Piano di Qualifica

Gruppo SteakHolders — ***Progetto MaaP***

Informazioni sul documento

Versione	5.0.0
Redazione	Nicolò Tresoldi
Verifica	Luca De Franceschi
Approvazione	Federico Poli
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo SteakHolders
	CoffeeStrap

Descrizione

Questo documento descrive le operazioni di verifica e validazione seguite dal gruppo SteakHolders relativi al progetto MaaP.

Registro delle modifiche

Versione	Data	Persone coinvolte	Descrizione
5.0.0	2014-04-01	Federico Poli (Responsabile)	Approvazione.
4.1.0	2014-03-31	Luca De Franceschi (Verificatore)	Verifica.
4.0.2	2014-03-29	Nicolò Tresoldi (Verificatore)	Incrementate appendici verifiche, misure e PDCA.
4.0.1	2014-03-21	Nicolò Tresoldi (Verificatore)	Aggiornati range di accettazione metriche.
4.0.0	2014-03-12	Enrico Rotundo (Responsabile)	Approvazione.
3.2.0	2014-03-12	Giacomo Fornari (Verificatore)	Verifica.
3.1.3	2014-03-11	Federico Poli (Amministratore)	Incremento PDCA.
3.1.2	2014-03-11	Giacomo Fornari (Amministratore)	Aggiunto dettaglio dell'esito delle revisioni.
3.1.1	2014-03-11	Enrico Rotundo (Amministratore)	Aggiunto riassunto e dettaglio verifiche RQ.
3.1.0	2014-02-17	Giacomo Fornari (Verificatore)	Verifica.
3.0.2	2014-02-15	Federico Poli (Amministratore)	Aggiunta metrica e calcolo Use Case Points.
3.0.1	2014-02-13	Nicolò Tresoldi (Verificatore)	Correzione.
3.0.0	2014-02-05	Luca De Franceschi (Responsabile)	Approvazione.
2.2.0	2014-01-03	Gianluca Donato (Verificatore)	Verifica
2.1.3	2014-01-29	Nicolò Tresoldi (Amministratore)	Stesura dell'appendice "PDCA"
2.1.2	2014-01-26	Giacomo Fornari (Amministratore)	Aggiunte metriche architettura software
2.1.1	2014-01-20	Serena Girardi (Verificatore)	Correzioni indicate nella verifica
2.1.0	2014-01-17	Luca De Franceschi (Verificatore)	Verifica
2.0.2	2014-01-15	Enrico Rotundo (Verificatore)	Correzioni in base a tabella di valutazione interna al gruppo.
2.0.1	2014-01-14	Enrico Rotundo (Amministratore)	Correzioni struttura documento.

2.0.0	2014-01-10	Nicolò Tresoldi (Amministratore)	Aggiornamento sistema di versionamento.
1.3.1	2013-12-20	Nicolò Tresoldi (Responsabile in deroga)	Approvazione.
1.2.3	2013-12-19	Giacomo Fornari (Verificatore)	Verifica.
1.2.2	2013-12-19	Gianluca Donato (Verificatore)	Stesura resoconto verifiche.
1.2.1	2013-12-17	Giacomo Fornari (Verificatore)	Verifica.
1.1.7	2013-12-15	Enrico Rotundo (Progettista)	Stesura Pianificazione strategica.
1.1.6	2013-12-14	Enrico Rotundo (Progettista)	Stesura Gestione amministrativa.
1.1.5	2013-12-13	Enrico Rotundo (Progettista)	Stesura Appendice Qualità.
1.1.4	2013-12-12	Enrico Rotundo (Progettista)	Stesura Misure e Metriche.
1.1.3	2013-12-11	Enrico Rotundo (Progettista)	Stesura Tecniche.
1.1.2	2013-12-10	Enrico Rotundo (Progettista)	Stesura Introduzione.
1.1.1	2013-12-10	Enrico Rotundo (Progettista)	Creazione documento.

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

L'obiettivo primario è la *qualità*$_G$ del prodotto e dei suoi processi, ottenibile tramite una serie di controlli proattivi stabiliti al tempo zero. L'assenza di tali verifiche abbinata ad un team di più soggetti senza particolari accortezze e competenze, portano al progressivo deterioramento del materiale prodotto, sia esso codice sorgente o documentazione. Questo fenomeno è noto sotto il nome di *Broken windows theory*$_G$ ed è intrinseco alla componente sociale dell'uomo. Il concetto chiave è prevenire l'inserimento di materiale non aderente alle *Norme di Progetto v5.0.0* poiché, secondo la teoria succitata, innescherebbe un meccanismo che deteriora la qualità all'interno del *repository*$_G$.

Si vogliono gestire le componenti *accidentali*$_G$ dei processi, ossia tutte quelle problematiche non intrinseche alla produzione, ma che ne sono direttamente collegate; si desidera scongiurare il pericolo di operare *by correction*$_G$ per evitare modifiche in corso d'opera che possono bloccare la maturazione del prodotto e richiedere dispendiose correzioni.

1.1 Scopo del documento

Il Piano di Qualifica illustra la strategia di verifica e validazione che il gruppo SteakHolders ha deciso di adottare. È necessario dare una dimensione alla qualità dei prodotti e dei processi, operazioni che non rientrano nei normali ruoli di progetto, bensì rappresentano una *funzione aziendale*. Vi sono molteplici punti di vista della qualità, il *committente*$_G$ sarà in grado di valutare su basi oggettive quanto prodotto. Inoltre, la direzione del progetto potrà fare affidamento sulla consistenza dello stato dello stesso e il proponente avrà una solida base di verifica ideata e funzionante.

Questo documento si colloca nella parte relativa al *Project* delle *quattro P del Software Engineering* (People, Product, Project, Process) perché tratta una parte fondamentale sulle attività a supporto della produzione di MaaP.

1.2 Scopo del prodotto

Lo scopo del progetto è la realizzazione di un *framework*$_G$ per generare interfacce web di amministrazione dei dati di *business*$_G$ basato su *stack*$_G$ *Node.js*$_G$ e *MongoDB*$_G$. L'obiettivo è quello di semplificare il processo di implementazione di tali interfacce che lo sviluppatore, appoggiandosi alla produttività del framework MaaP, potrà generare in maniera semplice e veloce ottenendo quindi un considerevole risparmio di tempo e di sforzo. Il fruitore finale delle pagine generate sarà infine l'*esperto di business*$_G$ che potrà visualizzare, gestire e modificare le varie entità e dati residenti in *MongoDB*$_G$. Il prodotto atteso si chiama *MaaP*$_G$ ossia *MongoDB as an admin Platform*.

1.3 Glossario

Ogni occorrenza di termini tecnici, di dominio e gli acronimi sono marcati con una G in pedice e riportati nel documento *Glossario v5.0.0*.

1.4 Riferimenti

Vengono elencanti i riferimenti su cui si è basata l'organizzazione dell'attività di qualifica.

1.4.1 Normativi

- **Norme di Progetto:** *Norme di Progetto v5.0.0*;
- **Capitolato d'appalto C1:**
  http://www.math.unipd.it/~tullio/IS-1/2013/Progetto/C1.pdf;
- **Piano di Progetto:** *Piano di Progetto v5.0.0*;
- **ISO/IEC 15504**
  http://en.wikipedia.org/wiki/ISO/IEC_15504;
- **ISO/IEC 9126**
  http://en.wikipedia.org/wiki/ISO/IEC_9126.

1.4.2 Informativi

- **Slide di Ingegneria del Software mod. A:**
  http://www.math.unipd.it/~tullio/IS-1/2013/;
- **SWEBOK 2004 Version - capitolo 5 e 11:**
  http://www.computer.org/portal/web/swebok/htmlformat;
- **Software Engineering 9th - I. Sommerville (Pearson, 2011) - capitoli: 8, 24, 26:**
  http://www.pearsoned.co.uk/bookshop/detail.asp?item=100000000377819;
- **Software Cost Estimation With Use Case Points:**
  http://tynerblain.com/blog/2007/02/12/software-cost-estimation-ucp-1/.

2 Visione generale della strategia di verifica

La strategia generale adottata è quella di automatizzare il più possibile il lavoro di verifica; questo richiede scelta e uso di *tools*$_G$ adeguatamente configurati. L'obiettivo è avere un riscontro affidabile e numericamente trattabile che permetta di assicurare il grado di qualità predeterminato. Il lavoro manuale verrà così ridotto al minimo e confinato all'opera di validazione. La speranza è che dei buoni processi portino ad un buon software.

2.1 Definizione obiettivi

2.1.1 Qualità di processo

La qualità del processo è un fattore determinante per la qualità del prodotto. Si è deciso di perseguire la qualità servendosi di due modelli:

- *SPICE*$_G$[1], definito nello standard ISO/IEC 15504, ai fini di una valutazione oggettiva dei processi, per darne un giudizio di maturità e per individuare azioni migliorative;
- *PDCA*$_G$[2], per il controllo delle attività di processo ripetibili e misurabili e per la manutenibilità dei processi stessi incrementandone la qualità.

2.1.2 Qualità di prodotto

Oltre alla qualità di processo, sono necessari degli obiettivi rivolti direttamente alla qualità del prodotto per massimizzare l'efficacia. A tal fine, lo standard ISO/IEC 9126[3] classifica la qualità del software e definisce delle metriche per la sua misurazione.

2.2 Procedure di controllo di qualità di processo

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

- La pianificazione deve essere dettagliata;
- Le attività pianificate devono essere monitorate;
- Le risorse necessarie per conseguire gli obiettivi devono essere definite;
- Il controllo deve servirsi delle metriche per verificare il miglioramento della qualità del processo.

La pianificazione delle attività volte al miglioramento continuo dei processi sono descritte nel *Piano di Progetto v5.0.0*.

[1] Si veda appendice C.1 per approfondimenti
[2] Si veda appendice C per approfondimenti
[3] Si veda appendice C.2 per approfondimenti

2.3 Procedure di controllo di qualità di prodotto

Il controllo per la qualità di prodotto definisce i seguenti processi:

- **Software Quality Assurance** (SQA_G): deve assicurare che i processi pianificati siano appropriati e successivamente implementati secondo la pianificazione, e che siano forniti sistemi di misurazione dei processi. Tale processo deve essere preventivo invece che correttivo;
- **Verifica**: si occupa di accertare che l'esecuzione delle attività di processi svolti nel periodo in esame non abbia introdotto errori nel prodotto. La verifica viene svolta sui prodotti dei processi per accertare il rispetto delle regole, delle convenzioni e delle procedure;
- **Validazione**: si occupa di accertare che i prodotti finali realizzati siano conformi alle attese.

2.4 Organizzazione

Viene verificata la qualità di ogni processo e di ogni output da essi prodotti. Ogni periodo descritto nel *Piano di Progetto v5.0.0* produce output di diverso tipo, per questo è necessario programmare le attività di verifica in modo mirato:

- **Analisi**: in questo periodo si controlla che i processi e la documentazione prodotta rispettino le *Norme di Progetto v5.0.0* e verrà verificato che ogni requisito abbia corrispondenza in un caso d'uso;
- **Progettazione Architetturale**: in questo periodo si verificano i processi incrementali relativi all'analisi e ai nuovi documenti di progettazione, e che i test siano adeguatamente pianificati come descritto nel *Piano di Progetto v5.0.0* ed eseguiti secondo quanto descritto nelle *Norme di Progetto v5.0.0*;
- **Progettazione di Dettaglio e Codifica**: in questo periodo vanno verificati i processi incrementali relativi alla progettazione assieme alla verifica delle attività di codifica grazie a tecniche di analisi statica e dinamica.

Il *Diario delle modifiche* viene incluso in ogni documento al fine di tracciarne uno storico dell'evoluzione.

2.5 Strategia

Il *Piano di Progetto* fissa una serie di scadenze improrogabili, pertanto è 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 3.1). La qualità di ogni incremento si basa sul fatto che la struttura di qualifica garantisce il rispetto delle *Norme di Progetto v5.0.0*. Questo 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 v5.0.0*.

2.6 Responsabilità

La responsabilità della verifica viene attribuita al *Responsabile* di progetto e ai *Verificatori*. I compiti e le modalità di attuazione sono definiti nel *Piano di Progetto v5.0.0*.

2.7 Risorse

La qualifica dei processi, essendo un processo, consuma delle risorse, che si dividono 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 v5.0.0*. Le ore di produttività sono fissate dalle regole di progetto (http://www.math.unipd.it/~tullio/IS-1/2013/Progetto/PD01b.html) in un minimo di 85 e un massimo di 105 ore individuali. Il *Piano di Progetto v5.0.0* 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 per 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 dettagliatamente nelle *Norme di Progetto v5.0.0*.

2.8 Misure e Metriche

Vengono adottate delle metriche per rendere misurabili e valutabili i processi, i documenti ed il software prodotto. La visione non vuole essere comparativa, ma serve al gruppo per monitorare l'andamento dei processi e la qualità del prodotto.

2.8.1 Metriche per i processi

Le seguenti metriche rappresentano un indicatore volto a monitorare e prevedere l'andamento delle principali variabili critiche del progetto (i tempi e i costi). Sono state scelte metriche di tipo $consuntivo_G$ perché danno 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.

2.8.1.1 Schedule Variance

Indica se si è in linea, in anticipo o in ritardo rispetto alla schedulazione delle attività di progetto pianificate.

$$SV = BCWP - BCWS$$

Dove $BCWP$ indica il valore delle attività realizzate alla data corrente e $BCWS$ rappresenta il costo pianificato per realizzare le attività di progetto alla data corrente. È un indicatore di efficacia soprattutto nei confronti del Cliente. Se $SV > 0$ significa che il progetto sta procedendo con maggior velocità a quanto pianificato, viceversa se negativo.

2.8.1.2 Budget Variance

Indica se alla data corrente si è speso di più o di meno rispetto a quanto previsto.

$$BV = BCWS - ACWP$$

Dove $BCWS$ indica il costo pianificato per realizzare le attività di progetto alla data corrente e $ACWP$ rappresenta il costo effettivamente sostenuto alla data corrente. È un indicatore che ha un valore unicamente contabile e finanziario. Se $BV > 0$ significa che il progetto sta spendendo il proprio budget con minor velocità di quanto pianificato, viceversa se negativo.

2.8.1.3 Produttività

Produttività di documentazione Indica la produttività media di documentazione delle risorse impiegate, cioè delle persone coinvolte, nei diversi stadi del progetto.

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

Dove *Parole* indica il numero di parole presenti nei documenti e *Ore persona* rappresenta 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 $Numeroditest$ indica il numero di test eseguiti e $Orepersona$ rappresenta il numero di ore produttive dei componenti del gruppo.

I parametri utilizzati per questa metrica verranno stabiliti entro l'inizio dell'esecuzione dei test.

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 e *Ore persona* rappresenta il numero di ore produttive dei componenti del gruppo.

I parametri utilizzati per questa metrica verranno stabiliti alla entro l'inizio del processo di codifica.

2.8.1.4 Impegno

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

$$Impegno = Dimensione/Produttività$$

Dove *Dimensione* indica il tempo produttivo impiegato e *Produttività* rappresenta la media delle produttività totali (di documentazione, di test, di codifica).

Parametri utilizzati:

- Range-ottimale: [≥0,6].

2.8.1.5 Modifiche

Indica quante modifiche sono state 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].

