PIANO DI QUALIFICA

Versione 0.0.1 in data 17-12-2017

Gruppo 353 - Progetto Marvin

Informazioni sul documento

Responsabili	Parwinder Singh
Redazione	Elena Mattiazzo
	Gianluca Marraffa
	Valentina Marcon
Verifica	
Stato	In corso
Uso	Esterno
Destinato a	Red Babel
	Gruppo 353
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
Email di contatto	353swe@gmail.com

Diario delle modifiche

Versione	Data	Descrizione	Autore	Ruolo
0.0.4	04-01-2018	Stesura specifica dei test	Gianluca Marraffa	Amministratore
0.0.2	27-12-2017	Stesura qualità di prodotto	Elena Mattiazzo	Amministratore
0.0.1	6-12-2017	Creazione scheletro del documento	Riccardo E. Giorato	Amministratore

Indice

Elenco delle figure

Elenco delle tabelle

1. Introduzione

1.1 Scopo del documento

Lo scopo di questo documento consiste nel documentare le norme utilizzate dal Gruppo *353*adottate per la verifica e la validazione dei prodotti e dei processi. Per ottenere lo scopo proposto, i processi attuati e i prodotti realizzati saranno continuamente verificati, affinché non vengano introdotti errori che influiscano negativamente sul risultato finale tramite strategie e metriche qui descritte.

1.2 Scopo del prodotto

Lo scopo del prodotto è quello di realizzare una piattaforma web chiamata *Marvin* che simuli le funzionalità di base per studenti, docenti e università di Uniweb. L'applicativo al posto del database dovrà utilizzare la rete Ethereum interagendo con degli smart contract.

1.3 Glossario

All'interno del documento sono presenti termini che presentano significati ambigui a seconda del contesto. Per evitare questa ambiguità è stato creato un documento di nome Glossario che conterrà tali termini con il loro significati specifico. Per segnalare che un termine del testo è presente all'interno del *Glossario v 1.0.0* verrà aggiunta una G a pedice a fianco del termine.

1.4 Riferimenti

1.4.1 Riferimenti Normativi

- ***Norme di progetto v 1.0.0***;

- **Standard ISO/IEC 9126**
  https://it.wikipedia.org/wiki/ISO/IEC_9126
  - Modello di qualità.

1.4.2 Riferimenti Informativi

- **Verifica e validazione: introduzione - Slide del corso di Ingegneria del Software**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L17.pdf
- **Indice di Gulpese**
  https://it.m.wikipedia.org/wiki/Indice_Gulpease
  - Descrizione e formula di calcolo.
- **Formula di Flesch**
  https://it.m.wikipedia.org/wiki/Formula_di_Flesch
  - Descrizione e formula di calcolo.
- **Qualità del software - Slide del corso di Ingegneria del Software**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L13.pdf
- **Qualità di processo - Slide del corso di Ingegneria del Software**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L15.pdf
- **Processi SW - Slide del corso di Ingegneria del Software**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L03.pdf
- **ISO/IEC 15504 - Pagina Wikipedia**
  https://en.wikipedia.org/wiki/ISO/IEC_15504
- **The Art of Software Testing, third edition, chapter 6: High-Order Testing**
  http://shop.oreilly.com/product/9781118031964.do
- **64 Test Metrics For Measuring Progress, Quality, Productivity & More**
  https://www.qasymphony.com/blog/64-test-metrics/

2. Qualità di processo

2.1 Scopo

Per garantire la qualità del prodotto è necessario perseguire la qualità dei processi che lo definiscono. Per raggiungere questo obiettivo, si è deciso di seguire il principio di miglioramento continuo (PDCA) e di adottare lo standard ISO/IEC 15504 denominato SPICE (Software Process Improvement and Capability Determination).

