il team MINT si impegna a garantire nel prodotto MaaS le seguenti qualità da esso definite, per ognuna delle verranno definite misura, metriche e strumenti:

- **Funzionalità**: il sistema prodotto deve garantire tutte le funzionalità indicate nel documento *Analisi dei Requisiti v1.0.0*. L'implementazione dei requisiti deve essere più completa ed economica possibile.
  - **Misura**: l'unità di misura usata sarà la quantità di requisiti mappati in componenti del sistema create e funzionanti;
  - **Metrica**: la sufficienza è stabilita nel soddisfacimento di tutti i requisiti obbligatori;
  - **Strumenti**: per soddisfare questa qualità il sistema deve superare tutti i test previsti. Per informazioni dettagliate sugli strumenti si veda *Norme di Progetto v1.0.0*.
- **Affidabilità**: il sistema deve dimostrarsi il più possibile robusto e di facile ripristino in caso di errori.
  - **Misura**: l'unità di misura utilizzata sarà la quantità di esecuzioni del sistema andate a buon fine;
  - **Metrica**: le esecuzioni dovranno spaziare il più possibile nella gamma di tutte le possibilità. Non è possibile definire una soglia oggettiva di sufficienza perchè non è possibile valutare ogni possibile casistica di utilizzo;
  - **Strumenti**: da definire.
- **Usabilità**: il sistema prodotto deve risultare di facile utilizzo per la classe di utenti a cui è destinato. Tale sistema deve essere facilmente apprendibile e allo stesso tempo deve soddisfare tutte le necessità dell'utente.
  - **Misura**: l'unità di misura usata sarà una valutazione soggettiva dell'usabilità. Questo è dovuto all'inesistenza di una metrica oggettiva adatta allo scopo;
  - **Metrica**: purtroppo non esiste una metrica adeguata che determinerà la sufficienza su questa qualità. Il team si impegnerà comunque nel fornire la miglior esperienza d'uso possibile;
  - **Strumenti**: si vedano le *Norme di Progetto v1.0.0*.
- **Efficienza**: il sistema deve fornire tutte le funzionalità nel più breve tempo possibile, riducendo al minimo l'utilizzo di risorse.
  - **Misura**: il tempo di latenza per ottenere una risposta nella homepage;
  - **Metrica**: la sufficienza viene definita come un tempo di latenza minore di 5 secondi nei casi in cui non si verifichino problemi di connessione;
  - **Strumenti**: si vedano le *Norme di Progetto v1.0.0*.
- **Manutenibilità**: il sistema deve essere comprensibile ed estensibile in modo facile e verificabile.
  - **Misura**: le metriche sul codice descritte nella sezione 3.7.3 costituiranno l'unità di misura;
  - **Metrica**: il software avrà le caratteristiche di manutenibilità descritte. Questo sarà possibile se il prodotto avrà la sufficienza in tutte le metriche, descritte nella sezione 3.7.3;
  - **Strumenti**: si vedano le *Norme di Progetto v1.0.0*.

- **Portabilità**: Il sistema deve essere il più portabile possibile. Il *front end*$_G$ deve essere utilizzabile da più browser possibili.
  - **Misura**: il front end deve rispettare gli standard *W3C*$_G$;
  - **Metrica**: il software avrà le caratteristiche di manutenibilità descritte. Questo sarà possibile se il prodotto avrà la sufficienza in tutte le metriche, descritte nella sezione 3.7.3;
  - **Strumenti**: si vedano le *Norme di Progetto v1.0.0*.
- **Altre qualità** Saranno inoltre importanti per il prodotto le seguenti qualità:
  - **Incapsulamento**: applicare le tecniche di *incapsulamento*$_G$ per aumentare la manutenibilità e la possibilità di riuso del codice. Sarà quindi favorito l'uso di interfacce ove possibile;
  - **Coesione**: riguarda le funzionalità che collaborano al fine di raggiungere uno stesso obiettivo. Esse devono risiedere nello stesso componente, ed hanno lo scopo di ridurre l'indice di dipendenza, favorire la semplicità e la manutenibilità.
- **Metriche**: Lo standard definisce inoltre le seguenti metriche sul prodotto:
  - **External metrics**: sono le metriche rilevate tramite l'analisi dinamica specificate in 3.7.3;
  - **Internal metrics**: sono le metriche rilevate in analisi statica specificate in 3.7.3;
  - **Quality in use metrics**: si tratta di metriche rilevabili allo stato di prodotto usabile in condizioni reali, si rimanda la definizione di tale aspetto a quando verranno trattate le considerazioni sull'usabilità del prodotto in uno scenario di utilizzo reale, questo deve avvenire non oltre la Progettazione di Dettaglio e Codifica.