2.8.1.6 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].

2.8.2 Metriche per i documenti

La *leggibilità* dei documenti è indispensabile per garantirne la qualità. Si è scelto di adottare un indice per misurare la leggibilità dei testi in lingua italiana:

2.8.2.1 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. Permette di misurare la complessità dello stile di un documento. 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:

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

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.

Parametri utilizzati:

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

2.8.3 Metriche per il software

La prima release di *Node.js*$_G$ risale a Maggio 2009. È stata riscontrata una forte differenza tra le metriche disponibili per l'analisi statica rispetto a quelle per i linguaggi meno recenti. Inoltre nessun membro del gruppo ha conoscenza di tale linguaggio e delle sue particolarità, come l'aspetto *funzionale*$_G$. Tali differenze con i linguaggi studiati nel percorso universitario si sono tradotte nella difficoltà di individuare 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. Di seguito vengono elencate le metriche per il software prodotto.

2.8.3.1 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. Questo indice può essere applicato indistintamente a singole funzioni, moduli, metodi o package di un programma. Si vuole utilizzare tale metrica per limitare la complessità durante le attività di sviluppo del prodotto software. Durante il testing è utile 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[4].

Parametri utilizzati:

- Range-accettazione: [0 - 25];
- Range-ottimale: [0 - 10][5] .

2.8.3.2 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].

2.8.3.3 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].

2.8.3.4 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[6] :
"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 [7]

[4]Stein, C., G. Cox and L. Etzkorn, 2005. Exploring the Relationship between Cohesion and Complexity. J. Comput. Sci., 1: 137-144.
[5]McCabe (dicembre 1976). A Complexity Measure. IEEE Transactions on Software Engineering: 308–320.
[6]Robert Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall (2008)
[7]Steve McConnell, Code Complete: A Practical Handbook of Software Construction. Microsoft Press (2004)

"limit the number of a routine's parameters to about seven, seven is a magic number for people's comprehension"
ci atteniamo come linea guida ai parametri sotto evidenziati.

Parametri utilizzati:

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

2.8.3.5 *Halstead*$_G$

La metrica di *Halstead*$_G$ non è solamente un indice di complessità, ma identifica 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 dal esecuzione su una specifica piattaforma.

In un problema vengono identificati:

- n_1 = il numero di operatori distinti
- n_2 = il numero di operandi distinti
- N_1 = il numero totale di operatori
- N_2 = il numero totale di operandi

Da cui vengono calcolati:

- $n = n_1 + n_2$: vocabolario della funzione
- $N = N_1 + N_2$: lunghezza della funzione

Data la bassa 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. Tali 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*$_G$ 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*$_G$ 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*$_G$ 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].

2.8.3.6 Maintainability index

Questa metrica[8] è una scala logaritmica da $-\infty$ a 171, calcolata sulla base delle linee di codice logiche, della complessità ciclomatica e dall'indice *Halstead*$_G$ effort. Un valore alto indica una maggiore manutenibilità.

Parametri utilizzati:

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

2.8.3.7 Use Case Points

Gli Use Case Points (UCP)[9] 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;
- *Fattori ambientali*, per lo più caratterizzati dalla composizione del team;
- *Quantità e complessità degli use case*, 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.

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

2.8.3.8 Statement Coverage

Calcola 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].

2.8.3.9 Branch Coverage

Calcola quanti rami della logica di flusso sono attraversati almeno una volta nell'esecuzione dei test. Tale metrica è espressa in percentuale.

Parametri utilizzati:

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

3 Gestione amministrativa della revisione

3.1 Comunicazione delle anomalie

Il processo di *Software Quality Managment*$_G$ è finalizzato alla ricerca dei difetti. L'identificazione delle anomalie ne permette la correzione e informa il *Responsabile* di progetto sullo stato del prodotto. Distinguere e catalogare le anomalie è utile per discutere durante revisioni e riunioni su che correzioni attuare e con quale priorità. Di seguito vengono elencate le 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*$_G$;
- **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 catalogazione delle anomalie permette l'impostazione di metriche in grado di valutarne l'andamento e in alcuni casi di predirlo. In particolare è stata scelta la metrica che conteggia il numero di *bug*$_G$ *per lines of code*. Il *SCR*$_G$ (software change request) utilizzato dal gruppo viene individuato nelle *Norme di Progetto v5.0.0*.

3.2 Controlli per la qualità di processo

Le procedure di controllo per la qualità di processo sono finalizzate a migliorare la qualità del prodotto e/o 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.

Verrà utilizzato il primo approccio, in quanto più adatto ad un team 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:

- 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.

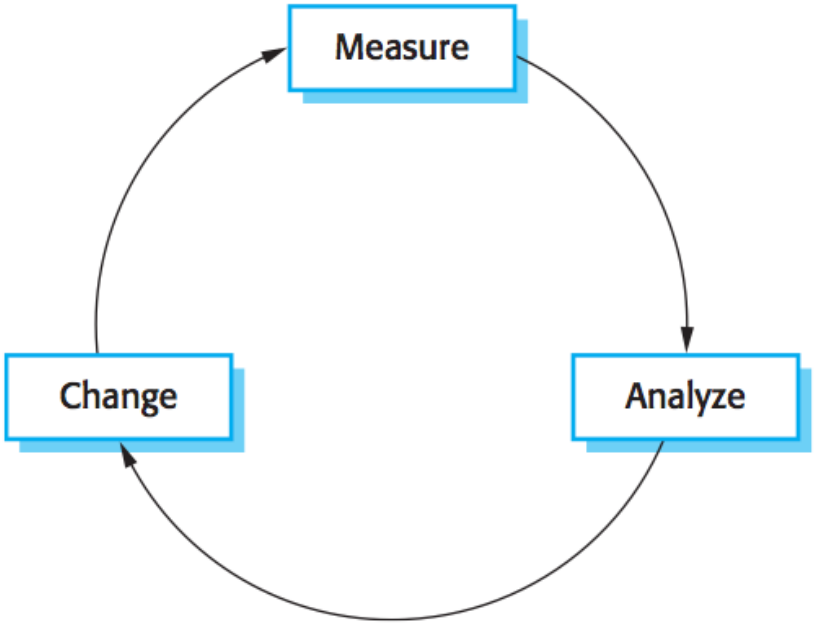

Figura 1: Il ciclo di miglioramento dei processi

Il gruppo procederà nel seguente modo:

- Nella sezione *Dettaglio delle verifiche tramite analisi* (B.3) verranno inserite le misurazioni rilevate sulle le metriche descritte in *Misure e Metriche* (2.8);
- L'analisi viene effettuata i giorni precedenti alle consegne previste dal committente; il *Riassunto delle attività di verifica* (B.1) contiene l'analisi del processo, 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 v5.0.0*.

A Pianificazione dei test

Si vuole adottare una strategia di verifica del software tramite test opportunamente predeterminati e che garantiscano almeno un test per ogni requisito. I test sono l'applicazione delle tecniche di verifica dinamica introdotte nelle *Norme di Progetto v5.0.0*; tali attività, oltre a richiedere l'esecuzione del programma, devono poter essere ripetibili, ossia tramite delle specifiche su come riprodurre i test vogliamo che il loro output sia deterministico. È importante che i test di unità vengano svolti in parallelo, dando precedenza alle unità che producono risultati utili alla comprensione del loro funzionamento integrato, l'ambiente di testing deve soddisfare tale obiettivo. L'attività di test deve produrre un *log*$_G$ che specifica quando e chi ha eseguito il test e con quali input; l'insorgenza di *failure*$_G$ deve essere tracciata e catalogata.

A.1 Livelli di testing

Il testing del software viene suddiviso in livelli differenti e si concretizzano in un esecuzione bottom-up che avanza sequenzialmente alle attività di codifica e di validazione. I test che si andranno ad applicare sono di cinque tipi, riservando la specifica delle ultime due tipologie alla prossima revisione:

1. Test di Validazione (TV): viene verificato che il prodotto soddisfi quanto richiesto dal *proponente*$_G$ individuando delle macro azioni da eseguire sul sistema che un normale utente svolge comunemente;
2. Test di Sistema (TS): sono test relativi al comportamento dell'intero sistema ossia viene verificato che la sua architettura generale funziona complessivamente bene;
3. Test di Integrazione (TI): vengono verificate le componenti del sistema contenute nella *Specifica Tecnica v4.0.0*, ossia viene verificato che i *package*$_G$ siano funzionanti e in grado di funzionare nel loro insieme;
4. Test di Unità (TU): viene testata ogni unità, ossia la più piccola parte di lavoro assegnabile ad un programmatore. In questo progetto una unità dovrebbe corrispondere ad una `function` o a un `method`;
5. Test di Regressione (TR): possono essere test di tutte le tipologie succitate che devono mostrare il funzionamento del prodotto a seguito di una modifica.

La figura 2 illustra come i test elencati vengono distribuiti durante in ciclo di vita del prodotto.

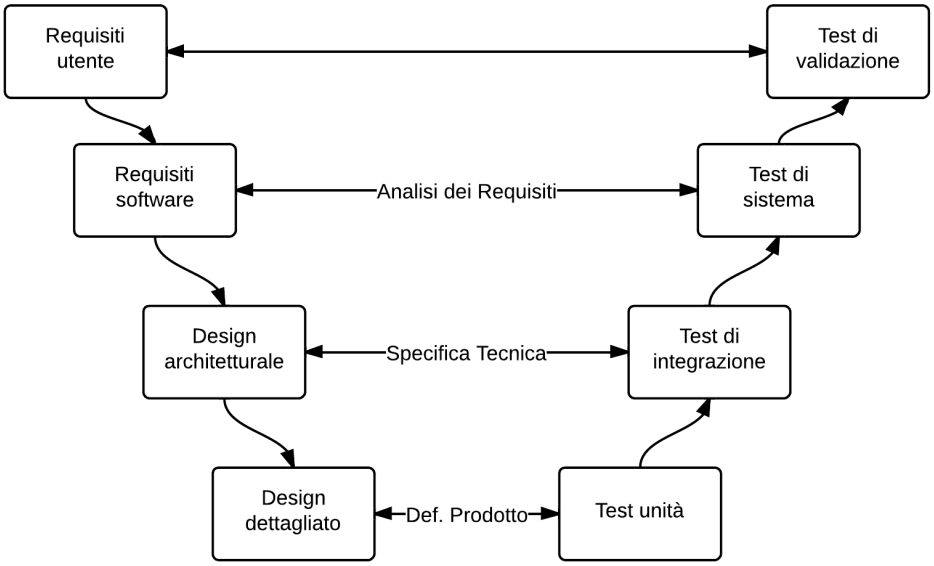

Figura 2: V-Model per il testing software

A.2 Test di validazione

In questa sezione vengono elencati i test di validazione per verificare che il prodotto sia conforme alle attese. I test si svolgono seguendo e verificano tutti passi di cui si compongono. I requisiti che non sono stati accettati nel *Analisi dei Requisiti v4.0.0* sono qui marcati con un $*$ ad indicare che il test associato non verrà effettuato. I test eseguiti sono indicati con una *E* (Eseguito), mentre i test non eseguiti sono indicati con *N.E* (Non Eseguito).

Test di Validazione	Descrizione	Stato	Requisito
TV-RA1O 1	L'utente non autenticato intende accedere all'applicazione, per farlo deve inserire le proprie credenziali composte da una email ed una password. All'utente è richiesto di: • Raggiungere la pagina di autenticazione; • Inserire la mail nel campo apposito; • Inserire la password; • Procedere con l'autenticazione.	E	RA1O 1
TV-RA1O 2	L'utente intende recuperare la password d'accesso all'applicazione. All'utente è richiesto di: • Essere autenticato; • Raggiungere la pagina per il reset della password; • Richiedere il reset; • Raggiungere la casella email collegata all'account del sistema; • Seguire il link contento nella mail: • Compilare il form richiedente la nuova password; • Eseguire il Logout e autenticarsi con la nuova password.	E	RA1O 2
TV-RA1D 3	L'utente autenticato può visualizzare la pagina di Dashboard nella quale potrà aver accesso ad esempio alla lista delle collection presenti e ad altre funzionalità disponibili. All'utente è richiesto di: • Accertarsi di essere autenticato; • Accedere alla pagina Dashboard tramite il menu di navigazione;	E	RA1D 3

TV-RA1O 4	L'utente autenticato, selezionata una *Collection*$_G$, ne visualizza in forma ta: bellare tutti i documenti che contiene. Di questa collection può filtrarne i risultati visualizzabili, può eseguire tramite bottoni predisposti nella pa: gina azioni personalizzati e per ogni *Document*$_G$, selezionarlo e visualizzar: ne la show-page corrispondente. L'Admin ha i permessi per modifi: care un documento o eliminare un *Document*$_G$. All'utente è richiesto di: • Essere autenticato; • Aprire la show-page relativa ad un Document; • Usare i filtri per filtrare la Collection • Eseguire un azione personalizza: ta, laddove presente; • Se admin, modificare un Document; • Se admin, eliminare un Document;	E	RA1O 4
TV-RA1O 5	L'utente visualizza la pagina show-pa: ge corrispondente ad un *Document*$_G$ selezionato visualizzandone gli attribu: ti in forma tabellare. In questa pagina può aprire la sho: w-page o l'index-page dell'array di *Document*$_G$ degli attributi innestati se presenti, eseguire un'operazione perso: nalizzata se disponibile. L'Admin può eliminare il *Document*$_G$ a cui la show-page corrisponde o modificarlo. All'utente è richiesto di: Essere autenticato; Aprire la show-page degli attributi innestati; Aprire l'index-page dell'arra di Document; Eseguire, se presente, un operazione personalizzata; Se admin, modificare il Document; Se admin, eliminare il Document.	E	RA1O 5

TV-RA1O 6	L'Admin entra nella sua pagina di amministrazione nella quale visualizza una *Collection-Index*$_G$ di tutti gli utenti registrati al sistema. All'utente è richiesto di: • Essere autenticato come admin; • Accedere alla pagina di creazione nuovi utenti; • Creare un nuovo utente; • Accedere alla pagina degli utenti registrati al sistema; • Visualizzare la pagina Collection-Show di un utente;	E	RA1O 6
TV-RF1O 8	Lo sviluppatore deve poter creare un nuovo progetto tramite linea di comando. Allo sviluppatore è richiesto di: • Richiamare il comando di creazione di un nuovo progetto; • Passare come parametro il nome della directory che conterrà il progetto; • Verificare che siano state im: portate le librerie necessarie al corretto funzionamento del sistema; • Verificare che sia stato creato il file di configurazione di default dell'applicazione generata; • Verificare che sia stato creato il sistema di autenticazione per l'applicazione generata; • Verificare che siano state create le directory di descrizione delle pagine web; • Verificare che sia stato creato un account admin di default.	E	RF1O 8