2.2 Procedure di controllo di qualità di processo

La qualità dei processi verrà garantita dall'applicazione del principio PDCA, descritto dell'appendice A. Grazie a questo principio, sarà possibile ottenere un miglioramento continuo della qualità di tutti i processi, inclusa la verifica, e come diretta conseguenza si otterrà il miglioramento dei prodotti risultanti.

Per ottenere qualità dei processi, bisogna:

- Definire il processo: affinché sia controllabile;
- Controllare il processo: in funzione dell'ottenimento di efficacia, efficienza ed esperienza;
- Usare buoni strumenti di valutazione: SPICE e PDCA;

2.3 Processi

2.3.1 Pianificazione di progetto, impostazione e controllo processi

Il macro-processo ha lo scopo di produrre dei piani di sviluppo per il progetto, comprendenti la scelta del modello di ciclo di vita del prodotto, descrizioni delle

attività e dei compiti da svolgere, pianificazione temporale del lavoro e dei costi da sostenere, allocazione di compiti e responsabilità, e misurazioni per rilevare lo stato del progetto rispetto alle pianificazioni prodotte.

Obiettivi

Lo sviluppo del progetto dovrà porre particolare attenzione a rispettare dei particolari obiettivi:

- **Budget:** utilizzando le metriche descritte nella sezione seguente, tenere sempre controllato l'utilizzo del budget, al fine di non avere scarti eccessivi con il costo preventivato;
- **Task:** porre attenzione alla pianificazione delle task e al loro completamento, assicurandosi che seguano il principio di miglioramento continuo, affinché nessun compito non venga migliorato;
- **Educazione personale:** avere accortezza che ogni membro del gruppo abbia un livello di preparazione adatto all'esecuzione dei task assegnati, al fine di evitare ritardi sul calendario;
- **Calendario:** assicurare una pianificazione adatta ai compiti da svolgere, per evitare le condizioni di sforamento del budget;
- **Standard:** definire standard di processo ogni qualvolta sia possibile, per facilitare il lavoro del gruppo e favorire un incremento continuo.

Strategie

Ogni eventuale valore negativo a livello di Schedule o Budget Variance rilevato sarà compensato con la revisione delle attività da svolgere e i requisiti da ottenere, per valutare se nei tempi di calendario stabiliti la pianificazione sia corretta o se sia necessario rivedere la programmazione. Utilizzare gli strumenti come Asana e i diagrammi di Gantt per verificare l'andamento del progetto, per avere sempre una visione chiara e quantificabile del lavoro in corso affinché il lavoro non subisca ritardi. Porre sempre delle finestre di slack, per evitare sovrapposizioni di task dovute a ritardi.

Metriche

MPS001 Schedule Variance (SV)

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

È un indicatore di efficacia soprattutto nei confronti del Cliente.
Se SV è positivo, significa che il progetto sta producendo con maggior velocità rispetto a quanto pianificato, viceversa se negativo.
Misurazione:

$$SV = BCWP–BCWS$$

Dove

- BCWP (Budgeted Cost of Work Performed) è il valore (in giorni o Euro) delle attività realizzate alla data corrente. Rappresenta il valore prodotto dal progetto ossia il valore dei deliverable rilasciati fino al momento della misurazione in seguito alle attività svolte.
- BCWS (Budgeted Cost of Work Scheduled) è il costo pianificato (in giorni o Euro) per realizzare le attività di progetto alla data corrente.

MPS002 Budget Variance (BV)
Indica se alla data corrente si è speso di più o di meno rispetto a quanto previsto a budget alla data corrente.
È un indicatore che ha un valore unicamente contabile e finanziario.
Se BV è positivo significa che il progetto sta spendendo il proprio budget con minor velocità di quanto pianificato, viceversa se negativo.

Misurazione:

$$BV = BCWS–ACWP$$

Dove

- BCWS (Budgeted Cost of Work Scheduled) è il costo pianificato (in giorni o Euro) per realizzare le attività di progetto alla data corrente.
- ACWP (Actual Cost of Work Performed) è il costo effettivamente sostenuto (in giorni o Euro) alla data corrente.