3 Visione generale della strategia di verifica

La strategia adottata ha lo scopo di automatizzare il lavoro di verifica; tale scelta necessita l'uso di tool adeguatamente configurati. Lo scopo è ottenere un riscontro affidabile e numericamente trattabile che consenta di assicurare il grado di qualità predeterminato. L'aspettativa è la riduzione del lavoro manuale permettendo così una validazione semplificata.

3.1 Procedure di controllo di qualità di processo

Le linee guida per la gestione della qualità di processo seguono il modello PDCA descrivendo come devono essere attuate le procedure di controllo:

- Pianificazione dettagliata;
- Monitoraggio delle attività pianificate;
- Definizione delle risorse necessarie al conseguimento degli obbiettivi;
- Utilizzo di metriche per verificare il miglioramento della qualità dei processi.

All'interno del *Piano di Progetto v1.0.0* è descritta in dettaglio la pianificazione di queste attività per il miglioramento continuo dei processi, ottenibile in modo indiretto con la costante analisi della qualità di prodotto. Un prodotto di bassa qualità indica indubbiamente che a monte vi sta un processo migliorabile. Inoltre, per quantificare la qualità dei processi, si possono usare delle metriche. Quelle adottate sono descritte nella sezione 3.7.

3.2 Procedure di controllo di qualità di prodotto

Grazie ai sequenti processi verrà garantito il controllo di qualità del prodotto:

- **Software Quality Assurance (SQA_G)**: l'insieme delle attività realizzate al fine di garantire il raggiungimento degli obiettivi di qualità; è importante che tale processo sia preventivo e non correttivo;
- **Verifica**: assicura che l'esecuzione delle attività dei processi svolti non introduca errori nel prodotto. Durante l'intera durata del progetto verranno svolte attività di verifica sugli output dei processi, accertando che esso sia corretto, completo e rispetti regole, convenzioni e procedure;
- **Validazione**: la conferma oggettiva che assicura che i prodotti finali soddisfino i requisiti e le aspettative attese.

3.3 Organizzazione

Viene verificata la qualità dei singoli processi e dei loro output. La verifica degli output dei periodi descritti nel *Piano di Progetto v1.0.0* è programmata in attività mirate.

3.3.1 Analisi

Periodo che controlla il rispetto dei processi e della documentazione prodotta rispetto le *Norme di Progetto v1.0.0*. Sarà inoltre verificata la corrispondenza tra requisiti e casi d'uso.

3.3.2 Progettazione Architetturale

Periodo di verifica dei processi incrementali relativi all'analisi e ai nuovi documenti di progettazione. Verifica inoltre che i test siano adeguatamente pianificati come descritto nel *Piano di Progetto v1.0.0* ed eseguiti secondo quanto descritto nelle *Norme di Progetto v1.0.0*.

3.3.3 Progettazione di Dettaglio e Codifica

Periodo di verifica dei processi incrementali relativi alla progettazione assieme alla verifica delle attività di codifica tramite tecniche di analisi statica e dinamica.

Il Diario delle modifiche viene incluso in ogni documento mostrando così lo storico del documento.

3.4 Strategia

Il *Piano di Progetto v1.0.0* fissa una serie di scadenze improrogabili, risulta necessario definire con chiarezza una strategia di qualifica efficace. Gli incrementi sulla documentazione o sul codice possono essere di natura programmata, quindi prefissati nel calendario, oppure possono insorgere come inaspettati. In questo caso sarà necessario programmare le dovute modifiche; è questo il caso di bug_G o errori (vedi paragrafo 4.1). La qualità di ogni incremento è basata sul fatto che la struttura di qualifica garantisce il rispetto delle *Norme di Progetto v1.0.0*. Tale lavoro verrà svolto con l'aiuto di automatismi che segnaleranno le problematiche rilevate in modo da permettere una rapida correzione. L'utilizzo di software apposito permette di eseguire controlli mirati senza consumare risorse umane. L'implementazione di tali controlli viene descritta nelle *Norme di Progetto v1.0.0*.