TV-RF1O 9	Lo sviluppatore deve poter configurare le Collection tramite il DSL di Maap Framework. All'utente è richiesto di: • creare una Collection-index tramite DSL; • creare una Collection-show tramite DSL; • modificare il nome della Collection; • modificare l'ordine di visualizza: zione della Collection.	E	RF1O 9
TV-RS1F 10	L'utente autenticato verifica che il *framework*$_G$ MaaP sia messo a disposi: zione dal sistema *MaaS*$_G$ come servizio Web. All'utente è richiesto di: • Accedere alla pagina di modifica del proprio profilo; • Modificare i dati associati al proprio profilo; • Verificare che i dati siano stati aggiornati; • Gestire i file di configurazione; • Eliminare il proprio account; • Verificare l'inaccessibilità al ser: vizio tramite l'autenticazione con le credenziali associate all'account eliminato.	N.E	RS1F 10*
TV-RA1D 11	L'utente non autenticato deve potersi registrare all'applicazione MaaP. All'utente è richiesto di: • Inserire la mail nell'apposito campo di testo; • Inserire la password nell'apposito campo di testo; • Verificare che l'account sia stato registrato tramite l'autenticazione all'applicazione.	E	RA1D 11

TV-RA1D 12	L'utente autenticato deve poter eseguire il logout dall'applicazione. All'utente è richiesto di: • Selezionare l'apposita opzione di logout; • Verificare di non essere più autenticato.	E	RA1D 12
TV-RA1D 13	L'utente autenticato deve poter modificare le proprie credenziali d'accesso all'interno della propria pagina profilo. All'utente viene richiesto di: • Accedere alla propria pagina profilo; • Modificare la propria mail; • Modificare la propria password; • Eseguire il logout; • Autenticarsi con le nuove credenziali.	E	RA1D 13
TV-RF1O 14	Lo sviluppatore deve poter configurare i database che compongono il sistema MaaP. Allo sviluppatore è richiesto di: • Configurare la connessione al database delle credenziali degli utenti; • Configurare il $namespace_G$ corrispondente, se la funzione di $namespace_G$ è abilitata; • Configurare la connessione al database delle $Collection_G$; • Configurare il $namespace_G$ corrispondente, se la funzione di $namespace_G$ è abilitata; • Selezionare un $namespace_G$ per il database da configurare, se la funzione di $namespace_G$ è abilitata.	E	RF1O 14

TV-RA1F 15	L'admin deve poter gestire gli indici da un'apposita pagina. All'admin è richiesto di: • Accedere alla pagina di gestione degli indici; • Visualizzare i suggerimenti per la creazione degli indici; • Creare un indice; • Creare un indice da quelli suggeriti; • Eliminare un indice; • Eliminare un indice da quelli suggeriti.	N.E	RA1F 15*
TV-RF1F 16	Lo sviluppatore deve poter abilitare i $namespace_G$ per l'applicazione creata. Allo sviluppatore è richiesto di: • Attivare il $namespace_G$.	N.E	RF1F 16*

Tabella A.1: Tracciamento Test di Validazione - Requisiti

A.3 Test di sistema

Vengono qui descritti i test di sistema che andranno a verificare il funzionamento complessivo delle componenti. I requisiti che non sono stati accettati nel *Analisi dei Requisiti v4.0.0* sono qui marcati con un $*$ ad indicare che il test associato non verrà effettuato. I test eseguiti sono indicati con una *E* (Eseguito), mentre i test non eseguiti sono indicati con *N.E* (Non Eseguito).

Test Sistema	**Descrizione**	**Stato**	**Requisito**
TS-RA1O 1.1	Verificare che durante la verifica del: le credenziali l'indirizzo email venga immesso tramite un campo di testo apposito.	E	RA1O 1.1
TS-RA1O 1.2	Verificare che durante la verifica delle credenziali, la password venga immessa tramite un capo di testo apposito.	E	RA1O 1.2
TS-RA1O 1.3	Verificare che il sistema verifichi le credenziali di un utente tramite un database indipendente da quello che contiene la Collection.	E	RA1O 1.3
TS-RA1O 1.3.1	Verificare che, in caso di fallimento dell'autenticazione di un utente, il sistema visualizzi una pagina di errore.	E	RA1O 1.3.1
TS-RA1O 1.3.2	Verificare che in caso in cui auten: ticazione vada a buon fine, l'utente venga reindirizzato automaticamente sulla dashboard dell'applicazione.	E	RA1O 1.3.2
TS-RA1O 2.1	Verificare che il sistema permet: ta il recupero password attraverso l'inserimento dell'email.	E	RA1O 2.1
TS-RA1O 2.2	Verificare che un utente non autenti: cato che richiede il reset della propria password riceva un email con un link per il reset.	E	RA1O 2.2
TS-RA1O 2.3	Verificare che un utente non autentica: to possa resettare la propria password tramite l'inserimento di una nuova password.	E	RA1O 2.3
TS-RA1O 4.1	Verificare che la visualizzazione di una Collection-index consista in una tabella le cui righe corrispondono ai document presenti nel database e le cui colonne siano i relativi attributi.	E	RA1O 4.1
TS-RA1O 4.1.1	Verificare che ogni riga della tabella corrispondente ad un Document abbia una chiave selezionabile che rimanda alla corrispondente pagina show.	E	RA1O 4.1.1
TS-RA1D 4.1.2	Verificare che l'admin possa eliminare un documento tramite un link rapido.	E	RA1D 4.1.2

TS-RA1D 4.1.3	Verificare che l'admin possa modificare un document della collection-index.	E	RA1D 4.1.3
TS-RA1D 4.2	Verificare che sia possibile visualizzare un sottoinsieme di Document tramite dei filtri personalizzati sugli attributi.	N.E	RA1D 4.2*
TS-RA1F 4.3	Verificare che l'amministratore possa creare un nuovo Document nella base di dati.	N.E	RA1F 4.3*
TS-RA1O 5.1	Verificare che l'admin possa editare ogni singolo attributo modificabile del documento della pagina show.	E	RA1O 5.1
TS-RA1F 5.2	Verificare che l'utente possa esegui: re un'azione personalizzata tramite l'esecuzione di un pulsante.	N.E	RA1F 5.2*
TS-RA1O 5.3	Viene verificato che l'utente possa eliminare il Document selezionato nella show-page.	E	RA1O 5.3
TS-RA1O 6.1	Viene verificato che l'admin possa creare un nuovo utente dalla pagina di amministrazione.	E	RA1O 6.1
TS-RA1O 6.1.1	Viene verificato che l'admin disponga di una pagina di creazione di un nuovo utente.	E	RA1O 6.1.1
TS-RA1O 6.1.1.1	Verificare che l'admin possa inserire l'indirizzo email del nuovo utente in un apposito campo di testo presente all'interno della pagina di creazione di un nuovo utente.	E	RA1O 6.1.1.1
TS-RA1O 6.1.1.2	Viene verificato che l'admin possa inserire la password del nuovo utente in un apposito campo di testo presente all'interno della pagina di creazione di un nuovo utente.	E	RA1O 6.1.1.2
TS-RA1O 6.1.1.3	Verificare che l'admin possa inserire il "livello utente" del nuovo utente tramite una combo-box presente all'in: terno della pagina di creazione di un nuovo utente.	E	RA1O 6.1.1.3

TS-RA1O 6.1.2	Viene verificato che l'applicazione pre: levi tutti i dati inseriti dall'admin nella pagina di creazione di un nuovo utente e li invii al database delle credenziali, il quale provvederà all'inserimento del nuovo record.	E	RA1O 6.1.2
TS-RA1O 6.1.3	Verificare che venga visualizzato un messaggio d'errore nel caso in cui l'admin non abbia compilato corret: tamente i campi presenti all'interno della pagina di creazione di un nuovo utente.	E	RA1O 6.1.3
TS-RA1O 6.2	Viene verificato che l'admin abbia la possibilità di selezionare un utente dalla index-page e visualizzare la sua relativa show-page.	E	RA1O 6.2
TS-RA1O 6.2.1	Verificare che l'admin possa elevare l'utente normale selezionato al livello "admin" dalla show-page relativa.	E	RA1O 6.2.1
TS-RA1O 6.2.2	Verificare che l'admin possa declassare l'admin selezionato a livello di utente normale dalla show-page relativa.	E	RA1O 6.2.2
TS-RA1O 6.2.3	Viene verificato che l'admin possa modificare l'attributo email dell'utente selezionato dalla relativa show-page.	E	RA1O 6.2.3
TS-RA1O 6.2.4	Verificare che l'admin possa modifi: care l'attributo password dell'utente selezionato dalla relativa show-page.	E	RA1O 6.2.4
TS-RA1O 6.2.5	Viene verificato che l'admin possa eliminare l'utente visualizzato nella *show-page*$_G$.	E	RA1O 6.2.5
TS-RF1O 7	Verificare che il linguaggio DSL all'in: terno di MaaP Framework sia stato implementato e sia funzionante.	E	RF1O 7
TS-RF1O 8.1	Verificare che Maap Framework generi automaticamente lo schele: tro dell'applicazione creata dallo sviluppatore.	E	RF1O 8.1

TS-RF1O 8.1.1	Verificare che Maap Framework importi automaticamente in un'apposita directory del progetto tutte le librerie necessarie al corretto funzionamento del sistema. Librerie necessarie: • Express v-3.4.8 • MongoDB v-1.3.23 • Mongoose v-3.8.4	E	RF1O 8.1.1
TS-RF1O 8.1.2	Verificare che Maap Framework crei automaticamente in un'apposita directory il file di configurazione di default dell'applicazione generata.	E	RF1O 8.1.2
TS-RF1O 8.1.3	Viene verificato che Maap Framework crei automaticamente il sistema di autenticazione per l'applicazione generata.	E	RF1O 8.1.3
TS-RF1O 8.1.4	Verificare che Maap Framework crei automaticamente le directory di descrizione delle pagine web.	E	RF1O 8.1.4
TS-RF1O 8.2	Verificare che Maap Framework crei automaticamente un account admin di default.	E	RF1O 8.2
TS-RF1F 8.3	Verificare che il framework MaaP permetta allo sviluppatore di definire un namespace per l'applicazione generata.	N.E	RF1F 8.3*
TS-RF1O 9.1	Verificare che il DSL permetta allo sviluppatore di creare una pagina Collection-index.	E	RF1O 9.1
TS-RF1O 9.1.1	Verificare che il DSL deve permettà allo sviluppatore di poter definire una serie di attributi da visualizzare all'interno della pagina Collection-index.	E	RF1O 9.1.1
TS-RF1O 9.1.2	Viene verificato che il DSL permetta allo sviluppatore di poter definire un ordinamento di default (ordine alfanumerico) di visualizzazione dei document all'interno della pagina Collection-index.	E	RF1O 9.1.2

TS-RF1O 9.1.3	Verificare che il DSL permetta al: lo sviluppatore di poter definire un eventuale limite di elementi da visualizzare all'interno della pagina Collection-index.	E	RF1O 9.1.3
TS-RF1O 9.1.4	Viene verificato che il DSL permetta allo sviluppatore di poter definire quali attributi sono ordinabili all'interno della pagina Collection-index.	E	RF1O 9.1.4
TS-RF1O 9.1.5	Verificare che il DSL permetta allo sviluppatore di definire la funzione populate per far si che una chiave riferisca ad un documento esterno.	E	RF1O 9.1.5
TS-RF1O 9.1.6	Verificare che il DSL permetta allo sviluppatore di definire delle query per creare la pagina Collection-index in base al risultato della loro estrazione.	E	RF1O 9.1.6
TS-RF1O 9.1.7	Viene verificato che il DSL permet: ta allo sviluppatore di definire del: le trasformazioni sugli attributi da visualizzare.	E	RF1O 9.1.7
TS-RF1O 9.2	Viene verificato che il DSL permetta allo sviluppatore di creare una pagina Collection-show.	E	RF1O 9.2
TS-RF1O 9.2.1	Verificare che il DSL permetta allo sviluppatore di definire una serie di attributi visualizzabili all'interno della pagina Collection-show.	E	RF1O 9.2.1
TS-RF1O 9.2.2	Verificare che il DSL permetta allo sviluppatore la definizione degli at: tributi del Document come attributi innestati o array di Document tramite la funzione populate.	E	RF1O 9.2.2
TS-RF1O 9.2.3	Verificare che lo sviluppatore abbia la possibilità di personalizzare la show page definendone l'ordinamento degli attributi.	E	RF1O 9.2.3
TS-RF1O 9.2.4	Viene verificato che lo sviluppato: re possa definire trasformazioni agli attributi per poi visualizzarli nella show-page.	E	RF1O 9.2.4

TS-RF1F 9.2.5	Verificare che lo sviluppatore possa personalizzare la show-page definendo delle operazioni personalizzate che l'u: tente potrà utilizzare tramite appositi pulsanti.	N.E	RF1F 9.2.5*
TS-RF1O 9.3	Viene verificato che il framework MaaP permetta allo sviluppatore di cambiare il nome della Collection da visualizzare nel menu di navigazione.	E	RF1O 9.3
TS-RF1O 9.4	Verificare che il framework MaaP permetta allo sviluppatore di modifi: care l'ordine di visualizzazione della Collection nel menu di navigazione.	E	RF1O 9.4
TS-RS1F 10.1	Verificare che il sistema MaaS per: metta allo sviluppatore di scrivere una Collection tramite editor di testo presente nella pagina web.	N.E	RS1F 10.1*
TS-RS1F 10.2	Verificare che il sistema MaaS per: metta all'utente di poter scrivere una Collection caricando un file prodotto dal framework MaaP.	N.E	RS1F 10.2*
TS-RS1F 10.3	Verificare che il sistema MaaS per: metta ad un utente non registrato di registrarsi al suo servizio.	N.E	RS1F 10.3*
TS-RS1F 10.4	Verificare che il sistema MaaS assegni automaticamente un $namespace_G$ sul sistema al nuovo utente registrato.	N.E	RS1F 10.4*
TS-RS1F 10.5	Verificare che il servizio MaaS visua: lizzi un messaggio d'errore nel caso in cui la registrazione fallisca a causa di credenziali già esistenti.	N.E	RS1F 10.5*
TS-RS1F 10.6	Verificare che il servizio MaaS metta a disposizione di un utente non autenti: cato la possibilità di effettuare il login al sistema.	N.E	RS1F 10.6*
TS-RS1F 10.7	Verificare che il servizio MaaS visua: lizzi un messaggio d'errore nel caso in cui l'utente non autenticato abbia inserito credenziali errate nel sistema di login.	N.E	RS1F 10.7*

TS-RS1F 10.8	Verificare che il sistema MaaS permetta ad un utente non autenticato di modificare il proprio profilo.	N.E	RS1F 10.8*
TS-RS1F 10.9	Verificare che il sistema MaaS permetta ad un utente non autenticato di eliminare il proprio account dal sistema.	N.E	RS1F 10.9*
TS-RS1F 10.9.1	Verificare che il sistema MaaS provveda all'eliminazione dei file di configurazione associati all'utente rimosso dal sistema.	N.E	RS1F 10.9.1*
TS-RS1F 10.10	Verificare che il sistema MaaS permetta allo sviluppatore di eliminare una Collection esistente.	N.E	RS1F 10.10*
TS-RA1D 13.1	Verificare che l'utente possa modificare la password di accesso all'applicazione.	E	RA1D 13.1
TS-RF1O 14.1	Verificare che il framework MaaP renda possibile la configurazione dei database delle credenziali.	E	RF1O 14.1
TS-RF1O 14.2	Verificare che il framework MaaP renda possibile la configurazione dei database delle Collection.	E	RF1O 14.2
TS-RF1F 14.3	Verificare che il framework MaaP renda possibile la selezione di un name-space per un database se la funzione di $namespace_G$ è abilitata.	N.E	RF1F 14.3*
TS-RA1F 15.1	Verificare che l'applicazione MaaP metta a disposizione dell'admin la visualizzazione degli indici in base alle query più richieste dall'applicazione.	N.E	RA1F 15.1*
TS-RA1F 15.2	Verificare che l'applicazione MaaP permetta all'admin di aggiungere gli indici in base ai suggerimenti forniti.	N.E	RA1F 15.2*
TS-RA1F 15.3	Verificare che l'applicazione MaaP permetta all'admin di rimuovere gli indici in base ai suggerimenti forniti.	N.E	RA1F 15.3*
TS-RS1F 17	Verificare che Il sistema MaaS si accerti che documenti creati rispettano i vincoli del database.	N.E	RS1F 17*

TS-RA1O 18	Verificare che il sistema metta a disposizione un validatore del codice DSL e visualizzi gli eventuali errori logici o di sintassi in un'apposita pagina.	E	RA1O 18
TS-RS1F 19	Verificare che il sistema MaaS salvi le pagine definite dagli utenti nel database e non su disco.	N.E	RS1F 19*

Tabella A.2: Tracciamento Test di Sistema - Requisiti

A.4 Test di integrazione

I test di integrazione vanno a controllare il corretto funzionamento delle componenti descritti dalla progettazione ad alto livello. Si è scelto di utilizzare un approccio *top-down*$_G$ ad eccezione del test TI 9 che viene eseguito con la metodologia *bottom-up*$_G$. Di seguito viene riportato un diagramma informale per chiarire l'albero dei test di integrazione. I test eseguiti sono indicati con una *E* (Eseguito), mentre i test non eseguiti sono indicati con *N.E* (Non Eseguito).