2.3.2 Verifica software

Il processo punta a verificare se qualsiasi elemento del sistema soddisfa completamente i requisiti ad esso correlati.

Obiettivi

Per poter definire delle baseline per lo sviluppo del software, è necessario che il codice venga sempre verificato.

- **Commenti al codice:** ogni pezzo di codice dovrà essere commentato affinché sia ritenuto verificabile;
- **Prevenzione di bug:** fare in modo che ogni pezzo di codice non sia affetto da bug prima dell'utilizzo.

Strategie

Per prevenire bug e vulnerabilità al codice, si utilizzano strumenti come Sonarlint e Sonarqube al fine di evitare la propagazione di errori e avere una panoramica sullo stato generale del codice prodotto. Utilizzare delle metriche di Code Coverage per avere consapevolezza della quantità di codice testato e poter agire di conseguenza.

Metriche

Code Coverage Per avere una misura di codice testato e verificato, si utilizzano dei coverage criteria:

- **MPS003 Code coverage:** verificare che ogni funzione sia stata chiamata;
- **MPS004 Statement coverage:** verificare che ogni statement del codice sia stato eseguito;
- **MPS005 Branch coverage:** verificare se tutti i possibili branch (derivanti da if e case statement) sono stati eseguiti;
- **MPS006 Condition coverage:** verificare se ogni condizione booleana è stata valutata sia nella condizione true che false.

Misurazione: Viene calcolato in percentuale la quantità di codice testato e verificato sul totale delle linee di codice scritte.

2.4 Tabella delle metriche

Questa tabella indica i **range** di accettazione e di ottimalità per le metriche utilizzate per la qualità di processo:

ID	Nome	Range di accettazione	Range di ottimalità
MPS001	Schedule Variance	$\geq -5\%$	≥ 0
MPS002	Budget Variance	$\geq -10\%$	≥ 0
MPS003	Function coverage	$\geq 98\%$	100%
MPS004	Statement coverage	$\geq 97\%$	100%
MPS005	Branch coverage	$\geq 95\%$	100%
MPS006	Condition coverage	$\geq 99\%$	100%

Tabella 2.1: Tabella delle metriche della qualità di processo

A. Ciclo di Deming o PDCA

Ogni processo deve essere organizzato basandosi sul principio del miglioramento continuo (o ruota di Deming):

Plan (pianificare): viene definito un piano che parte dalla definizione di problemi e obiettivi, pianifica compiti, assegna responsabilità, studia il caso, analizza le cause della criticità, definisce azioni correttive;

Do (eseguire): vengono implementate le attività secondo le linee definite durante la fase Plan;

Check (valutare): viene verificato l'esito delle azioni di miglioramento rispetto alle attese;

Act (agire): vengono applicate le correzioni necessarie per colmare le carenze rilevate, e vengono standardizzate le attività correttamente eseguite.

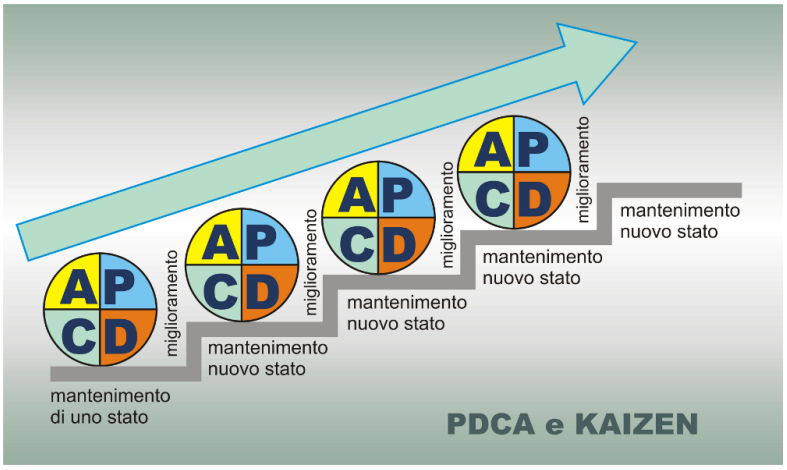

Figura A.1: Ciclo di Deming

B. ISO/IEC 15504

Lo standard ISO/IEC 15504 contiene un modello di riferimento che definisce

- Process dimension;
- Capability dimension.

