Zing-BDSMApp
YOUR SOCIAL STORE

MashUp
Software Engineering Group

info@mashup-unipd.it

Informazioni Documento

Nome documento	*Piano di Qualifica*
Versione	*v1.0.0*
Data redazione	2014-12-23
Redattori	Ceccon Lorenzo Faccin Nicola
Verificatori	Santacatterina Luca
Approvazione	Tesser Paolo
Lista distribuzione	*MashUp* *Prof. Tullio Vardanega* *Prof. Riccardo Cardin* *Dott. David Santucci* - Zing Srl
Uso	Esterno

Sommario

Documento che descrive le attività di verifica e validazione adottate dal gruppo *MashUp* per il progetto BDSMApp.

Diario Revisioni

Modifica	Autore & Ruolo	Data	Versione
Approvazione documento	Tesser Paolo *Responsabile di progetto*	2014-01-20	v1.0.0
Eseguita verifica documento	Santacatterina Luca *Verificatore*	2014-01-19	v0.2.0
Modificato documento a seguito della verifica	Faccin Nicola *Verificatore*	2014-01-18	v0.1.1
Eseguita verifica documento	Santacatterina Luca *Verificatore*	2014-01-16	v0.1.0
Iniziata stesura capitolo Resoconto delle attività di verifica	Ceccon Lorenzo *Verificatore*	2015-01-13	v0.0.8
Terminata stesura capitolo Visione generale della strategia di verifica	Faccin Nicola *Verificatore*	2015-01-12	v0.0.7
Iniziata stesura capitolo Visione generale della strategia di verifica	Ceccon Lorenzo *Verificatore*	2014-12-31	v0.0.6
Terminata stesura appendici	Faccin Nicola *Verificatore*	2014-12-30	v0.0.5
Iniziata stesura appendici	Ceccon Lorenzo *Verificatore*	2014-12-23	v0.0.4
Stesura capitolo Gestione amministrativa della revisione	Faccin Nicola *Verificatore*	2014-12-23	v0.0.3
Stesura capitolo Introduzione	Ceccon Lorenzo *Verificatore*	2014-12-15	v0.0.2
Generazione struttura del documento	Ceccon Lorenzo *Verificatore*	2014-12-14	v0.0.1

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di definire la strategia e descrivere le modalità di verifica e validazione$_G$ che il gruppo MashUp intende adottare per lo sviluppo del progetto, al fine di raggiungere gli obbiettivi qualitativi prefissati. Per perseguire questi obbiettivi è necessaria una costante attività di verifica in modo da permettere di rilevare e risolvere eventuali anomalie.

1.2 Scopo del prodotto

Lo scopo del prodotto è di creare una nuova infrastruttura che permetta di interrogare Big Data recuperati dai social network, quali: Facebook, Twitter, Instagram. L'applicazione sarà composta da due parti:

- consultazione e interrogazione con interfaccia web per utente;
- servizi web REST interrogabili.

1.3 Glossario

Al fine di evitare ogni ambiguità relativa al linguaggio usato nei documenti viene allegato il *Glossario v1.0.0*. Esso ha lo scopo di definire ed analizzare tutti i termini tecnici del progetto e di fugare eventuali ambiguità fornendo un'accurata descrizione. Tutte le occorrenze di tali termini nei documenti verranno contrassegnate con una "G" a pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Norme di progetto**: *Norme di Progetto v1.0.0*;
- **Capitolato d'appalto C1:** *BDSMApp: Big Data Social Monitoring App* http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C1.pdf;
- **Standard ISO/IEC 9126:** http://en.wikipedia.org/wiki/ISO/IEC_9126;
- **Standard ISO/IEC 15504:** http://en.wikipedia.org/wiki/ISO/IEC_15504.

1.4.2 Informativi

- **Piano di progetto**: *Piano di Progetto v1.0.0*;
- **Slides di Ingegneria del Software modulo A:** http://www.math.unipd.it/~tullio/IS-1/2014/;
- **SWEBOK 2004:** *Chapter 11 - Software Quality* http://www.computer.org/portal/web/swebok/html/ch11;
- **Software Engineering - Ian Sommerville - 9th Edition(2010)**: *Chapters 8, 24, 26.*

2 Visione generale della strategia di verifica

2.1 Organizzazione

L'attività di verifica accompagnerà l'intero ciclo di vita del software e sarà effettuata su tutti i processi realizzati e sugli output prodotti da questi ultimi. La verifica sarà applicata solamente ai cambiamenti effettuati dall'ultima versione approvata del prodotto.
Il team ha scelto di adottare un modello di ciclo di vita di tipo incrementale suddiviso in diverse fasi e riportate dettagliatamente nel *Piano di Progetto v1.0.0*. Per ciascuna di queste fasi verranno utilizzate specifiche attività di verifica.
Il processo di verifica sarà così composto:

- **Ricerca e implementazione degli strumenti:** in questa fase l'attività di verifica consente di verificare che tutti gli script creati siano corretti e che l'output prodotto dagli stessi sia uguale alle aspettative;
- **Analisi dei requisiti e di dettaglio:** in questa fase si verificherà che ogni requisito abbia corrispondenza in un caso d'uso e si effettueranno dei test sulla documentazione prodotta per verificare che rispetti le *Norme di Progetto v1.0.0*;
- **Progettazione architetturale:** l'attività di verifica in questa fase consiste nell'analizzare che la soluzione generale ad alto livello soddisfi i requisiti richiesti oltre a verificare i processi utilizzati per ottenere questa soluzione;
- **Progettazione di dettaglio e codifica dei requisiti obbligatori, desiderabili e opzionali:** si eseguiranno attività di verifica sui processi di progettazione e codifica del codice prodotto dai programmatori facendo uso di tecniche di analisi statica e dinamica;
- **Validazione$_G$:** in quest'ultima fase verrà effettuato il collaudo del prodotto che garantirà il corretto funzionamento del prodotto realizzato.

2.2 Pianificazione strategica e temporale

L'attività di verifica necessaria, per il miglioramento della qualità dei processi e del prodotto, deve essere sistematica ed organizzata. Ciò permetterà l'individuazione e la correzione degli errori il prima possibile evitando la propagazione di questi ultimi in larga scala.
Ciascuna attività che riguarda la documentazione o la codifica dovrà essere preceduta da uno studio preliminare che ci permetta di rendere chiaro la struttura degli stessi. Questo studio preventivo ci consentirà di ottenere un maggiore livello di qualità e una minore possibilità di fallimento.
Per quanto riguarda le tempistiche, l'obbiettivo primario è quello di rispettare le scadenze forniteci del committente$_G$ e riportate nel *Piano di Progetto v1.0.0*.

2.3 Responsabilità

Le responsabilità relative all'assegnazione degli incarichi appartengono al *Responsabile di Progetto*, mentre le responsabilità relative all'adeguamento dell'ambiente di lavoro per lo svolgimento di tutti i compiti necessari alla realizzazione del progetto appartengono all'*Amministratore di Progetto*.

2.4 Risorse necessarie

Le risorse necessarie alla verifica della qualità dei processi e del prodotto sono:

- **Risorse umane:** il *Responsabile di Progetto* controlla la qualità dei processi interni, l'*Amministratore di Progetto* definisce le norme e i piani per le attività di verifica, il *Programmatore* esegue le prove di verifica e validazione$_G$ del codice, il *Verificatore* esegue la verifica dei documenti e fornisce i risultati delle prove effettuate. Per una descrizione più dettagliata dei ruoli consultare il *Piano di Progetto v1.0.0*;
- **Risorse software:** sono necessari strumenti per il tracciamento dei requisiti, per la stesura dei documenti in LaTeX, per la creazione di diagrammi UML$_G$, per lo sviluppo del prodotto e per il supporto e la verifica del codice;
- **Risorse hardware:** sono necessari computer per scrivere documenti e creare il prodotto software finale. È richiesto un ambiente di lavoro stabile in cui poter lavorare al progetto.

2.5 Tecniche

2.5.1 Analisi Statica

Comprende due tipologie di analisi che verranno utilizzate durante tutto il ciclo di vita del software, queste sono applicabili sia alla documentazione che appunto al software:

- **Walkthrough:** tecnica necessaria nelle prime fasi del progetto, vista l'inesperienza da parte del gruppo nell'attuare un tipo di verifica più precisa e mirata. Questo tipo di analisi è molto dispendiosa in quanto prevede una lettura critica del codice o del documento in analisi, senza avere idea del tipo di anomalia ricercata. Una volta trovata l'anomalia, nella fase di redazione dei documenti o nella fase di codifica che sia, verrà aggiunta alla lista di controllo così che in futuro si venga ad utilizzare sempre di più l'Inspection a discapito del Walkthrough;
- **Inspection:** lettura mirata del codice e/o dei documenti, guidata da una lista di controllo definita gradualmente, sia grazie all'esperienza personale, sia attraverso l'arricchimento derivante dall'attività di Walkthrough. Ogni difetto riscontrato verrà corretto e documentato con dei rapporti sulle attività svolte. La lista di controllo inizialmente non sarà abbastanza ampia da garantire l'affidabilità dell'elemento analizzato, per questo almeno nelle prime fasi verrà affiancata al Walkthrough.

2.5.2 Analisi Dinamica

La verifica e la validazione$_G$ verrà applicata solamente al prodotto software in sviluppo tramite l'esecuzione di determinati test. Questi test dovranno essere automatici e ripetibili, in modo da poter essere eseguiti in qualsiasi momento e per tutta la durata dello sviluppo.
Per una corretta valutazione in base ai risultati di questi test è richiesto che siano rispettate le seguenti regole:

- **Ambiente:** specifica l'ambiente software e hardware in cui viene eseguito il test;

- **Specifica:** devono essere specificati i dati in input ed output in modo da poter effettuare un test di congruenza;
- **Procedure:** possono essere inserite ulteriori istruzioni per l'esecuzione dei test.

I test, dato lo stesso input sul medesimo ambiente di esecuzione, dovranno fornire gli stessi risultati. In questo modo sarà più facile riconoscere le anomalie.