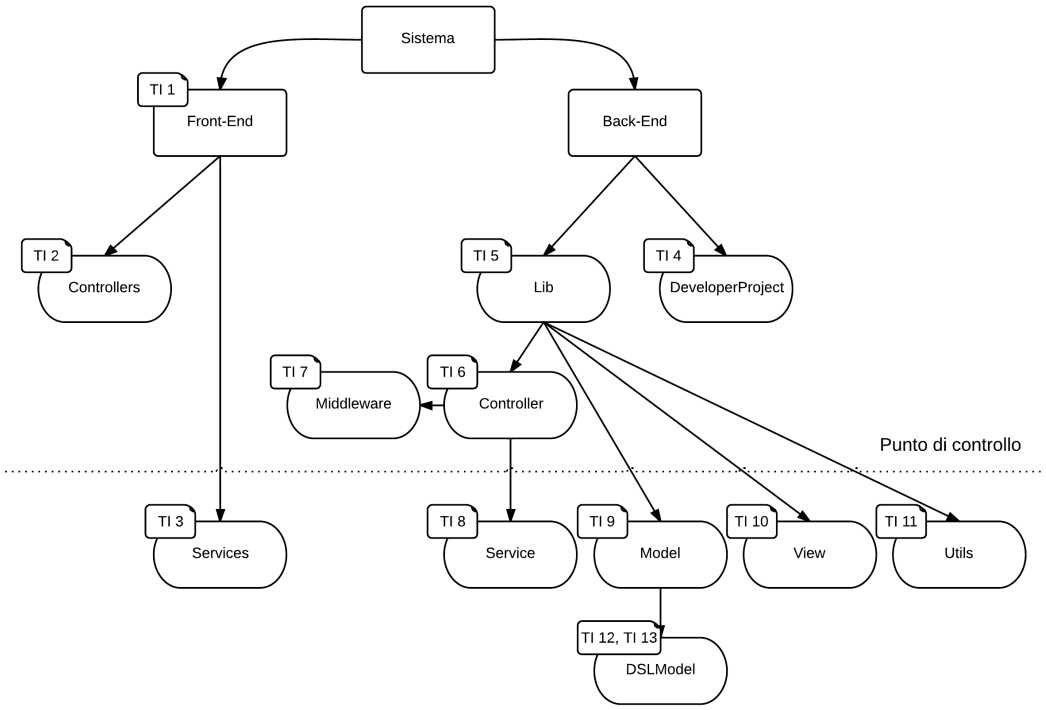

Figura 3: Sequenza d'integrazione delle componenti

Con la tecnica *top-down*$_G$ le componenti di più alto livello sono testate non appena sono implementate. Le componenti del sottosistema che non sono ancora state sviluppate, vengono simulate dagli *stub*$_G$. Man mano che si procede con la codifica delle componenti di più basso livello, queste vengono integrate e viene eseguito il relativo test. Grazie all'integrazione incrementale delle componenti del sistema, è più semplice determinare quale componente crea problemi e le funzioni di più alto livello sono testate prima.

La componente Front-end::Model non ha associato test d'integrazione poiché le classi di questo *package*$_G$ si prevede che non verranno codificate in quanto verrà sfruttato lo stile di *duck-typing*$_G$ della gestione dei tipi di *JavaScript*$_G$.

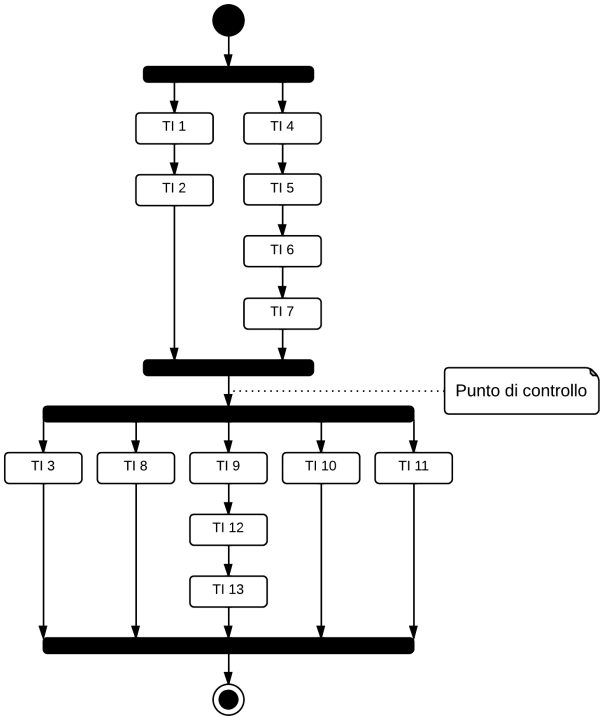

Figura 4: Diagramma di attività dei test

Test	**Descrizione**	**Componenti aggiunte**	**Stato**
TI 1	Si verifica che l'applicazione Web carichi correttamente le librerie JavaScript utilizzate.	Front-end	E.
TI 2	Si verifica che i controller si integrino correttamente nell'applicazione Web.	Front-end::Controllers	E.
TI 3	Si verifica che i services permettono di interagire correttamente con il back-end.	Front-end::Services	E.
TI 4	Si verifica che il DeveloperPro: ject avvii correttamente il ser: ver, fornendo in particolare i file statici del front-end.	Back-end::DeveloperProject	E.
TI 5	Si verifica che la libreria si in: tegri correttamente con il *Node Package Manager*$_G$ (npm) e che il suo script di installazione produca un DeveloperProject funzionante.	Back-end::Lib	E.

TI 6	Si verifica che i controller si integrino correttamente tra loro e nella gestione delle richieste che arrivano al server.	Back-end::Lib::Controller	E.
TI 7	Si verifica che il Middleware si integri correttamente nel: la gestione delle richieste che arrivano al server.	Back-end::Lib::Controller:: Middleware	E.
TI 8	Si verifica che il Service si inte: gri correttamente nella gestione delle richieste che arrivano al server.	Back-end::Lib::Controller:: Service	E.
TI 9	Si verifica che il Model si in: tegri correttamente della ge: stione dell'inserimento, della modifica, della creazione e del: l'eliminazione consistente dei dati.	Back-end::Lib::Model	E.
TI 10	Si verifica che la View si inte: gri correttamente con il Midd: leware per fornire i template come la gestione dell'invio delle mail.	Back-end::Lib::View	E.
TI 11	Si verifica che Utils si integri correttamen: te con il funzionamento dell'applicazione.	Back-end::Lib::Utils	E.
TI 12	Si verifica che le classi che compongono il DSLModel in: teragiscano correttamente tra loro.	Back-end::Lib::DSLModel	E.
TI 13	Si verifica che il DSLMo: del si integri correttamen: te con il funzionamento dell'applicazione.	Back-end::Lib::DSLModel	E.

Tabella A.3: Descrizione test d'Integrazione

A.5 Test di unità

Di seguito vengono elencati i testi di unità pianificati. I test che non sono stati implementati riportano lo stato "N.E".

Nome	Descrizione	Stato
TU - 2	Verifica che il service sia stato iniettato correttamente.	N.I.

TU - 4	Il costruttore ServerLoader viene invocato con alcuni oggetti di configurazione di tipo Config predefiniti. Si verifica che in ogni caso l'oggetto ServerLoader costruito sia effettivamente configurato con i parametri forniti in input.	Success
TU - 5	Viene verificato che un oggetto della classe, dati determinati input, venga costruito in modo corretto secondo quanto atteso.	Success
TU - 6	Viene verificato che il metodo restituisca l'errore in formato JSON atteso.	Success
TU - 7	Viene verificato che il metodo restituisca l'errore in formato stringa atteso.	Success
TU - 8	Viene verificato che il metodo restituisca l'errore in formato `Error` di *Node.js*$_G$ atteso.	Success
TU - 9	Verifica, iniettando un service, che lo scope venga popolato correttamente.	Success
TU - 10	Viene simulato uno scope tramite rootScope.new() e un service per testare che il controller gestisca correttamente l'invocazione dei metodi sul service per la modifica dell'utente e il popolamento dello scope.	Success
TU - 11	Verifica, iniettando un service, che il login venga effettuato quando i dati inseriti sono corretti e visualizzi correttamente l'errore altrimenti.	Success
TU - 12	Viene verificato che un oggetto della classe venga costruito correttamente secondo quanto atteso.	Success
TU - 13	Viene verificato che il metodo, dati determinati input, effettui una chiamata alla classe `Back-end::Lib::Model::DSLModel::DSLConcreteStrategy` e che quest'ultima restituisca tramite una callback un array di collections da inserire nel registro. Viene inoltre verificato che nel caso in cui venga passato in input il nome di un file non esistente il metodo generi un opportuno errore da restituire con una callback.	Success
TU - 14	Viene verificato che il metodo, dati determinati input, aggiunga correttamente il CollectionModel passatogli al registro dei modelli.	Success
TU - 15	Viene verificato che il metodo, dato l'id della collection, ne restituisca il model.	Success
TU - 16	Viene verificato che il metodo restituisca l'array di errori atteso.	Success

TU - 17	Viene verificato che il metodo inizializzi correttamente lo Schema mongoose degli utenti e renda disponibili i metodi attesi su di esso.	Success
TU - 18	Viene verificato che il metodo restituisca i dati degli utenti nel formato JSON atteso e gestisca gli eventuali errori di connessione a MongoDB.	Success
TU - 19	Viene verificato che il metodo, dato il suo input, registri correttamente l'utente nel database e gestisca nel modo atteso gli eventuali errori.	Success
TU - 20	Viene verificato che il metodo, dato il suo input, modifichi correttamente il livello dell'utente indicato e gestisca nel modo atteso gli eventuali errori.	Success
TU - 21	Viene verificato che il metodo, dati i suoi input, crei correttamente l'utente atteso sul database MongoDB degli utenti e gestisca gli eventuali errori generati.	Success
TU - 23	Viene verificato che il metodo, dati i suoi input, modifichi correttamente la password dell'utente indicato la nuova fornita e gestisca gli eventuali errori generati nella maniera attesa.	Success
TU - 24	Viene verificato che il metodo, dato un id, ricerchi l'utente indicato all'interno del database MongoDB degli utenti restituendo le informazioni associate e gestisca gli eventuali errori generati.	Success
TU - 25	Viene verificato che il metodo costruisca un oggetto della classe nel modo corretto e atteso.	Success
TU - 26	Viene verificato che il metodo inizializzi correttamente la classe e gestisca nel modo atteso gli eventuali errori generati.	Success
TU - 27	Viene verificato che il metodo, dati determinati input, carichi correttamente il file DSL, lo esegua in modo corretto e gestisca in modo atteso gli eventuali errori generati.	Success
TU - 28	Viene verificato che un oggetto della classe, dati determinati input, venga costruito correttamente e secondo le attese. Viene verificato inoltre che il metodo gestisca correttamente gli eventuali errori generati.	Success
TU - 29	Viene verificato che il metodo restituisca correttamente il nome della Collection dell'oggetto su cui viene invocato in formato stringa, secondo quanto atteso.	Success
TU - 30	Viene verificato che il metodo restituisca secondo quanto atteso l' `IndexModel` dell'oggetto su cui viene invocato.	Success

TU - 31	Viene verificato che il metodo restituisca secondo quanto atteso lo `ShowModel` dell'oggetto su cui viene invocato.	Success
TU - 32	Viene verificato che il metodo, dato il suo input, setti correttamente e in modo atteso il campo `indexModel` dell'oggetto su cui viene invocato.	Success
TU - 33	Viene verificato che il metodo, dato il suo input, setti correttamente e in modo atteso il campo `showModel` dell'oggetto su cui viene invocato.	Success
TU - 34	Viene verificato che un oggetto della classe venga costruito correttamente dato un certo input.	Success
TU - 35	Viene verificato che il metodo, dato il suo input, aggiunga in modo corretto e atteso l'attributo indicato all'array `attributes` dell'oggetto.	Success
TU - 36	Viene verificato che il metodo restituisca correttamente l'array `attributes` dell'oggetto su cui viene invocato e che quest'ultimo sia coerente rispetto a quanto atteso.	Success
TU - 37	Viene verificato che il metodo, dato il suo input, restituisca correttamente e in maniera coerente rispetto a quanto atteso la configurazione della *index-page* in formato JSON. Viene verificato inoltre che il metodo gestisca in modo corretto e atteso gli eventuali errori generati.	Success
TU - 38	Viene verificato che un oggetto della classe venga costruito correttamente dato un certo input.	Success
TU - 39	Viene verificato che il metodo, dato il suo input, aggiunga in modo corretto e atteso l'attributo indicato all'array `attributes` dell'oggetto.	Success
TU - 40	Viene verificato che il metodo restituisca correttamente l'array `attributes` dell'oggetto su cui viene invocato e che quest'ultimo sia coerente rispetto a quanto atteso.	Success
TU - 41	Viene verificato che il metodo, dato il suo input, restituisca correttamente e in maniera coerente rispetto a quanto atteso la configurazione della *show-page* in formato JSON. Viene verificato inoltre che il metodo gestisca in modo corretto e atteso gli eventuali errori generati.	Success
TU - 42	Viene verificato che un oggetto della classe venga costruito correttamente a partire da valori presi in input. Il test deve verificare inoltre che il metodo sia in grado di gestire gli eventuali errori generati dall'inserimento di un input scorretto.	Success