La dimensione di processo divide i processi in cinque categorie:

- Customer-supplier;
- Engineering;
- Supporting;
- Management;
- Organization.

Per ogni processo, lo standard ISO/IEC 15504 definisce dei livelli di capacità:

Livello 5 **Optimizing process**: il processo è continuamente migliorato;

Livello 4 **Predictable process**: il processo è adottato sistematicamente, entro limiti definiti;

Livello 3 **Established process**: un processo stabilito si basa su un processo standard;

Livello 2 **Managed process**: il processo è gestito e i prodotti sono stabiliti, controllati e mantenuti;

Livello 1 **Performed process**: il processo è implementato e raggiunge lo scopo stabilito;

Livello 0 **Incomplete process**: il processo non è implementato o non raggiunge lo scopo stabilito.

La capacità dei processi viene misurata attraverso degli attributi di processo.

- Livello 1
  - **Process performance:** capacità di un processo di raggiungere gli obiettivi trasformando input identificabili in output identificabili;
- Livello 2
  - **Performance management:** capacità del processo di elaborare un prodotto coerente con gli obiettivi fissati;
  - **Work product management:** capacità del processo di elaborare un prodotto documentato, controllato e verificato;
- Livello 3
  - **Process definition:** l'esecuzione del processo si basa su standard di processo per raggiungere i propri obiettivi;
  - **Process deployment:** capacità del processo di attingere a risorse tecniche e umane appropriate per essere attuato efficacemente;
- Livello 4
  - **Process measurement:** gli obiettivi e le misure di prodotto e di processo vengono usati per garantire il raggiungimento dei traguardi definiti in supporto ai target aziendali;
  - **Process control:** il processo viene controllato tramite misure di prodotto e processo per effettuare correzioni migliorative al processo stesso;
- Livello 5
  - **Process innovation:** i cambiamenti strutturali, di gestione e di esecuzione vengono gestiti in modo controllato per raggiungere i risultati fissati;
  - **Process optimization:** le modifiche al processo sono identificate e implementate per garantire il miglioramento continuo nella realizzazione degli obiettivi di business dell'organizzazione.

Ogni attributo consiste di una o più pratiche generiche che sono ulteriormente elaborate in indicatori pratici per aiutare la valutazione delle performance, sotto forma di indici N-P-L-F:

- Non soddisfatto (0 - 15%);

- Parzialmente soddisfatto (>15% - 50%);
- Largamente soddisfatto (>50% - 85%);
- Totalmente soddisfatto (>85% - 100%)

3. Qualità di prodotto

3.1 Scopo

Per garantire una buona qualità di prodotto, il gruppo *353* ha individuato dallo standard ISO/IEC 9126 le qualità che ritiene più importanti nell'arco del ciclo di vita del prodotto e le ha istanziate individuando obiettivi e metriche coerenti con i livelli di qualità perseguiti.

3.2 Qualità dei documenti

I documenti prodotti dal gruppo *353* dovranno essere leggibili, comprensibili e corretti dal punto di vista ortografico, sintattico, logico e semantico.

3.2.1 Comprensione

Obiettivi di qualità

- **Leggibilità:** i documenti prodotti dovranno essere leggibili e comprensibili a persone con licenza di istruzione media;
- **Correttezza ortografica:** i documenti prodotti non dovranno contenere errori ortografici.