3.5 Responsabilità

La responsabilità delle verifiche è attribuita al Responsabile di progetto e ai Verificatori. All'interno del *Piano di Progetto v1.0.0*sono definiti i compiti e le modalità di attuazione.

3.6 Risorse

La qualifica dei processi, essendo anch'essa un processo, consuma delle risorse distinguibili in due categorie:

- **Umane**: le figure coinvolte sono il Responsabile di progetto e il Verificatore. I processi da loro effettuati consumano ore di produttività contabilizzate e schedulate secondo il *Piano di Progetto v1.0.0*. Le ore di produttività sono fissate dalle regole di progetto (http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/PD01b.html) in un minimo di 85 e un massimo di 105 ore individuali. Il *Piano di Progetto v1.0.0* determina la distribuzione di tali quote orarie con la relativa retribuzione. Ai fini della qualifica si potrà parlare di ore di produttività, tralasciandone l'aspetto economico, in quanto non rientra nel dominio del documento succitato;
- **Tecnologiche**: riguardano i mezzi utilizzati per gli automatismi, la qualità e la loro gestione. Trattandosi esclusivamente di mezzi informatici, vengono consumate unità di calcolo considerate a costo nullo. Tale considerazione si basa sul fatto che tutti i tipi di elaborazioni informatiche sono svolte su mezzi per i quali non è richiesto né un contributo economico, né un quantitativo temporale abbastanza consistente da poter essere considerato degno di nota.

Le modalità del loro impiego sono descritte all'interno del documento *Norme di Progetto v1.0.0*.

3.7 Misure e metriche

Per monitorare l'andamento dei processi e la qualità del prodotto vengono adottate delle metriche che rendono misurabili e valutabili i processi, i documenti ed il software prodotto.

3.7.1 Metriche per i documenti

La leggibilità dei documenti ne garantisce la qualità. Si utilizza un indice per misurare la leggibilità dei testi in lingua italiana.

3.7.1.1 Gulpease

L'indice Gulpease è un indice per la leggibilità di un testo tarato sulla lingua italiana. Questo indice ha il vantaggio di utilizzare la lunghezza delle parole in lettere anziché in sillabe, semplificandone il calcolo automatico. Permette di misurare la complessità dello stile di un documento. In questo calcolo sono considerate due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere.

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

Il range dei risultati varia tra 0 e 100, dove 100 indica la leggibilità massima e 0 quella minima. In generale i documenti con:

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

Parametri utilizzati

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

3.7.2 Metriche per i processi

Tali metriche monitorano e prevedono l'andamento delle principali variabili critiche del progetto: tempi e costi. Sono utizzate metriche di tipo consultivo le quali consentono un riscontro immediato sullo stato attuale del progetto; ad ogni incremento verranno valutati tali indici e, se necessario, verranno stabiliti opportuni provvedimenti da parte del Responsabile di progetto.

3.7.2.1 Schedule Variance

Permette di calcolare le tempistiche rispetto la schedulazione delle attività pianificate alla data corrente. È un indicatore di efficacia soprattutto nei confronti del Cliente.

$$SV = BCWP - BCWS$$

Dove:

- **BCWP**: indica il valore delle attività reallizzate alla data corrente;
- **BCWS**: indica il costo pianificato per realizzare le attività di progetto alla data corrente.

Quindi con:

- **SV>0**: il lavoro prodotto è in aticipo rispetto quanto pianificato;
- **SV<0**: il lavoro è in ritardo;
- **SV=0**: il lavoro è in linea con quanto stabilito.

3.7.2.2 Budget Variance

Permette di calcolare i costi rispetto alla data corrente. È un indicatore che ha un valore unicamente contabile e finanziario.

$$BV = BCWS - ACWP$$

Dove:

- **BCWS**: indica il costo pianificato per realizzare le attività di progetto alla data corrente;
- **ACWP**: indica il costo effettivamente sostenuto per realizzare le attività di progetto alla data corrente.