TU - 43	Viene verificato che il metodo restituisca correttamente il campo `label` dell'oggetto sul quale viene invocato e che quest'ultimo sia una stringa e sia coerente rispetto a quanto atteso.	Success
TU - 44	Viene verificato che il metodo restituisca correttamente il campo `name` dell'oggetto sul quale viene invocato e che quest'ultimo sia una stringa e sia coerente rispetto a quanto atteso.	Success
TU - 45	Viene verificato che il metodo restituisca correttamente il campo `transformation` dell'oggetto sul quale viene invocato e che quest'ultimo sia una *function* e sia coerente rispetto a quanto atteso.	Success
TU - 46	Viene verificato che il metodo restituisca correttamente il campo `selectable` dell'oggetto sul quale viene invocato e che quest'ultimo sia di tipo *Boolean* e sia coerente rispetto a quanto atteso.	Success
TU - 47	Viene verificato che il metodo restituisca correttamente il campo `sortable` dell'oggetto sul quale viene invocato e che quest'ultimo sia di tipo *Boolean* e sia coerente rispetto a quanto atteso.	Success
TU - 48	Viene verificato che il metodo comunichi correttamente con lo `UserModel` richiedendo l'eliminazione dell'utente ricevuto come parametro nella richiesta del server e che sappia gestire correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 49	Viene verificato che il metodo comunichi correttamente con lo `UserModel` richiedendo la registrazione dell'utente ricevuto come parametro nella richiesta del server e che sappia gestire correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 50	Viene verificato che il metodo comunichi correttamente con lo `UserModel` richiedendo la creazione dell'utente ricevuto come parametro nella richiesta del server e che sappia gestire correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 51	Viene verificato che il metodo comunichi correttamente con lo `UserModel` richiedendo i dati dell'utente ricevuto come parametro nella richiesta del server in formato JSON e che sappia gestire correttamente e in modo atteso gli eventuali errori generati.	Success

TU - 52	Viene verificato che il metodo comunichi correttamente con lo `UserModel` richiedendo la lista degli utenti presenti nel database MongoDB degli utenti in formato JSON e che sappia gestire correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 53	Viene verificato che il metodo comunichi correttamente con lo `UserModel` richiedendo la modifica del livello dell'utente ricevuto come parametro nella richiesta del server e che sappia gestire correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 54	Viene verificato che il metodo comunichi correttamente con la classe `DSLCollectionModel` e che ottenga cor: rettamente la configurazione della *index-page* in formato JSON secondo quanto atteso. Viene verificato inoltre che il metodo sia in grado di gestire correttamente gli eventuali errori generati.	Success
TU - 55	Viene verificato che venga costruito correttamente l'ogget: to, configurando il servizio di invio mail con i parametri impostati nella configurazione dell'applicazione passata come parametro.	Success
TU - 56	Viene verificato che il metodo re: stituisca un puntatore alla classe `Back-end::Lib::Controller::Controller::CollectionController` e che quest'ultimo non sia nullo.	Success
TU - 57	Viene verificato che il metodo re: stituisca un puntatore alla classe `Back-end::Lib::Controller::Controller::ProfileController` e che quest'ultimo non sia nullo.	Success
TU - 58	Viene verificato che il metodo re: stituisca un puntatore alla classe `Back-end::Lib::Controller::Controller::AuthController` e che quest'ultimo non sia nullo.	Success
TU - 59	Viene verificato che il metodo re: stituisca un puntatore alla classe `Back-end::Lib::Controller::Controller::ForgotController` e che quest'ultimo non sia nullo.	Success
TU - 60	Viene verificato che il metodo re: stituisca un puntatore alla classe `Back-end::Lib::Controller::Controller::UserController` e che quest'ultimo non sia nullo.	Success

TU - 61	Viene verificato che il metodo restituisca un puntatore alla classe `Back-end::Lib::Controller::Controller::ShowController` e che quest'ultimo non sia nullo.	Success
TU - 62	Viene verificato che il metodo restituisca un puntatore alla classe `Back-end::Lib::Controller::Controller::IndexController` e che quest'ultimo non sia nullo.	Success
TU - 63	Viene verificato che un oggetto della classe venga costruito in modo corretto e secondo le attese.	Success
TU - 64	Viene verificato che il metodo, dato il suo input, invochi correttamente il metodo `browseFileSystem` andando a cercare tutti i file DSL e successivamente invochi correttamente il metodo di caricamento dei file DSL, andando a costruire quindi il `DSLModel`. Viene verificato inoltre che il metodo gestisca correttamente gli eventuali errori generati dalle chiamate alle varie funzioni.	Success
TU - 66	Viene verificato che il metodo, dato il suo input, restituisca correttamente e in modo atteso l'array di file presenti ne path indicato tramite una callback e sappia gestire in modo corretto e atteso gli eventuali errori generati.	Success
TU - 67	Viene verificato che il metodo, dato il suo input (che sarà una richiesta del server), si interfacci correttamente con la classe `ShowModel` e restituisca dunque al server la configurazione della show-page attesa in formato JSON. Viene verificato inoltre che il metodo sappia gestire correttamente gli eventuali errori generati.	Success
TU - 67	Viene verificato che il metodo configuri correttamente la gestione delle uri specificate nella sezione "Interfaccia REST" della *Specifica Tecnica v4.0.0*. Per far questo, verrà passato come parametro app un oggetto fittizio, i cui metodi conterranno il codice necessario a verificare che vengano configurate tutte e sole le uri della specifica, associandole ai giusti controller.	Success
TU - 68	Viene verificato che il metodo inserisca nell'oggetto di risposta res gli errori nel formato JSON generati dal parametro err, impostando il corretto codice HTTP di errore.	Success

TU - 69	Viene verificato che il metodo inserisca nell'oggetto di risposta res i dati attesi, cioè l'errore nel formato JSON che segnala al client che la richiesta ricevuta richiede una risorsa che non è stata trovata. Deve anche essere impostando il corretto codice HTTP di errore.	Success
TU - 71	Viene verificato che il metodo, dato il suo input (che sarà una richiesta dal server), si interfacci correttamente con la class `ShowModel`, la quale si occuperà di eliminare il Document indicato. Viene verificato inoltre che il metodo sappia gestire gli eventuali errori generati.	Success
TU - 72	Viene verificato che il metodo, dato il suo input (che sarà una richiesta del server), reindirizzi correttamente l'utente alla Dashboard dell'applicazione.	Success
TU - 73	Viene verificato che il metodo, dato il suo input (che sarà una richiesta del server), distrugga correttamente la sessione dell'utente indicato e reindirizzi l'utente alla pagina di login. Viene verificato inoltre che il metodo sappia gestire correttamente gli eventuali errori generati.	Success
TU - 74	Viene verificato che il metodo, dato il suo input (che sarà una richiesta del server), si interfacci correttamente con la classe `UserModel` e restituisca dunque correttamente e in modo atteso al server i dati dell'utente richiesto in formato JSON. Viene verificato inoltre che il metodo sappia gestire correttamente gli eventuali errori generati.	Success
TU - 75	Viene verificato che il metodo, dato il suo input (che sarà una richiesta del server), si interfacci correttamente con la classe `UserModel` ed effettui correttamente l'update della nuova password dell'utente indicato, secondo quanto atteso. Viene verificato inoltre che il metodo sappia gestire correttamente gli eventuali errori generati.	Success
TU - 76	Viene verificato che il metodo restituisca correttamente al server un errore 404.	Success
TU - 77	Viene verificato che il metodo, dato il suo input, generi correttamente e in modo atteso il token di reset password e invii correttamente un'email all'indirizzo indicato. Viene verificato inoltre che il metodo sappia gestire in modo corretto gli eventuali errori generati.	Success
TU - 78	Viene verificato che il metodo, dato i suoi input, restituisca correttamente e in modo atteso l'array dei file presenti nella root indicata tramite una callback. Viene verificato inoltre che il metodo sappia gestire correttamente gli eventuali errori generati.	Success

TU - 79	Viene verificato che il metodo verifichi se l'utente auten: ticato ha un livello admin e che gestisca correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 80	Viene verificato che il metodo verifichi se l'utente è autenticato e che gestisca correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 81	Viene verificato che il metodo verifichi se l'utente è autenticato e che gestisca correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 82	Viene simulato un backend tramite httpBackend per testare che il service richieda e riceva in modo corretto la risorsa user.	Success
TU - 82	Viene verificato che il metodo verifichi se l'utente ha livello di super admin e che gestisca correttamente e in modo atteso gli eventuali errori generati.	Success
TU - 83	Viene simulato un backend tramite httpBackend per testare che il service richieda in modo corretto la modifica di una risorsa user.	Success
TU - 84	Viene simulato un backend tramite httpBackend per testare che il service richieda in modo corretto la modifica di una risorsa user.	Success
TU - 85	Viene simulato un backend tramite httpBackend per testare che il service richieda e riceva in modo corretto la risorsa document richiesta.	Success
TU - 86	Viene simulato un backend tramite httpBackend per testare che il service richieda e riceva in modo corretto le risorse document di una collection.	Success
TU - 87	Viene simulato un backend tramite httpBackend per testare che il service richieda e riceva in modo corretto le collection presenti.	Success
TU - 88	Viene simulato un backend tramite httpBackend per testare che il service richieda in modo corretto la creazione di una risorsa user.	Success
TU - 89	Viene simulato un backend tramite httpBackend per testa: re che il service richieda in modo corretto l'eliminazione di una risorsa user.	Success
TU - 90	Viene simulato un backend tramite httpBackend per testare che il service richieda e riceva in modo corretto le risorse user.	Success

TU - 91	Viene simulato un backend tramite httpBackend per testare che il service richieda in modo corretto la modifica della risorsa user.	Success
TU - 92	Viene simulato un backend tramite httpBackend per testare che il service richieda e riceva in modo corretto la risorsa user. (Dell'user loggato)	Success
TU - 93	Viene simulato uno scope tramite rootScope.new() e un service per testare che il controller gestisca correttamente il prelievo dati dallo scope e l'invocazione dei metodi sul service.	Success
TU - 94	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 95	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 96	Viene simulato uno scope tramite rootScope.new() e un service per testare che il controller gestisca correttamente l'invocazione dei metodi sul service per la cancellazione e l'aggiornamento dello scope.	Success
TU - 97	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 98	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success

TU - 99	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 100	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 101	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 102	Viene simulato uno scope tramite rootScope.new() e un service per testare che il controller venga costruito correttamente.	Success
TU - 103	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 104	Si verifica che il controller venga costruito correttamente.	Success
TU - 105	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 106	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success

TU - 107	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo. Questo test verrà eseguito per tanti valori predefiniti d input e output.	Success
TU - 108	Viene verificato che il metodo, a partire dai parametri in input, costruisca e restituisca un email nel formato Email di NodeMailer. Di questo oggetto Email si controlla che il valore di tutti i campi dati coincidano con i valori attesi.	Success
TU - 108	Viene simulato un backend tramite httpBackend per testare che il service modifichi in modo corretto i campi del document richiesto.	Success
TU - 109	Viene simulato un backend tramite httpBackend per testare che il service elimini in modo corretto la risorsa document.	Success
TU - 110	Si verifica che il controller popoli correttamente i campi dello scope.Lo scope e i service vengono forniti al metodo come stub, in particolare lo scope viene utilizzato per fornire l'output e i service per dare l'input al metodo.	Success
TU - 111	Viene simulato un backend tramite httpBackend per testare che il service richieda il login in modo corretto.	Success
TU - 112	Viene simulato un backend tramite httpBackend per testare che il service richieda il logout in modo corretto.	Success
TU - 113	Si verifica che la classe venga costruita correttamente.	Success
TU - 114	Viene simulato un backend tramite httpBackend per testare che il service invii in modo corretto i dati necessari.	Success
TU - 115	Viene simulato un backend tramite httpBackend per testare che il service invii la nuova password correttamente.	Success
TU - 116	Viene simulato un backend tramite httpBackend per testare che il service resetta in modo corretto la password utente.	Success
TU - 117	Viene simulato uno scope tramite rootScope.new() e un service per testare che il controller gestisca correttamente l'invocazione dei metodi sul service per l'ordinamento dei documenti.	Success

TU - 118	Viene simulato uno scope tramite rootScope.new() e un service per testare che il controller gestisca correttamente l'invocazione dei metodi sul service per la paginazione dei documenti.	Success
TU - 119	Viene verificato che il metodo dato un input, ritorni correttamente il nome di tutte le collection come atteso e viene verificato inoltre che sia in grado di gestire eventuali errori.	N.I.
TU - 120	Viene verificato che il metodo, dato come input una richiesta dal server, si interfacci correttamente con la class `ShowModel`, la quale si occuperà di effettuare la modifica del Document indicato in maniera conforme. Viene verificato inoltre che il metodo sappia gestire gli eventuali errori generati.	N.I.
TU - 121	Viene verificato che il metodo, dati come input il token e la nuova password, sia in grado di trovare l'utente a cui appartiene il token passato e di effettuare correttamente il reset della password. Viene verificato inoltre che sia in grado di gestire eventuali errori in maniera corretta.	N.I.
TU - 122	Viene verificato che dato come input un errore di tipo MaapError questo venga registrato correttamente nel registro.	Success
TU - 123	Viene verificato che il metodo presi come input due oggetti di tipo `DSLCollectionModel` li compari restituendo il giusto output secondo le specifiche.	N.I.
TU - 124	Viene verificato che il metodo restituisca l'array conte: nente oggetti di tipo `DslCollectionModel` ordinato in base al peso che ogni oggetto ha, verificando inoltre che gestisca correttamente gli errori.	N.I.
TU - 125	Viene verificato che il metodo componga in maniera corretta la funzione mongoose di estrazione paginata dei Document e viene verificato che restituisca il riferimento alla query in maniera attesa al chiamante gestendo correttamente eventuali errori avvenuti nell'elaborazione.	N.I.
TU - 126	Viene verificato che dato un input, il metodo restituisca correttamente un document a cui è stata applicata la funzione *populate* di mongoose.	N.I.
TU - 127	Viene verificato che il metodo dato come input l'id di un Document restituisca quest'ultimo.	N.I.
TU - 128	Viene verificato che il metodo dato come input un id di un Document effettui correttamente l'eliminazione di quest'ultimo dal database.	N.I.

TU - 129	Questo metodo si occupa di effettuare l'update del Document indicato. Verifica che il metodo dato, con input le modifiche da apportare al documento e l'id di quest'ultimo, lo modifichi correttamente e gestisca in modo atteso gli eventuali errori.	N.I.
TU - 130	Viene verificato che il metodo riesca ad ottenere correttamente la lista degli utenti e restituire la lista con il numero di utenti corretto rispetto all'input dato, corrispondenti alla pagina indicata.	N.I.
TU - 131	Viene verificato che dato un input, questo metodo sia in grado di eliminare l'utente indicato gestendo gli errori in maniera attesa.	N.I.
TU - 132	Viene verificato che dato come input un'email, il metodo restituisca l'id corrispondente.	N.I.
TU - 133	Viene verificato che il metodo generi un token corretto e un tempo di vita conforme alle attese, salvando queste informazioni e gestendo eventuali errori in maniera corretta.	N.I.
TU - 134	Viene verificato che dato un input, il metodo restituisca il token atteso, verificando inoltre che quest'ultimo dopo esser stato restituito al chiamante, venga invalidato.	N.I.
TU - 135	Viene verificato che il metodo, dato in input un token e una password, rilevi se il token è valido e trovi l'utente a cui appartiene effettuando il reset della password con la password data in input. Viene verificato inoltre che rilevi correttamente un token non valido e restituisca errore.	Success
TU - 136	Viene verificato che dato come input un token, il metodo restituisca l'utente a cui corrispondente. Viene verificato inoltre che gestisca gli errori correttamente.	N.I.
TU - 137	Viene verificato che data una colonna questa venga aggiunta correttamente all'array delle colonne.	N.I.
TU - 138	Viene verificato che questo metodo invochi in maniera aspettata il metodo `setDefaultColumnSelectable`.	N.I.
TU - 139	Viene verificato che dato un input, il metodo si occupi di impostare la colonna _id o la prima colonna disponibile come selectable se non ne è stata definita una.	N.I.
TU - 140	Verifica che il metodo verifichi correttamente se un array di colonne è vuoto e se lo è si occupi di estrarre tutti gli attributi di un Document della Collection e di creare e aggiungere colonne a partire da essi.	N.I.