Metriche

- **MPDD001 Indice di Gulpease:** è l'indice di leggibilità tarato sulla lingua italiana. Considera due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero di lettere. La formulata per il suo calcolo è la seguente:

$$IG = 89 + \frac{300 * N_F - 10 * N_L}{N_P}$$

dove N_F è il numero delle frasi, N_P il numero delle lettere e N_P il numero delle parole. Il risultato I è un numero compreso tra 0 e 100. In generale risulta che i testi con indice inferiore a:

  - 80 sono difficili da leggere per chi ha una licenza elementare;
  - 60 sono difficili da leggere per chi ha una licenza media;
  - 40 sono difficili da leggere per chi ha un diploma superiore.

- **MPDD002 Formula di Flesch:** è una formula che serve per misurare la leggibilità di un testo in inglese:

$$F = 206,835 - (0,846 * S) - (1,015 * P)$$

dove S è il numero delle sillabe, calcolato su un campione di 100 parole e P è il numero medio di parole per frase. La leggibilità è alta se F è superiore a 60, media se fra 50 e 60, bassa sotto a 50;

- **MPDD003 Errori ortografici:** gli errori ortografici possono essere identificati tramite lo strumento 'Controllo ortografico' presente in TexStudio. Sarà poi compito del Verificatore correggerli.

3.3 Qualità del software

3.3.1 Funzionalità

Rappresenta la capacità del prodotto di fornire tutte le funzioni che sono state individuate attraverso l'*Analisi dei requisiti*.

Obiettivi qualità

Il gruppo *353* si impegnerà affinché:

- **Adeguatezza**: le funzionalità fornite siano conformi rispetto le aspettative;
- **Accuratezza**: il prodotto fornisca i risultati attesi, con il livello di dettaglio richiesto.

Metriche

- **MPDS001 Copertura requisiti obbligatori:** indica la percentuale dei requisiti obbligatori coperti dall'implementazione. La formula di misurazione è

$$CRO = (\frac{N_{ROS}}{N_{RO}}) * 100$$

dove N_{ROS} è il numero di requisiti obbligatori soddisfatti e N_{RO} è il numero totale dei requisiti obbligatori;

- **MPDS002 Copertura requisiti accettati:** indica la percentuale dei requisiti desiderabili e facoltativi coperti dall'implementazione. La formula di misurazione è
$$CRA = (\frac{N_{RAS}}{N_{RA}}) * 100$$
dove N_{RAS} è il numero di requisiti accettati soddisfatti e N_{RA} è il numero totale dei requisiti accettati;

- **MPDS003 Accuratezza rispetto alle attese:** indica la percentuale di risultati concordi alle attese. La formula di misurazione è
$$ARA = (1 - \frac{N_{TD}}{N_{TE}}) * 100$$
dove N_{TD} è il numero di test che producono risultati discordi alle attese e N_{TE} è il numero di test-case eseguiti.

3.3.2 Affidabilità

Rappresenta la capacità del prodotto software di svolgere correttamente le sue funzioni durante il suo utilizzo, anche in caso in cui si presentino situazioni anomale.

Obiettivi di qualità

L'esecuzione del prodotto dovrà presentare le seguenti caratteristiche:

- **Maturità:** evitare che si verifichino malfunzionamenti, operazioni illegali e failure in seguito a fault;

- **Tolleranza agli errori:** nel caso in cui si presentino degli errori, dovuti a guasti o ad un uso scorretto dell'applicativo, questi devo essere gestiti in modo da mantenere alto il livello di prestazioni.

Metriche

- **MPDS004 Densità di failure:** indica la percentuale di testing che si sono concluse in failure. La sua formula di misurazione è
$$DF = (\frac{N_{FR}}{N_{TE}}) * 100$$
dove N_{FR} è il numero di failure rilevati durante l'attività di testing e N_{TE} è il numero di test-case eseguiti;

- **MPDS005 Blocco di operazioni non corrette:** indica la percentuale di funzionalità in grado di gestire correttamente i fault che potrebbero verificarsi . La sua formula di misurazione è

$$BNC = (\frac{N_{FE}}{N_{ON}}) * 100$$

dove N_{FE} è il numero di failure evitati durante i test effettuati e N_{ON} è il numero di test-case eseguiti che prevedono l'esecuzione di operazioni non corrette, causa di possibili failure.