Quindi:

- **BV>0**: il buget speso è minore rispetto quanto pianificato;
- **BV<0**: il budget è maggiore di quanto pianificato in ritardo;
- **BV=0**: il budget speso è in linea con quanto stabilito.

3.7.2.3 Produttività

Produttività di documentazione
Indica la produttività media di documentazione delle risorse impiegate, valutando quindi le persone coinvolte durante i diversi stadi del progetto.

$$Produttività\ di\ documentazione = Parole/Ore\ persona$$

Dove:

- **Parole**: indica il numero di parole presente nei documenti;
- **Ore persona**: indica il numero di ore produttive dei componenti del gruppo.

Parametri utilizzati

- **Range-ottimale**: [≥100].

Produttività di test
Indica la produttività media dei test realizzati.

$$Produttività\ di\ test = Numero\ di\ test/Ore\ persona$$

Dove:

- **Numero di test**: indica il numero di test eseguiti;
- **Ore persona**: indica il numero di ore produttive dei componenti del gruppo.

Produttività di codifica
Indica la produttività media delle attività di codifica.

$$Produttività\ di\ codifica = LOCs/Ore\ persona$$

Dove:

- **LOCs**: indica il numero di linee di codice prodotte;
- **Ore persona**: indica il numero di ore produttive dei componenti del gruppo.

3.7.2.4 Impegno

Indica l'impegno richiesto dal gruppo per la realizzazione del progetto.

$$Impegno = Dimensione/Produttività$$

Dove:

- **Dimensione**: indica il tempo produttivo impiegato;
- **Produttività**: indica la media delle produttività totali (di documentazione, di test, di codifica).

Parametri utilizzati

- **Range-ottimale**: [≥0,6].

3.7.2.5 Modifiche

Indica il numero di modifiche approvate dal responsabile. Le modifiche possono riguardare requisiti, funzionalità, codice e documenti.

$$Modifiche = Numero\ di\ modifiche$$

Dove:

- **Numero di modifiche**: indica le issue etichettate come "richiesta di modifica" e "approvate".

Parametri utilizzati

- **Range-accettazione**: [0 - 20];
- **Range-ottimale**: [0 - 10].

3.7.2.6 Copertura dei test

Indica la percentuale di casi coperti da test eseguiti.

$Copertura\ del\ test = Numero\ di\ funzioni\ testate * 100 / Numero\ totale\ di\ funzioni\ disponibili$

Parametri utilizzati

- **Range-accettazione**: [70 - 100];
- **Range-ottimale**: [80 - 100].

3.7.3 Metriche per il software

La prima *release*$_G$ di *Node.js*$_G$ risale a maggio 2009. È riscontrata una forte differenza tra le metriche disponibili per l'analisi statica rispetto a quelle per i linguaggi meno recenti. Nessun membro del gruppo ha conoscenza di tale linguaggio e delle sue particolarità, soprattutto riguardo all' aspetto *funzionale*$_G$. Le differenze con i linguaggi studiati nel percorso universitario portano a difficoltà nell'individuare le metriche non incentrate sulla visione ad oggetti del codice. Infine si è osservata l'assenza di strumenti per la misurazione di metriche tradizionali come la coesione e l'instabilità dei *package*$_G$. Di seguito vengono elencate le metriche per il software prodotto.

3.7.3.1 Complessità ciclomatica

La complessità ciclomatica è una metrica software che indica la complessità di un programma misurando il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso. Nel grafo sopracitato i nodi 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. Tale indice può essere applicato indistintamente a singole funzioni, *moduli*$_G$, metodi o package di un programma. Si vuole utilizzare tale metrica per limitare la complessità durante le attività di sviluppo del prodotto software. Può rivelarsi utile durante il testing per determinare il numero di casi di test necessari, infatti l'indice di complessità è un limite superiore al numero di test necessari per raggiungere il coverage completo del modulo testato. Inoltre, uno studio ha mostrato forti corrispondenze tra le metriche di complessità e il livello di coesione nei package presi in esame[1].

[1] Stein, C., G. Cox and L. Etzkorn, 2005. Exploring the Relationship between Cohesion and Complexity. J. Comput. Sci., 1: 137-144.

Parametri utilizzati

- **Range-accettazione**: [0 - 25];
- **Range-ottimale**: [0 - 10][2] .