TU - 141	Viene verificato che il metodo trasformi un Document in formato JSON in maniera corretta.	N.I.
TU - 142	Viene verificato che il metodo restituisca in maniera aspettata il campo `id` della classe.	Success
TU - 143	Viene verificato che il metodo restituisca in maniera corretta secondo input dati, il campo `label` della classe.	Success
TU - 144	Viene verificato che il metodo restituisca il campo `weight` della classe in maniera attesa secondo gli input dati.	Success
TU - 145	Viene verificato che il metodo richiami il me: todo `getCollectionName` e restituisca il campo `collectionName` della classe.	Success
TU - 146	Viene verificato se dato in input un Document, il metodo verifica correttamente se l'array degli attributi è vuoto, se questo avviene deve occuparsi di inserire nell'array tutti gli attributi del Document.	N.I.
TU - 147	Viene verificato che il metodo restituisce correttamente un JSON il cui contenuto sono gli attributi del Document. Viene inoltre verificata la corretta gestione in caso di errori.	N.I.
TU - 148	Viene verificato che dato come input un id relativo ad un Document, il metodo lo elimini correttamente dal database.	N.I.
TU - 149	Viene verificato che dati i corretti input il metodo riesca ad effettuare l'update del Document in maniera attesa. Viene inoltre verificato che in caso di eventuali errori questo risponda correttamente.	N.I.
TU - 150	Viene verificato che il metodo restituisca correttamente il campo *label* della classe.	Success
TU - 151	Viene verificato che il metodo restituisca il campo `name` della classe in modo atteso.	Success
TU - 152	Viene verificato che il metodo restituisca il corretto campo `transformation` della classe.	Success
TU - 153	Viene verificato che il metodo restituisca il il campo `name` della classe.	N.I.
TU - 154	Viene verificato che l'oggetto venga costruito correttamente.	Success
TU - 155	Verifica che il metodo ritorna correttamente il campo `name` della classe.	Success

TU - 156	Viene verificato che il metodo imposti in maniera attesa il campo `selectable` della classe.	Success
TU - 157	Viene verificato che dato in input un oggetto, il metodo applichi correttamente la trasformazione e la restituisca al chiamante. Viene verificato che il metodo sia in grado di gestire eventuali errori in maniera corretta.	Success

Tabella A.4: Test di Unità

B Resoconto delle attività di verifica

B.1 Riassunto delle attività di verifica

B.1.1 Revisione dei Requisiti

L'attività di verifica svolta dai *Verificatori* è avvenuta come determinato dal *Piano di Progetto v5.0.0* al termine della stesura di ogni documento previsto. La verifica svolta sui documenti è avvenuta seguendo le indicazioni delle *Norme di Progetto v5.0.0* e misurando le metriche indicate in 2.8.2. L'attività di *walkthrough* ha evidenziato una serie di anomalie, in questo modo è stato possibile stilare la lista di anomalie frequenti (vedi *Norme di Progetto v5.0.0*) che si potranno controllare tramite *Inspection*. Successivamente si è proceduto con le misurazioni delle metriche relative ai documenti. In questa revisione non è stato possibile valutare i processi poiché lo stato embrionale del team e impegni universitari sovrapposti non hanno permesso il rilevamento accurato di tutti i parametri necessari. Il gruppo ha in programma di colmare tale mancanza per la revisione successiva.

B.1.2 Revisione di Progettazione

L'attività di verifica svolta dai *Verificatori* è avvenuta come determinato dal *Piano di Progetto v5.0.0* al termine della stesura di ogni documento previsto. La verifica svolta sui documenti è avvenuta seguendo le indicazioni delle *Norme di Progetto v5.0.0* e misurando le metriche indicate in 2.8.2. Successivamente si è proceduto con le misurazioni delle metriche relative ai documenti. Sono quindi state misurate le metriche sui processi per valutarne la bontà e fornire una base per la pianificazione dei cicli $PDCA_G$.

B.1.3 Revisione di Qualifica

L'attività di verifica svolta dai *Verificatori* è avvenuta come determinato dal *Piano di Progetto v5.0.0* al termine della stesura di ogni documento previsto. La verifica svolta sui documenti è avvenuta seguendo le indicazioni delle *Norme di Progetto v5.0.0* e misurando le metriche indicate in 2.8.2. Le anomalie evidenziate non incidono in modo determinante sulla

consistenza del prodotto del processo di documentazione. Le metriche hanno contribuito al controllo sui processi permettendo di monitorare e misurare il loro andamento.

B.1.4 Revisione di Accettazione

L'attività di verifica svolta dai *Verificatori* è avvenuta come determinato dal *Piano di Progetto v5.0.0* al termine degli incrementi di ogni documento. La verifica svolta sui documenti è avvenuta seguendo le indicazioni delle *Norme di Progetto v5.0.0* e misurando le metriche indicate in 2.8.2. La verifica ha coinvolto principalmente le correzioni segnalate dal proponente a seguito della revisione di qualifica.

B.2 Miglioramenti post Revisione

B.2.1 Miglioramenti post RR

A seguito delle attività di verifica e controllo è stato sottoposto un questionario ad ogni membro del gruppo che ha contribuito ad identificare le problematiche relative ai processi e a formulare proposte risolutive. Da queste idee sono nate diverse modifiche e miglioramenti ai documenti e in generale al nostro modo di lavorare. Seguendo questa linea abbiamo applicato coerentemente la politica di *plan-do-check-act*, utilissima per il miglioramento della qualità:

Problema	Possibile soluzione	Stato
Il dizionario personale di $Aspell_G$, essendo un file collaborativo compilato in automatico da tale $tool_G$, impone molto spesso attività manuali extra di gestione del $repository_G$, in particolare vanno risolti molti conflitti.	Uno script che ordina il file in questione dovrebbe diminuire i conflitti.	Eseguito.
Contrassegnare le parole di glossario con il relativo tag_G è un attività fortemente propensa a dimenticanze ed errori.	Uno script potrebbe contrassegnare le parole di glossario presenti nei documenti in automatico.	Non eseguito.
Scarsa frammentazione dei $task_G$	Incremento dell'utilizzo dello strumento di gestione dei processi.	Eseguito.
Mantenere la corrispondenza tra casi d'uso e relativi url_G dei diagrammi è un operazione lunga e manuale.	Uno script può automatizzare tale attività.	Eseguito.
Difficoltà nell'uso dello strumento di controllo di versione.	I membri del gruppo devono eseguire attività extra di autoformazione sulla base del materiale messo a disposizione da alcuni membri.	Eseguito.

Tabella A.5: Problemi individuati in RR e relative soluzioni.

B.2.2 Miglioramenti post RP

A seguito delle attività di verifica e controllo è stato sottoposto un questionario ad ogni membro del gruppo che ha contribuito ad identificare le problematiche relative ai processi e a formulare proposte risolutive. Da queste idee sono nate diverse modifiche e miglioramenti ai documenti e in generale al nostro modo di lavorare. Seguendo questa linea abbiamo applicato coerentemente la politica di *plan-do-check-act*, utilissima per il miglioramento della qualità:

Problema	Possibile soluzione	Stato
Forte sbilanciamento tra i membri del gruppo in merito alle conoscenze dello *stack tecnologico*$_G$ utilizzato.	Autoformazione teorica e pratica.	Eseguito.
Forte sbilanciamento in merito alla capacità di effettuare scelte di design architetturale, con la conseguente necessità di un'interazione ripetitiva tra i membri del gruppo.	Autoformazione.	Eseguito.
Difficoltà nell'apportare modifiche allo strumento *Requisteak*$_G$ (descritto in *Norme di Progetto v5.0.0*) dovute al linguaggio di sviluppo (*Ruby on Rails*$_G$) conosciuto solo da una parte dei membri del gruppo.	Utilizzo sistematico delle issues di *GitHub*$_G$ per lo strumento in questione.	Eseguito.
Sviluppo di *Requisteak*$_G$ (descritto in *Norme di Progetto v5.0.0*) ritardatario per mancanza di progettazione e tempi ristretti.	Per la progettazione è necessario interagire con i docenti per capire al meglio le associazioni tra quanto progettato ed i relativi test.	Eseguito.
Difficoltà nel approcciare con un documento al quale non si è lavorato in precedenza.	Va predisposta una maggiore sistematicità nel commentare il materiale prodotto, per la Specifica Tecnica sono necessari commenti per i diagrammi e le scelte architetturali.	Eseguito.
Segnalazioni delle anomalie nel repository dei documenti troppo informali e non tracciabili.	Utilizzo delle issues per il *repository*$_G$ dei documenti e relativa autoformazione per il loro corretto utilizzo.	Eseguito.

Tabella A.6: Problemi individuati in RP e relative soluzioni.

B.2.3 Miglioramenti post RQ

A seguito della revisione di qualifica è stato sottoposto, ad ogni membro del gruppo, un questionario da compilare con lo scopo di identificare quali problematiche hanno influito con l'avanzamento del progetto didattico.

Problema	Possibile soluzione	Stato
Difficoltà nell'utilizzo delle funzionalità della piattaforma GitHub.com	Autoformazione	Eseguito.
I file sorgente risultano non essere formattati secondo lo stile prescelto.	L'utilizzo del plugin *JsFormat* per *Sublime Text* permette di formattare il codice in modo automatico e in linea con quan: to specificato nelle *Norme di Progetto v5.0.0*	Eseguito.

Tabella A.7: Problemi individuati in RQ e relative soluzioni.

B.3 Dettaglio delle verifiche tramite analisi

B.3.1 Analisi

B.3.1.1 Documenti Vengono qui riportati i valori dell'indice Gulpease per ogni documento durante l'analisi e relativo esito basato sui range stabiliti in 2.8.2.1.

Documento	Valore indice	Esito
Analisi dei Requisiti v1.3.1	45	superato
Glossario v1.3.1	63	superato
Norme di Progetto v1.3.1	47	superato
Piano di Progetto v1.3.1	51	superato
Piano di Qualifica v1.3.1	53	superato
Studio di Fattibilità v1.3.1	43	superato

Tabella A.8: Esiti verifica documenti, Analisi

B.3.2 Progettazione Architetturale

B.3.2.1 Documenti Vengono qui riportati i valori dell'indice Gulpease per ogni documento durante la progettazione architetturale e relativo esito basato sui range stabiliti in 2.8.2.1.

Documento	Valore indice	Esito
Analisi dei Requisiti v3.0.0	51	superato
Glossario v3.0.0	49	superato
Norme di Progetto v3.0.0	47	superato
Piano di Progetto v3.0.0	73	superato
Piano di Qualifica v3.0.0	50	superato
Studio di Fattibilità v3.0.0	44	superato
Specifica Tecnica v3.0.0	58	superato

Tabella A.9: Esiti verifica documenti, Progettazione Architetturale

B.3.3 Progettazione di dettaglio e codifica

B.3.3.1 Documenti Vengono qui riportati i valori dell'indice Gulpease per ogni documento durante la progettazione di dettaglio e codifica, e relativo esito basato sui range stabiliti in 2.8.2.1.

Documento	Valore indice	Esito
Analisi dei Requisiti v4.0.0	54	superato
Definizione di Prodotto v3.0.0	42	superato
Glossario v5.0.0	49	superato
Manuale Admin v3.0.0	51	superato
Manuale Sviluppatore v3.0.0	43	superato
Manuale Utente v3.0.0	50	superato
Norme di Progetto v5.0.0	53	superato
Piano di Progetto v5.0.0	52	superato
Piano di Qualifica v5.0.0	51	superato
Specifica Tecnica v4.0.0	76	superato

Tabella A.10: Esiti verifica documenti, Progettazione di Dettaglio e Codifica

B.3.3.2 Codice Vengono qui riportate le misure rilevate con le metriche sull'analisi statica e dinamica del codice. Per ogni metrica si riportano i valore calcolati mantenendo una separazione tra backend e frontend. Per una descrizione delle metriche si rimanda alla sezione 2.8.3.

Complessità ciclomatica

- *Backend*:
  - medio: 1.38
  - massimo: 6
- *Frontend*:
  - medio: 1.34
  - massimo: 5

Numero di metodi

- *Backend*: 155
- *Frontend*: 62

Numero parametri per metodo

- *Backend*:
  - medio: 1.68
  - massimo: 5

- *Frontend*:
  - medio: 1.39
  - massimo: 2

$Halstead_G$ difficulty per-function

- *Backend*:
  - medio: 3.42
  - massimo: 16.90
- *Frontend*:
  - medio: 3.46
  - massimo: 21.36

$Halstead_G$ volume per-function

- *Backend*:
  - medio: 100.76
  - massimo: 1303.56
- *Frontend*:
  - medio: 102.23
  - massimo: 671.55

$Halstead_G$ effort per-function

- *Backend*:
  - medio: 653.64
  - massimo: 9778.01
- *Frontend*:
  - medio: 572.79
  - massimo: 14342.28

Maintainability index

- *Backend*:
  - medio: 77.05
  - minimo: 56.57
- *Frontend*:
  - medio: 71.98

– minimo: 50.2

Statement Coverage

- *Backend*: 63.45%
- *Frontend*: 66.2%

I valori non raggiungono i parametri di accettazione. Tale mancanza verrà colmata nel prossimo periodo di sviluppo.

Branch Coverage

- *Backend*: 52.8%
- *Frontend*: 48.69%

I valori non raggiungono i parametri di accettazione. Tale mancanza verrà colmata nel prossimo periodo di sviluppo.

B.3.4 Validazione

B.3.4.1 Documenti Vengono qui riportati i valori dell'indice Gulpease per ogni documento durante la validazione, e relativo esito basato sui range stabiliti in 2.8.2.1.

Documento	Valore indice	Esito
Analisi dei Requisiti v4.0.0	54	superato
Definizione di Prodotto v3.0.0	62	superato
Glossario v5.0.0	49	superato
Manuale Admin v3.0.0	51	superato
Manuale Sviluppatore v3.0.0	42	superato
Manuale Utente v3.0.0	49	superato
Norme di Progetto v5.0.0	67	superato
Piano di Progetto v5.0.0	55	superato
Piano di Qualifica v5.0.0	50	superato
Specifica Tecnica v4.0.0	76	superato

Tabella A.11: Esiti verifica documenti, Validazione

B.3.4.2 Codice Vengono qui riportate le misure rilevate con le metriche sull'analisi statica e dinamica del codice. Per ogni metrica si riportano i valore calcolati mantenendo una separazione tra backend e frontend. Per una descrizione delle metriche si rimanda alla sezione 2.8.3.