3.3.3 Usabilità

Rappresenta la capacità del prodotto di essere facilmente comprensibile e attraente in ogni sua parte per qualsiasi utente che lo andrà ad utilizzare.

Obiettivi di qualità

Il prodotto dovrà puntare ai seguenti obiettivi di usabilità:

- **Comprensibilità:** l'utente deve essere in grado di riconoscere le funzionalità offerte dal software e deve comprendere le modalità di utilizzo per raggiungere i risultati attesi;
- **Apprendibilità:** deve essere data la possibilità all'utente di imparare ad utilizzare l'applicazione senza troppo impegno;
- **Operabilità:** le funzioni presenti devono essere coerenti con le aspettative dell'utente;
- **Attrattiva:** il software deve essere piacevole per chi ne fa uso.

Metriche

- **MPDS006 Comprensibilità delle funzioni offerte:** indica la percentuale di operazioni comprese in modo immediato dall'utente, senza la consultazione del manuale. La sua formula di misurazione è

$$CFC = (\frac{N_{FC}}{N_{FO}}) * 100$$

dove N_{FC} è il numero di funzionalità comprese in modo immediato dall'utente durante l'attività di testing del prodotto e N_{FO} è il numero di funzionalità offerte dal sistema;

- **MPDS007 Facilità di apprendimento delle funzionalità:** indica il tempo medio impiegato dall'utente nell'imparare ad usare correttamente una data funzionalità. Si misura tramite un indicatore numerico, che indica i minuti impiegati da un utente per apprendere il funzionamento di una certa funzionalità;
- **MPDS008 Consistenza operazionale in uso:** indica la percentuale di messaggi e funzionalità offerte all'utente che rispettano le sue aspettative riguardo al comportamento del software. La sua formula di misurazione è

$$COU = (\frac{N_{MFU}}{N_{MFO}}) * 100$$

dove N_{MFU} è il numero di messaggi e funzionalità che non rispettano le aspettative dell'utente e N_{MFO} è il numero di messaggi e funzionalità offerte dal sistema.

3.3.4 Efficienza

Rappresenta la capacità di eseguire le funzionalità offerte dal software nel minor tempo possibile utilizzando al tempo stesso il minor numero di risorse disponibili.

Obiettivi di qualità

Il prodotto dovrà essere efficiente, in particolare:

- **Comportamento rispetto al tempo:** per svolgere le sue funzioni il software deve fornire adeguati tempi di risposta ed elaborazione;
- **Utilizzo delle risorse:** il software quando esegue le sue funzionalità deve utilizzare un appropriato numero e tipo di risorse.

Metriche

- **MPDS009 Tempo di risposta:** indica il tempo medio che intercorre fra la richiesta software di una determinata funzionalità e la restituzione del risultato all'utente. La sua formula di misurazione è

$$TR = \frac{\sum_{i=1}^{n} T_i}{n}$$

dove T_i è il tempo intercorso fra la richiesta i di una funzionalità ed il comportamento delle operazioni necessarie a restituire un risultato a tale richiesta.

3.3.5 Manutenibilità

Rappresenta la capacità del prodotto di essere modificato, tramite correzioni, miglioramenti o adattamenti del software a cambiamenti negli ambienti, nei requisiti e nelle specifiche funzionali.

Obiettivi di qualità

Le operazioni di manutenzione andranno agevolate il più possibile adottando le seguenti caratteristiche:

- **Analizzabilità:** il software deve consentire una rapida identificazione delle possibili cause di errori e malfunzionamenti;
- **Modificabilità:** il prodotto originale deve permettere eventuali cambiamenti in alcune sue parti;
- **Stabilità:** non devono insorgere effetti indesiderati in seguito a modifiche effettuate sul software;
- **Testabilità:** il software deve poter essere facilmente testato per valiare le modifiche effettuate.