3.7.3.2 Numero di metodi - NOM

Il Number of methods è una metrica usata per calcolare la media delle occorrenze dei metodi per package. Un package non dovrebbe contenere un numero eccessivo di metodi. Valori superiori al range ottimale massimo potrebbero indicare una necessità di maggiore decomposizione del package.

Parametri utilizzati

- **Range-accettazione**: [3 - 10];
- **Range-ottimale**: [3 - 7].

3.7.3.3 Variabili non utilizzate e non definite

La presenza di variabili non utilizzate viene considerata $pollution_{G}$, pertanto non viene tollerata. Tali occorrenze vengono rilevate analizzando l'$Abstract\ syntax\ tree_{G}$ (AST) eseguendo una cross-reference tra le variabili dichiarate e quelle inizializzate. Per sua natura, $Javascript_{G}$ non blocca l'insorgenza di tali occorrenze, pertanto si rischia di dichiarare una variabile e poi utilizzarne una con nome leggermente diverso, oppure semplicemente dichiarare una variabile che in seguito non verrà mai utilizzata.

Parametri utilizzati

- **Range-accettazione**: [0 - 0];
- **Range-ottimale**: [0 - 0].

3.7.3.4 Numero parametri per metodo

Un numero elevato di parametri per un metodo potrebbe evidenziare un metodo troppo complesso.
Non c'è una regola forte per il numero di parametri possibili in un metodo o costruttore, citando Robert Martin, in Clean Code[3] :
"The ideal number of arguments for a function is zero (niladic). Next comes one (monadic), followed closely by two (dyadic). Three arguments (triadic) should be avoided where possible. More than three (polyadic) requires very special justification – and then shouldn't be used anyway."
e Steve McConnell, in Code Complete [4]:
"limit the number of a routine's parameters to about seven, seven is a magic number for people's

[2]McCabe (dicembre 1976). A Complexity Measure. IEEE Transactions on Software Engineering: 308–320.
[3]Robert Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall (2008)
[4]Steve McConnell, Code Complete: A Practical Handbook of Software Construction. Microsoft Press (2004)

comprehension"
Vengono quindi seguite le linee guida dei seguenti parametri.

Parametri utilizzati

- **Range-accettazione**: [0 - 8];
- **Range-ottimale**: [0 - 4].

3.7.3.5 Halstead

La metrica di *Halstead*$_G$ oltre ad essere un indice di complessità, permette di identificare le proprietà misurabili del software e le relative relazioni. Si basa sull'osservazione che una metrica dovrebbe valutare l'implementazione di un algoritmo in linguaggi differenti ed essere indipendente dall' esecuzione su una specifica piattaforma.

Sono identificati i seguenti dati all'interno di un problema:

- $\mathbf{n_1}$: indica il numero di operatori distinti;
- $\mathbf{n_2}$: indica il numero di operandi distinti;
- $\mathbf{N_1}$: indica il numero totale di operatori;
- $\mathbf{N_2}$: indica il numero totale di operandi.

Da cui si ottiene:

- $\boldsymbol{n = n_1 + n_2}$: vocabolario della funzione;
- $\boldsymbol{N = N_1 + N_2}$: lunghezza della funzione.

Data la scarsa disponibilità nella rete di valori di riferimento, i range specificati sono frutto di un confronto tra il *report*$_G$ sulla complessità di una libreria *open source*$_G$ presa come esempio (https://github.com/philbooth/complexity-report/blob/master/EXAMPLE.md) e i valori dichiarati in http://www.mccabe.com/pdf/McCabeIQMetrics.pdf. Questi valori vengono dichiarati momentanei (RR) e saranno da rivalutare sia considerando altre fonti, sia considerando i valori rilevati in parti del codice che il gruppo considera come riferimento.

Halstead difficulty per-function

Il livello di difficoltà di una funzione misura la propensione all'errore ed è proporzionale al numero di operatori presenti.

$$D = \left(\frac{n1}{2}\right) * \left(\frac{N2}{n2}\right)$$

Parametri utilizzati

- **Range-accettazione**: [0 - 30];
- **Range-ottimale**: [0 - 15].