Complessità ciclomatica

- *Backend*:
  – medio: 1.47

  - massimo: 7
- *Frontend*:
  - medio: 1.51
  - massimo: 8

Numero di metodi

- *Backend*: 215
- *Frontend*: 105

Numero parametri per metodo

- *Backend*:
  - medio: 1.59
  - massimo: 5
- *Frontend*:
  - medio: 1.4
  - massimo: 7

***Halstead*$_G$ difficulty per-function**

- *Backend*:
  - medio: 3.6
  - massimo: 20
- *Frontend*:
  - medio: 3.2
  - massimo: 15

***Halstead*$_G$ volume per-function**

- *Backend*:
  - medio: 105
  - massimo: 1531
- *Frontend*:
  - medio: 106
  - massimo: 1218

$Halstead_G$ effort per-function

- *Backend*:
  - medio: 844
  - massimo: 5317
- *Frontend*:
  - medio: 635
  - massimo: 2019

Maintainability index

- *Backend*:
  - medio: 77
  - minimo: 53
- *Frontend*:
  - medio: 73.9
  - minimo: 50.14

Statement Coverage

- *Backend*: 50.56%

Branch Coverage

- *Backend*: 26.45%

I motivi per cui si misurano valori così bassi sono causati dalla difficoltà tecnica di scrivere dei test di unità su cui sia possibile eseguire un branch coverage automatico con le librerie disponibili online: in particolari casi la nostra implementazione degli stub, sebbene esegua correttamente il test, non viene registrata dallo strumento di code coverage. Per il front end in particolare, è stato riscontrato un bug nella libreria di Angular che impedisce di eseguire la funzione inject all'interno dell'ambiente di test coverage. Tale bug è stato risolto soltanto nella versione instabile di Angular (si veda https://github.com/angular/angular.js/commit/7e916455b36dc9ca4d4afc1e44cade90006d00e3).

B.4 Dettaglio dell'esito delle revisioni

Lo sviluppo di questo progetto didattico si basa sull'attraversamento di quattro revisioni presiedute dal committente. Tre delle quattro revisioni produrranno delle segnalazioni degli errori riscontrati da parte del committente, deve seguire un report di come sono state risolte in ogni documento.

B.4.1 Revisione dei Requisiti

Per la Revisione dei Requisiti le segnalazioni da parte del committente sono state corrette:

- *Norme di Progetto*: il documento è stato riorganizzato per processi, attività, procedure, strumenti. Sono state aggiunte indicazioni sugli strumenti per la gestione del *repository*$_G$ e le regole per la rotazione dei ruoli sono state definite in modo dettagliato;
- *Analisi dei Requisiti*: le *Norme di Progetto v5.0.0* descrivono la modalità di consegna che è stata ben definita che include la generazione dei nomi dei documenti con la relativa versione. Inoltre sono stati rivisti tutti i requisiti e casi d'uso segnalati dal committente;
- *Piano di Progetto*: l'Organigramma è stato spostato in appendice e sono stati rimossi i costi orari dei ruoli. Sono state ripartizionate le ore considerando attività di analisi successive al 2013-12-20 e la percentuale di ore di verifica è almeno del 30% del totale;
- *Piano di Qualifica*: la trattazione del *SEMAT*$_G$ è stata spostata ed approfondita nel piano di progetto e le tecniche adottate sono state spostate nelle norme di progetto;
- *Glossario*: è stato creato l'indice del documento e ogni gruppo di lettera inizia su una pagina nuova.

B.4.2 Revisione di Progettazione

Per la Revisione di Progettazione le segnalazioni da parte del committente sono state corrette:

- *Norme di Progetto*: la lista dei riferimenti del documento è stata sistemata. L'organizzazione del documento per processi è stata rivista nuovamente ponendo come riferimento ISO/IEC 12207-2008 su cui si basano i punti che descrivono un processo e le tipologie;
- *Specifica Tecnica*: i *framework*$_G$ utilizzati sono stati descritti in modo più approfondito e sono stati corretti i diagrammi di sequenza segnalati;
- *Definizione di Prodotto*: sono stati corretti i dettagli progettuali segnalati;
- *Manuali*: il glossario viene presentato all'interno del documento stesso;
- *Piano di Progetto*: rivista la struttura in modo da presentare in modo più leggibile la parte di preventivo/consuntivo.

B.4.3 Revisione di Qualifica

Per la Revisione di Qualifica le segnalazioni da parte del committente sono state corrette:

- *Norme di Progetto*: il documento è stato nuovamente riorganizzato ponendo come principale riferimento lo standard ISO/IEC 27000, i processi primari sono stati rivisiti come processo di sviluppo.
- *Specifica Tecnica*: le descrizioni dei *framework*$_G$ utilizzati sono state approfondite. Sono stati corretti i diagrammi di sequenza e attività indicati.
- *Definizione di Prodotto*: sono stati inseriti i diagrammi delle classi richiesti. Le didascalie sono state corrette. Corretti numerosi errori ortografici.

- *Manuali*: il glossario di ogni documento è ora collocato in appendice. Corretti errori ortografici e termini da glossario non segnati come tali. Integrazione con numerose immagini che guidano l'utilizzatore nelle procedure da svolgere.
- *Piano di Progetto*: riorganizzati i contenuti per migliorare la lettura del documento.

C 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. La qualità va ricercata non sul prodotto bensì sui **processi alla base del prodotto**, per questo tutti i processi seguono il metodo $PDCA_G$ che prevede l'iterazione ripetuta tra i quattro stadi, assicurando un **incremento della qualità** ad ogni ciclo.

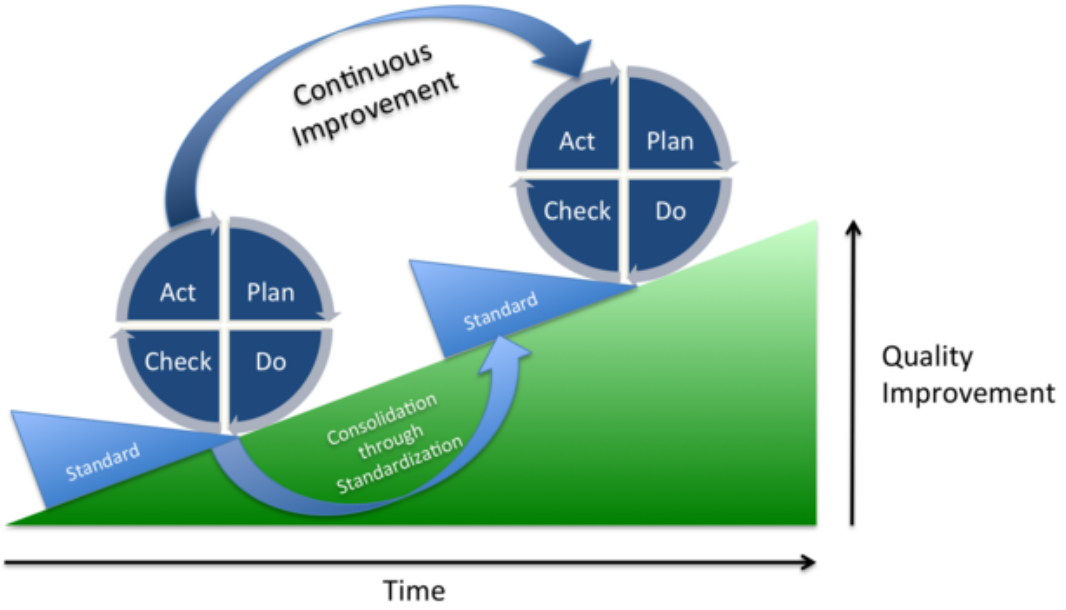

Figura 5: Continuous quality improvement with PDCA

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;
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 2.8.1.1) 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.

Il ciclo $PDCA_G$ è stato attuato nell'appendice D.

C.1 Qualità dei processi

Definita in ISO/IEC 15504 come $SPICE_G$, specifica come la qualità è collegata alla maturazione dei processi. Vengono individuati dei livelli di maturità al quale il fornitore può fare riferimento per determinare le proprie capacità organizzative. Vengono definiti:

- Dei **Modelli di riferimento** su:
  - *dimensione del processo*;
  - *livelli di capacità 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 2.8.1, in particolare la *Schedule Variance* permette di capire se un processo è incompleto o gestito; il gruppo giungerà a maturazione quando i processi diventeranno predicibili ossia quando la *Schedule Variance* subirà al più lievi oscillazioni;
- Delle **Stime** che si concretizzano in una struttura per la misurazione composta da:
  - I *processi* di misurazione, indicati nel *Piano di Progetto v5.0.0*;
  - Un *modello* per la misurazione identificabile in questo documento;
  - Gli *strumenti* utilizzati, specificati nelle *Norme di Progetto v5.0.0*.
- Le **Competenze e Qualifiche** di chi controlla; lo standard redige in modo rigoroso una serie di attività volte a formare chi opera l'attività di stesura del *Piano di Qualifica* e *Verifica*. 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.

C.2 Qualità del prodotto software

Specificata in ISO/IEC 9126 si suddivide in:

- **Quality model**: classifica la qualità del software in un set di caratteristiche che verranno approfondite nel corso del progetto:
  - Functionality: viene controllata grazie al tracciamento dei requisiti individuati ed analizzati e i componenti;
  - Reliability: viene dimostrata combinando i test;
  - Usability: viene controllata con i test di validazione, inoltre la stesura del manuale d'uso aiuterà a verificarne l'usabilità e ad intervenire laddove necessario;
  - Efficiency: combinando analisi statica e dinamica controlliamo che il prodotto sia efficiente;
  - Maintainability: viene realizzata con l'utilizzo di design pattern e la stesura di documentazione dettagliata;
  - Portability: essendo $MaaP_G$ un applicazione Web non ci sono particolari problemi di portabilità per gli utenti.
- **External metrics**: sono le metriche rilevate tramite analisi dinamica specificate in 2.8;
- **Internal metrics**: sono le metriche rilevate in analisi statica specificate in 2.8;
- **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*.

D PDCA

In questo capitolo verrà descritto come è stato applicato il modello $PDCA_G$ descritto nel *Piano di Qualifica v5.0.0*.

D.1 Revisione dei requisiti

In questo periodo è stata svolta un attività di *walkthrough* non avendo i dati necessari per effettuare attività di *inspection*, come descritto nel *Piano di Qualifica v5.0.0*. Gli errori frequenti rilevati sono visionabili nelle *Norme di Progetto v5.0.0*.

Non è stato possibile eseguire nessun ciclo $PDCA_G$ in mancanza di misurazioni sui processi, non avendo quindi modo di pianificare processi per la qualità, ma è stato studiato e descritto nel *Piano di Qualifica v5.0.0* e verrà attuato dalla prossima $milestone_G$.

D.2 Revisione di progettazione

PLAN Al fine di valutare su quali processi pianificare dei miglioramenti sono state eseguite diverse misurazioni utilizzando le metriche per i processi descritte nel *Piano di Qualifica v5.0.0*.

I risultati ottenuti sono riportati nella seguente tabella:

Metrica	Valore indice	Esito
Produttività di documentazione	199	Superato
Impegno	0,71	Superato
Modifiche	23	Non superato

Tabella A.12: Risultati metriche per i processi, Revisione di progettazione

Analizzando tali dati si è deciso di pianificare le seguenti attività per il miglioramento della qualità dei processi:

Un numero troppo elevato di modifiche incide pesantemente sulla produttività. È necessario decrementare tale valore al fine di aumentare la produttività e di conseguenza diminuire i costi. Questo primo ciclo $PDCA_G$ si prefigge dunque l'obiettivo di portare entro un range di accettazione[10] il numero di modifiche approvate.

Probabilmente un numero elevato di modifiche è causato dall'inesperienza del gruppo nel primo periodo, e ragionevolmente con l'aumentare delle conoscenze il numero di modifiche andrà calando di conseguenza.

In ogni caso, si pianifica di:

- Frammentare maggiormente i task assegnati in sotto-task;
- Specificare in modo esteso cosa prevede ogni singolo sotto-task, escludendo quindi dubbi che poi porteranno a successive richieste di modifica;

[10] Vedi *Piano di Qualifica v5.0.0*

- Creare la label "Domanda" nella sezione $issue_G$ di $GitHub_G$, per permettere la richiesta di delucidazioni sullo svolgimento di $task_G$ assegnati.

CHECK Al fine di valutare se le azioni pianificate hanno portato ad un miglioramento dei processi sono state eseguite le necessarie misurazioni.

I risultati ottenuti sono riportati nella seguente tabella:

Metrica	**Valore indice**	**Esito**
Produttività di documentazione	103	Superato
Impegno	2,69	Superato
Modifiche	18	Superato

Tabella A.13: Risultati metriche per i processi, Revisione di progettazione

Gli obiettivi posti nello stadio di pianificazione sono stati soddisfatti, si passerà dunque allo stadio di standardizzazione delle soluzioni applicate.

D.3 Revisione di qualifica

PLAN La revisione di qualifica prevede la stesura della Definizione di Prodotto, la manualistica del prodotto e la relativa codifica. La codifica è una parte corposa e ci aspettiamo una variazione visibile degli indici misurati.

L'autoformazione svolta fin'ora deve essere testata per verificarne il livello. Per comprendere se il gruppo ha raggiunto le competenze necessarie per affrontare la progettazione di dettaglio e la codifica, e, dove necessario, incrementarle, si pianifica di codificare un prototipo usa e getta assegnando ad ogni componente del gruppo una componente del sistema. Non ci si aspetta che il prodotto derivante da queste attività sia conforme alle attese finali, ma che tutti i membri del gruppo seguano le norme sulla codifica stabilite nelle *Norme di Progetto v5.0.0*.

Viene introdotta in questa revisione un'analisi delle issue di *GitHub*$_G$ tramite un grafico di *burndown*$_G$. Viene rappresentato il carico di lavoro in relazione al tempo considerando sia il consuntivo sia il backlog, permettendo di confrontare l'andamento reale con quello atteso a fine milestone.

CHECK La codifica del prototipo ha evidenziato lacune nella formazione sullo stack tecnologico sul quale si appoggia il prodotto. Contemporaneamente, ha contribuito a colmare le stesse lacune e a formare il team in direzione di un processo di codifica collaborativo, elemento mancante in molti membri, e coerente con le norme specificate.

In Figura 6 viene riportato il grafico di *burndown*$_G$ che si era pianificato di monitorare.