Metriche

- **MPDS010 Capacità di analisi di failure:** indica la percentuale di modifiche effettuate in risposta a failure che hanno portato all'introduzione di nuove failure in altre componenti del sistema. La sua formula di misurazione è
$$CAF = (\frac{N_{FI}}{N_{FR}}) * 100$$
dove N_{FI} è il numero di failure delle quali sono state individuate le cause e N_{FR} è il numero di failure rilevate;
- **MPDS011 Impatto delle modifiche:** indica la percentuale di modifiche effettuate in risposta a failure che hanno portato all'introduzione di nuove failure in altre componenti del sistema. La sua formula i misurazione è
$$IM = (\frac{N_{FRF}}{N_{FR}}) * 100$$
dove N_{FRF} è il numero di failure risolte con l'introduzione di nuove failure e N_{FR} è il numero di failure risolte.

3.3.6 Portabilità

Rappresenta la capacità del software di poter essere utilizzato su diversi ambienti.

Obiettivi di qualità

Sarò agevolata la portabilità del prodotto adottando i seguenti obiettivi:

- **Adattabilità:** il prodotto deve adattarsi a tutti quelli ambienti di lavoro nei quali è stato previsto un suo utilizzo, senza dover apportare modifiche dello stesso;
- **Sostituibilità:** l'applicativo deve poter sostituire un altro software che ha lo stesso scopo e lavora nel medesimo ambiente.

Metriche

- **MPDS012 Versioni dei browser supportate:** indica la percentuale di versioni di browser attualmente supportate, fra quelle individuate dai requisiti. La sua formula di misurazione è

  $$VB = (\frac{N_{VS}}{N_{VI}}) * 100$$

  dove N_{VS} è il numero di versioni di browser supportate dal prodotto e N_{VI} è il numero di versioni di browser che devono essere supportate dal prodotto;
- **MPDS013 Inclusione di funzionalità da altri prodotti:** indica la percentuale del software utilizzato in precedenza dall'utente che produce risultati simili a quelli ottenuti dal prodotto in oggetto. La sua formula di misurazione è

  $$IFP = (\frac{N_{FPA}}{N_{FPP}}) * 100$$

  dove N_{FPA} è il numero di funzionalità del software utilizzato in precedenza dall'utente che produce risultati simili a quelli ottenuti dal prodotto in oggetto e N_{FPP} è il numero di funzionalità offerte dal software utilizzato in precedenza dall'utente.

3.4 Tabella delle metriche

Questa tabella indica i **range** di accettazione e di ottimalità per le metriche utilizzate per la qualità di prodotto:

ID	Nome	Range di accettazione	Range di ottimalità
MPDD001	Indice di Gulpease	50 - 100	60 - 100
MPDD002	Formula di Flesch	40 - 60	50 - 60
MPDD003	Errori ortografici	100% corretti	100% corretti
MPDS001	Copertura requisiti obbligatori	100%	100%
MPDS002	Copertura requisiti accettati	60% - 100%	80% - 100%
MPDS003	Accuratezza rispetto alle attese	90% - 100%	100%
MPDS004	Densità di failure	0% - 10%	0%
MPDS005	Blocco di operazioni non corrette	80% - 100%	100%
MPDS006	Comprensibilità delle funzioni offerte	80% - 100%	90% - 100%
MPDS007	Facilità di apprendimento delle funzionalità	0 - 20 min	0 - 10 min
MPDS008	Consistenza operazionale in uso	80% - 100%	90% - 100%
MPDS009	Tempo di risposta	0 - 8 sec	0 - 3 sec
MPDS010	Capacità di analisi di failure	60% - 100%	80% - 100%
MPDS011	Impatto delle modifiche	0% - 20%	0% - 10%
MPDS012	Versioni di browser supportate	70% - 100%	100%
MPDS013	Inclusione di funzionalità da altri prodotti	80% - 100%	90% - 100%