Halstead volume per-function

Il volume descrive la dimensione dell'implementazione di un algoritmo e si basa sul numero di operazioni eseguite e sugli operandi di una funzione. Il volume di una function senza parametri composta da una sola linea è 20, mentre un indice superiore a 1000 indica che probabilmente la funzione esegue troppe operazioni.

$$V = N * \log_2 n$$

Parametri utilizzati

- **Range-accettazione**: [20 - 1500];
- **Range-ottimale**: [20 - 1000].

Halstead effort per-function

Lo sforzo per implementare o comprendere il significato di una funzione è proporzionale al volume a al suo livello di difficoltà.

$$E = V * D$$

Parametri utilizzati

- **Range-accettazione**: [0 - 400];
- **Range-ottimale**: [0 - 300].

3.7.3.6 Maintainability index

Questa metrica[5] è una scala logaritmica da $-\infty$ a 171, calcolata sulla base delle linee di codice logiche, della complessità ciclomatica e dall'indice Halstead effort. Un valore alto indica una maggiore manutenibilità.

Parametri utilizzati

- **Range-accettazione**: [>70];
- **Range-ottimale**: [>90].

3.7.3.7 Use Case Points

Gli Use Case Points (UCP)[6] stimano quanto sforzo è necessario per sviluppare il prodotto basandosi su quanto lavoro il software deve svolgere.

Al fine di stimare un costo in ore-uomo di sviluppo, tale tecnica valuta i seguenti fattori:

- **Fattori tecnici dell'implementazione**: principalmente i requisiti non funzionali del sistema;

[5] Definita nel 1991 da Paul Oman e Jack Hagemeister alla University of Idaho.
[6] Definiti da Gustav Karner della Rational Software Corporation a metà del 1990.

- **Fattori ambientali**: per lo più caratterizzati dalla composizione del team;
- **Quantità e complessità degli use case**: costituiscono il numero degli use case e il numero di transizioni all'interno degli use case;
- **Quantità e complessità degli attori**: il numero di attori e come si interfacciano al sistema.

3.7.3.8 Statement Coverage

Permette di calcolare quante linee di comando di ciascun modulo delle unità sono eseguite almeno una volta nell'esecuzione dei test. Tale metrica è espressa in percentuale.

Parametri utilizzati

- **Range-accettazione**: [70 - 100];
- **Range-ottimale**: [85 - 100].

3.7.3.9 Branch Coverage

Permette di calcolare quanti rami della logica di flusso sono attraversati almeno una volta durante l'esecuzione dei test. Tale metrica è espressa in percentuale.

Parametri utilizzati

- **Range-accettazione**: [70 - 100];
- **Range-ottimale**: [85 - 100].

4 Gestione amministrativa della revisione

4.1 Comunicazione delle anomalie

Identificare le anomalie permette la correzione dei difetti ricercati dal processo di *Software Quality Management*$_G$ e informa il Responsabile di progetto sullo stato del prodotto. Analizzare e catalogare le anomalie è utile per discutere, durante revisioni e riunioni, su quali modifiche e correzioni applicare e con quale priorità. Di seguito è presente la lista delle definizioni di anomalie (IEEE 610.12-90) adottate dal gruppo:

- **Error**: differenza riscontrata tra il risultato di una computazione e il valore teorico atteso (e.g. uscita dal range di accettazione degli indici di misurazione);
- **Fault**: un passo, un processo o un dato definito in modo erroneo (e.g. violazioni di norme tipografiche da parte di un documento). Corrisponde a quanto viene definito come bug;
- **Failure**: il risultato di un fault (e.g. incongruenza del prodotto con funzionalità indicate nell'analisi dei requisiti, incongruenza del codice con il design del prodotto);

- **Mistake**: azione umana che produce un risultato errato (e.g. anomalie nel repository).

La distinzione delle anomalie consente di impostare le metriche per valutarne l'andamento e in alcuni casi predirlo, in particolare è stata scelta la metrica che conta il numero di bug per lines of code. Il gruppo utilizzerà un SCR_G (Software Change Request) individuato nelle *Norme di Progetto v1.0.0*.