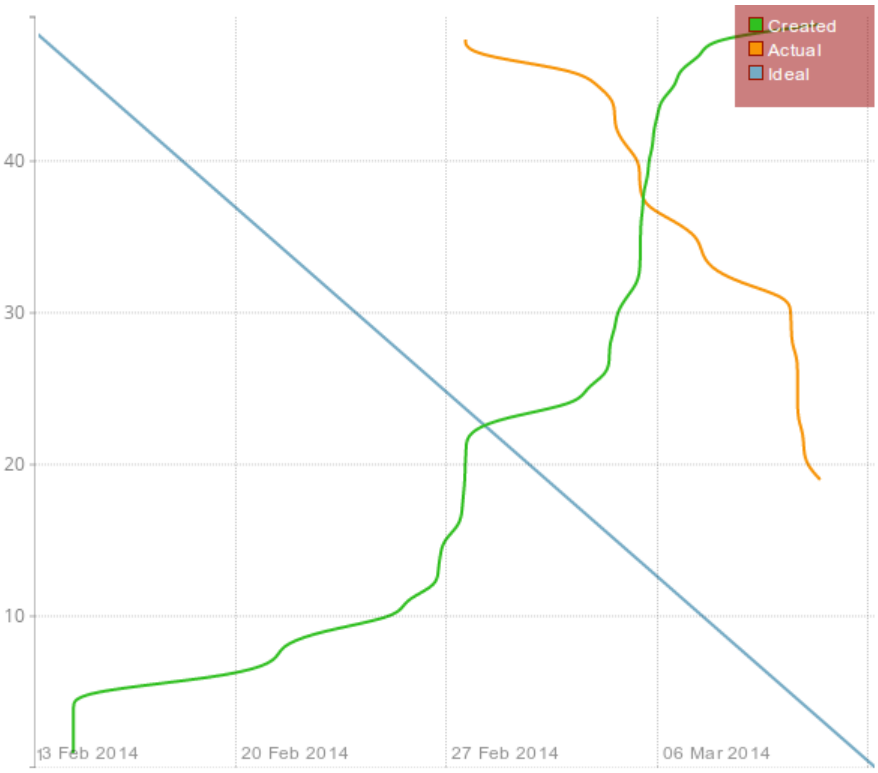

Figura 6: Burndown RQ

L'esplosione del numero di issue iniziale era atteso poiché le correzioni ricevute dai committenti sono state scomposte in parti atomiche per poterle assegnare ai componenti del gruppo in maniera chiara e tracciabile. Tale grafico si è rivelato utile per comprendere maggiormente l'andamento di sviluppo e accelerare i tempi quando necessario.

D.4 Revisione di accettazione

PLAN La revisione di accettazione prevede la chiusura della documentazione e la consegna del relativo software prodotto. Il software va testato approfonditamente in modo scrupoloso. La qualità del software dovrà migliorare andando ad intervenire laddove gli indici metrici segnalino valori degni di attenzione. Il collaudo verificherà le funzionalità del sistema, permetterà di scovare e segnalare buona parte delle anomalie, in questo modo si arriverà in sede di revisione di accettazione con un prodotto sufficientemente pronto ad una dimostrazione.

CHECK La validazione ha evidenziato alcune anomalie nel software, alcune segnalate dal proponente stesso. La correzione ha riguardato per lo più piccole funzionalità software.

In Figura 7 viene riportato il grafico di $burndown_G$ che si era pianificato di monitorare. Da notare il fatto che alcune issues per la revisione di accettazione sono state create prima della

revisione di qualifica, questo è dovuto ad una attenta pianificazione durante la codifica che ha individuato in tempo le issues da svolgere.

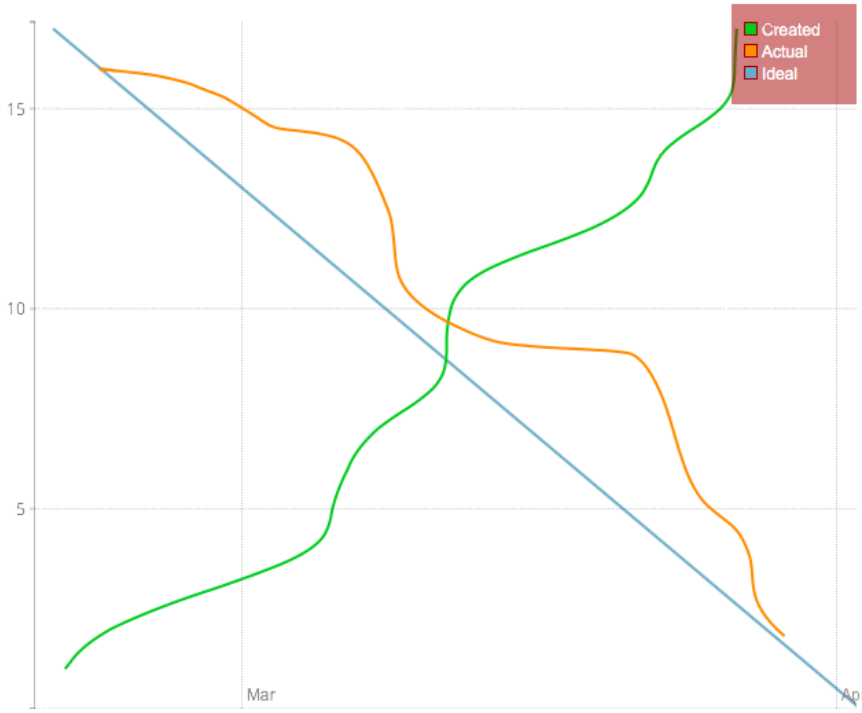

Figura 7: Burndown RA

E Use Case Points

La metrica Use Case Points è stata applicata al fine di valutare l'attuabilità dello sviluppo di quali e quanti requisiti prima dell'accettazione.

Segue un breve studio di fattibilità basato sui risultati derivanti dall'applicazione degli Use Case Points.

E.1 Fattori tecnici dell'implementazione

I 13 fattori tecnici valutati sono:

1. **Distributed System Required**: l'architettura della soluzione può essere centralizzata o single-tenant, o può essere distribuita (come una soluzione n-tier) o multi-tenant. Numeri più alti rappresentano un'architettura più complessa;
2. **Response Time Is Important**: la rapidità di risposta per gli utenti è un fattore importante (e non banale), tranne nel caso in cui il carico del server dovesse essere molto basso. Numeri alti rappresentano un'alta importanza del tempo di risposta;
3. **End User Efficiency**: l'applicazione è sviluppata per migliorare l'efficienza d'uso, o semplicemente la capacità? Numeri alti rappresentano progetti che si basano più pesantemente sull'applicazione per migliorare l'efficienza d'uso;
4. **Complex Internal Processing Required**: ci sono molti algoritmi complessi da implementare e testare? Algoritmi complessi hanno numeri alti. Delle semplici query al database dovrebbero avere numeri bassi;
5. **Reusable Code Must Be a Focus**: il riuso massivo del codice è un obiettivo? Maggiore è il riuso del codice, minore è il valore relativo;
6. **Installation Ease**: un'installazione agevole è un fattore chiave? Maggiore è il livello di competenza degli utenti, minore è il valore relativo;
7. **Usability**: la facilità di utilizzo è un criterio primario per l'accettazione? Maggiore è l'importanza dell'usabilità, maggiore è il valore relativo;
8. **Cross-Platform Support**: è richiesto il supporto multi-piattaforma? Più piattaforme devono essere supportate, più alto è il valore relativo;
9. **Easy To Change**: il fornitore richiede di poter cambiare o personalizzare l'applicazione in futuro? Più richieste sono mosse in questa direzione, maggiore è il valore relativo;
10. **Highly Concurrent**: bisogna preoccuparsi di questioni di concorrenza? Più attenzione bisogna prestare a risolvere conflitti su dati o sull'applicazione, maggiore è il valore relativo;
11. **Custom Security**: è possibile utilizzare soluzioni già esistenti riguardanti la sicurezza o è necessario svilupparne di personali? Maggiori soluzioni personalizzate si devono implementare, più alto è il valore relativo;
12. **Dependence on Third Party Code**: l'applicazione necessita dell'uso di librerie o altro software di terze parti? Più codice di terze parti (e più affidabile è), minore è li valore relativo;

13. **User Training**: quanta formazione dell'utente è richiesta? L'applicazione è complessa o supporta attività complesse? Più tempo l'utente impiega nella formazione, maggiore è il valore relativo.

Per ogni fattore è necessario attribuire un valore da 0 a 5 che costituisce l'importanza relativa.

Il Technical Complexity Factor è calcolato sommando le importanze relative moltiplicate per il peso corrispondente, diviso per 100 e sommato a 0,6.

Fattori tecnici		**Peso**	**Importanza relativa**	
			Con tutti i requisiti	**Solo requisiti acc.**
1	Distributed System Required	2	**4**	**0**
2	Response Time Is Important	1	**1**	**1**
3	End User Efficiency	1	**3**	**3**
4	Complex Internal Processing Required	1	**1**	**1**
5	Reusable Code Must Be A Focus	1	**2**	**2**
6	Installation Ease	0.5	**1**	**1**
7	Usability	0.5	**3**	**3**
8	Cross-Platform Support	2	**4**	**4**
9	Easy To Change	1	**1**	**1**
10	Highly Concurrent	1	**1**	**1**
11	Custom Security	1	**2**	**2**
12	Dependence On Third-Party Code	1	**2**	**2**
13	User Training	1	**3**	**2**
Technical Complexity Factor			**0.94**	**0.85**

Tabella A.14: UCP - Fattori tecnici

I fattori che variano sono *Distributed System Required* poiché non verrà implementato $MaaS_G$ e *User Training* in quanto la formazione degli utenti dovrà trattare di una minore mole di materiale.

E.2 Fattori ambientali

Gli 8 fattori ambientali valutati sono:

1. **Familiarity With The Project**: quanta esperienza ha maturato il team nel dominio di sviluppo? Un alto livello di esperienza corrisponde ad un numero alto;
2. **Application Experience**: quanta esperienza ha maturato il team con l'applicazione? Questo è rilevante solo quando si attuano cambiamenti a applicazioni già esistenti. Un valore alto corrisponde a molta esperienza;
3. **Object Oriented Programming Experience**: quanta esperienza ha maturato il team verso l'Object Oriented? Un valore alto corrisponde a molta esperienza nell'Object Oriented;
4. **Lead Analyst Capability**: quanto è esperto e abile la persona responsabile dei requisiti? Un valore alto rappresenta abilità e esperienza;

5. **Motivation**: quanto è motivato il team? Un valore alto corrisponde a molta motivazione;
6. **Stable Requirements**: cambiamenti nei requisiti comportano un incremento della mole di lavoro. Un valore alto corrisponde a molti cambiamenti;
7. **Part Time Staff**: un valore alto riflette un team part time, consulenti esterni e sviluppatori che si occupano contemporaneamente di più progetti. Un cambiamento di contesto è causa di inefficienza;
8. **Difficult Programming Language**: linguaggi di programmazione complessi corrispon: dono a numeri elevati. Questo valore è da attribuire in relazione alle capacità del team e non in senso assoluto.

Per ogni fattore è necessario attribuire un valore da 0 a 5 che costituisce l'importanza relativa.

Ponendo S la somma delle importanze relative moltiplicate per il peso corrispondente, l'Environmental Factor viene così calcolato:

$$EF = 1.4 - (0.03 \cdot S)$$

Fattori ambientali		**Peso**	**Importanza relativa**	
			Con tutti i requisiti	**Solo requisiti acc.**
1	Familiarity With The Project	1.5	**0**	**0**
2	Application Experience	0.5	**0**	**0**
3	OO Programming Experience	1	**1**	**1**
4	Lead Analyst Capability	0.5	**1**	**1**
5	Motivation	1	**5**	**5**
6	Stable Requirements	2	**2**	**3**
7	Part Time Staff	-1	**3**	**3**
8	Difficult Programming Language	-1	**5**	**5**
Environmental Factor			**1.325**	**1.265**

Tabella A.15: UCP - Fattori ambientali

L'unico fattore che varia è *Stable Requirements* poiché un maggior numero di requisiti porta ad una maggiore probabilità di cambiamento degli stessi.

E.3 Quantità e complessità degli use case

Ad ogni use case valutato viene attribuito un numero di *use case points* in base al suo grado di complessità, calcolato sul numero di transazioni, inteso come scambio, una risposta del sistema ad un'azione dell'utente:

- **Simple**: fino a 3 transazioni;
- **Average**: da 4 a 7 transazioni;
- **Complex**: più di 7 transazioni.

Il calcolo degli Unadjusted Use Case Points è uguale alla somma delle importanze relative moltiplicate per il peso corrispondente.

Use Case Points grezzi		**Peso**	**Numero di use case**	
			Con tutti i requisiti	**Solo requisiti acc.**
1	Simple	5	**21**	**15**
2	Average	10	**1**	**1**
3	Complex	15	**0**	**0**
Unadjusted Use Case Points			**115**	**85**

Tabella A.16: UCP - Quantità e complessità degli use case

Il tracciamento dei requisiti riportato in *Analisi dei Requisiti v4.0.0* permette di non considerare i casi d'uso che sono legati a requisiti non accettati.

E.4 Quantità e complessità degli attori

Ad ogni attore valutato viene attribuito un peso in base al suo grado di complessità:

- **Simple**: sono altri sistemi che comunicano con il proprio software attraverso delle API_G predefinite. L'elemento chiave è che si sta esponendo un'interazione con il proprio software attraverso un meccanismo specifico e ben definito;
- **Average**: possono essere sia uomini che interagiscono con un protocollo ben definito, sia sistemi che utilizzano API_G più complesse o flessibili;
- **Complex**: utenti che interagiscono con il sistema in modo imprevedibile anche attraverso interfacce grafiche.

Il calcolo dell'Actor Complexity Factor è uguale alla somma delle importanze relative moltiplicate per il peso corrispondente.

Attori		**Peso**	**Number of Actors**	
			Con tutti i requisiti	**Solo requisiti acc.**
1	Simple	1	**0**	**0**
2	Average	2	**1**	**1**
3	Complex	3	**6**	**4**
Actor Complexity Factor			**20**	**14**

Tabella A.17: UCP - Quantità e complessità degli attori

Gli attori implicati nell'interazione con il sistema $MaaS_G$ non vengono calcolati. Il fattore di complessità risulta diminuire sensibilmente considerando solo i requisiti accettati poiché gli utenti $MaaS_G$ sono ritenuti *complessi*.

E.5 Risultati e conclusioni

Ponendo

- Technical Complexity Factor = TCF;
- Environmental Factor = EF;
- Unadjusted Use Case Points = $UUCP$;
- Actor Complexity Factor = ACF;
- Use Case Points = UCP;
- Ore di impegno per Use Case Point = $HUCP$;

il calcolo delle ore di impegno sarà

$$Ore = [(UUCP + ACF) \cdot TCF \cdot EF] \cdot HUCP$$

Fattori di complessità			
TCF	Technical Complexity Factor	0.94	0.85
EF	Environmental Factor	1.325	1.265
UUCP	Unadjusted Use Case Points	115	85
ACF	Actor Complexity Factor	20	14
Calcolo Use Case Points			
UCP	Use Case Points	168.1	106.4
Calcolo ore di sviluppo			
Rapporto	Ore di impegno per Use Case Point	7	7
	Ore di impegno	**1,177**	**745**

Tabella A.18: UCP - Ore di impegno stimate

I risultati riportati in tabella A.18 mettono in evidenza come le ore necessarie per lo sviluppo di tutti i requisiti richieda uno sforzo quantificato in ore non sostenibile per il gruppo, e che eccede eccessivamente il limite superiore delle 105 ore produttive per ciascun componente del gruppo. Considerando solo i requisiti accettati (vedi *Analisi dei Requisiti v4.0.0*), si nota come le ore di impegno scendano ad un livello molto vicino alle attese. Dal momento che la quantificazione risultante dalla tecnica utilizzata è una stima, è plausibile ritenere accettabile tale livello e procedere verso la direzione scelta con l'accettazione dei requisiti.