Tabella 3.1: Tabella delle metriche della qualità di prodotto

4. Specifica dei test

4.1 Scopo

Per garantire la maggiore rilevazione di errori durante la fase di sviluppo, il team porrà grande attenzione sull'analisi dinamica del codice: ovvero l'esecuzione di test automatici.
Durante la fase di codifica, programmatori e verificatori seguiranno la filosofia TDD in tutte le sue varianti (ATDD e BDD) per garantire il corretto sviluppo dell'applicativo: verranno prima scritti i test e poi il codice necessario per soddisfarli.

4.2 Tipi di test

Sono stati individuate due macro categorie di test.

4.2.1 Test di modulo

I test appartenenti a questa categoria mirano alla verifica della logica del software e verranno scritti ed eseguiti dai programmatori, il loro successo costituirà vincolo per poter consegnare il codice all'interno del repository.
Si dividono in:

- **Test di unità** [**TU**]: Con questa tipologia si cerca di verificare la più piccola parte di lavoro prodotta da un programmatore, corrispondente alla più piccola unità di logica del prodotto, che può essere una singola classe, metodo o funzione oppure un insieme di essi.
  Verranno sviluppati test black-box per testare le funzionalità delle unità e test white-box per verificarne la struttura;

- **Test di integrazione**[**TI**]:Con questa categoria si cerca di verificare l'integrazione tra le unità logiche che formano i vari componenti del sistema.
  La tecnica scelta per testare l'integrazione è quella dal basso verso l'alto:

si testano prima le parti con minore dipendenza funzionale e con maggiore funzionalità, per poi risalire l'albero delle dipendenze.

4.2.2 Test ad alto livello

I test appartenenti a questa categoria mirano alla verifica delle funzionalità del sistema, si concentrano di più sul comportamento dell'applicazione e vengono gestite dal team di Quality Assurance (QA: i verificatori). Alcuni di questi test sono manuali e per garantirne la ripetibilità la loro organizzazione sarà gestita tramite dei tool appositi che verranno discussi in sede di technology baseline.

- **Test Funzionali [TF]:** possono essere visti come dei test di unità ad alto livello, verificano l'implementazione delle specifiche del prodotto e si concentrano sulle funzionalità delle suddette specifiche: l'analisi della struttura è infatti relegata ai test di unità [TU] veri e propri.
  Il fallimento di questi test può causare l'avvio dei test di regressione: l'esecuzione a ritroso dei test di modulo per scovare l'errore.
  Questi test possono essere automatizzati e scritti dai programmatori, ma è bene affiancarli ad una revisione ed esecuzione dei verificatori;

- **Test di sistema [TS]**: questa tipologia di test punta a verificare il sistema e l'architettura nella sua interezza, sono test pesanti e complessi; la loro implementazione verrà discussa in sede di technology baseline. Questi test necessitano di componenti software ma devono essere supervisionate da dei verificatori, verranno quindi eseguiti dal team di QA;

- **Test di validazione [TV]**: si tratta dei test finali che valutano se il sistema sviluppato corrisponde alle richieste del proponente, godono quindi di un forte accoppiamento con i requisiti. Sono principalmente test manuali e verranno eseguiti dal team di QA, nelle fasi finali dello sviluppo verranno effettuati assieme ai proponenti per determinare la validità del prodotto.

5. Tracciamento dei test

5.1 Test di Validazione

5.2 Test di Sistema

5.3 Test di integrazione

5.4 Test di unità

6. Resoconto attività di verifica

6.1 Revisione dei Requisiti

6.1.1 Tracciamento

6.1.2 Analisi Statica documenti

6.1.3 Verifiche automatiche