4.2 Procedure di controllo per la qualità di processo

Le procedure di controllo per la qualità di processo hanno il fine di migliorare la qualità del prodotto e diminuire i costi e tempi di sviluppo. Esistono due approcci principali:

- **A maturità di processo**: riflette le buone pratiche di management e tecniche di sviluppo. L'obiettivo primario è la qualità del prodotto e la prevedibilità dei processi;
- **Agile**: sviluppo iterativo senza l'overhead della documentazione e di tutti gli aspetti predeterminabili. Ha come caratteristica la responsività ai cambiamenti dei requisti cliente e uno sviluppo rapido.

Il team adotterà il primo approccio, essendo più adatto ad un gruppo inesperto. Con una visione proattiva si cerca di avere maggior controllo e previsione sulle attività da svolgere. Questa viene anche indicata come *best practice*$_G$ per gruppi poco esperti.
Il processo con maggiore influenza sulla qualità del sistema non è quello di sviluppo ma quello di progettazione. È qui che le capacità e le esperienze dei singoli danno un contributo decisivo.
Il miglioramento dei processi è un processo ciclico composto da tre sotto-processi:

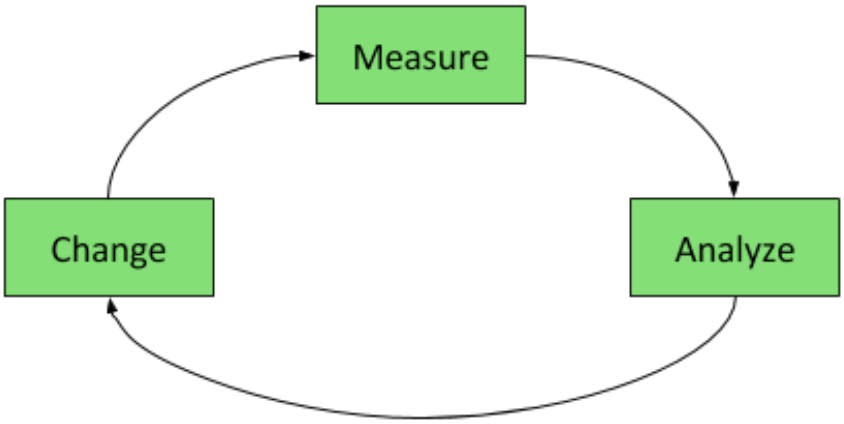

Figura 2: Il ciclo di miglioramento dei processi

- **Misurazione del processo**: misura gli attributi del progetto, punta ad allineare gli obiettivi con le misurazioni effettuate. Questo forma una *baseline*$_G$ che aiuta a capire se i miglioramenti hanno avuto effetto;
- **Analisi del processo**: vengono identificate le problematiche ed i colli di bottiglia dei processi;

- **Modifiche del processo**: i cambiamenti vengono proposti in risposta alle problematiche riscontrate.

Il team procederà nel seguente modo:

- Nella sezione Dettaglio delle verifiche tramite analisi (A.2) verranno inserite le misurazioni rilevate sulle le metriche descritte in Misure e Metriche (3.7);
- L'analisi viene effettuata i giorni precedenti alle consegne previste dal committente. Il Resoconto delle attività di verifica (A) contiene l'analisi del processo e le relative considerazioni comprendenti le problematiche riscontrate;
- Le modifiche al processo vengono attuate all'inizio del processo incrementale successivo. Queste attività sono programmate nel *Piano di Progetto v1.0.0*.

A Resoconto delle attività di verifica

A.1 Revisione dei Requisiti

L'attività di verifica svolta dai Verificatori è avvenuta come determinato dal *Piano di Progetto v1.0.0* al termine della stesura di ogni documento previsto. La verifica svolta sui documenti e sui processi è avvenuta seguendo le indicazioni delle *Norme di Progetto v1.0.0* e misurando le metriche indicate in 3.7.1.

A.2 Dettaglio delle verifiche tramite analisi

Documenti

Vengono qui riportati i valori dell'indice Gulpease per ogni documento durante l'analisi e relativo esito basato sui range stabiliti in 3.7.1.

Documento	Valore indice	Esito
Analisi dei Requisiti v1.0.0	54	superato
Glossario v1.0.0	51	superato
Norme di Progetto v1.0.0	55	superato
Piano di Progetto v1.0.0	60	superato
Piano di Qualifica v1.0.0	57	superato
Studio di Fattibilità v1.0.0	50	superato

Tabella A.1: Esiti verifica documenti in fase di Analisi