- **Test di unità:** verifica ogni singola unità del prodotto software, avvalendosi di strumenti come driver$_{\mathbf{G}}$ e/o stub$_{\mathbf{G}}$. In particolare si verifica che per ogni unità siano esplorati tutti i cammini di esecuzione possibili e che i requisiti per quella determinata unità siano soddisfatti;
- **Test di integrazione:** verifica dei componenti costituiti dall'integrazione di più unità che hanno soddisfatto i test di unità; aiuta ad identificare errori residui nella realizzazione dei componenti, modifiche delle interfacce o cambiamenti nei requisiti. A questo scopo verranno utilizzate classi simulate per verificarne l'interazione, senza più la necessità di verificare il loro comportamento interno;
- **Test di sistema:** validazione$_{\mathbf{G}}$ del sistema attraverso la verifica della copertura di tutti i requisiti individuati nell'*Analisi dei Requisiti v1.0.0*;
- **Test di regressione:** ogni qualvolta che si applica una modifica ad una parte del codice precedentemente testato, devono essere svolti tutti i test di unità e integrazione ad essa relativa;
- **Test di accettazione:** collaudo controllato dal committente$_{\mathbf{G}}$, se quest'ultimo ha esito positivo, si procede al rilascio del prodotto.

2.6 Misure e metriche

Descrizione delle metriche e delle misure per rendere quantificabili e conseguentemente qualificabili i processi, i documenti e il software prodotto.

2.6.1 Metriche per i processi

L'organizzazione interna dei processi si basa sul principio PDCA, vedi Figura 1, che è in grado di garantire un miglioramento continuo della qualità di tutti i processi e conseguentemente dei prodotti derivanti dai processi. I processi saranno pianificati dettagliatamente rispetto ai requisiti e alle risorse disponibili. Se durante il processo di verifica l'analisi evidenzia dei valori che si discostano, in modo peggiorativo, dai piani prefissati, questo denoterà la presenza di un problema che verrà risolto in modo correttivo sul processo o eventualmente sul piano iniziale dello stesso.
Le misurazioni sul processo consistono in:

- Tempo impiegato per essere completato;
- Cicli iterativi interni al processo;
- Risorse utilizzate e/o consumate durante il processo;
- Attinenza ai piani stabiliti;
- Soddisfazione dei requisiti richiesti.

2.6.2 Metriche per i documenti

Indice Gulpease$_G$: questo indice, tarato specificatamente per la lingua italiana, ha anche il vantaggio di utilizzare la lunghezza delle parole in lettere e non delle sillabe, semplificandone il calcolo.

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

100 indica la leggibilità più alta mentre 0 quella più bassa, sono presenti dei range così da poter quantificare meglio la complessità del documento in analisi:

- inferiori a 80 sono difficili da leggere per chi ha la licenza elementare;
- inferiori a 60 sono difficili da leggere per che ha la licenza media;
- inferiori a 40 difficili da leggere per chi possiede un diploma superiore;

Range-ottimale[50-100], range-accettazione [40-100].

2.6.3 Metriche per il software

- **Complessità Ciclomatica:** è utilizzata per misurare la complessità di un metodo, attraverso il grafo di controllo di flusso che misura direttamente il numero di cammini linearmente indipendenti. I nodi di questo grafo rappresentano gruppi indivisibili di istruzioni e gli archi connettono due nodi solamente se le istruzioni di un nodo possono essere eseguite immediatamente dopo le istruzioni dell'altro nodo.
  In questo progetto si cercherà di rispettare la raccomandazione di *McCabe*, che sviluppò tale teoria, ossia quella di non superare una complessità di 10. Rispettando questo vincolo si aumentano le possibilità di riuso del codice, manutenibilità, coesione e correttezza di quest'ultimo. Il vincolo presentato sarà di tipo lasco, ossia potrà essere portato a valori maggiori nell'eventualità porti a notevoli benefici in termini di velocità di esecuzione.

  Valore-ottimale <10, valore-accettazione <15;

- **Numero di metodi:** metrica utilizzata per calcolare una media delle occorrenze dei metodi per package$_G$; valori alti potrebbero indicare la necessità di scomporlo.

  Range-ottimale [3-8], range-accettazione [3-10];

- **Numero di parametri:** metrica utilizzata per calcolare il numero di parametri formali di un metodo. Un valore basso e indice di maggior manutenibilità e astrazione del codice.

  Range-ottimale [0-4], range-accettazione [0-8];

- **Linee di codice per linee di commento:** metrica atta a migliorare la manutenibilità del codice attraverso il monitoraggio del rapporto tra questi valori.

  Valore-ottimale <0.20, valore-accettazione <0.35;

- **Bugs for lines of code:** metrica per la misura dei bug_G trovati per un certo quantitativo di linee di codice. Questa metrica è utile in quanto all'aumentare dell'ampiezza del codice si aumenta la probabilità di nascondere degli errori. Presupponendo che nessuno del gruppo avrà conoscenze sufficienti dello stack tecnologico che si andrà ad utilizzare si partirà con un valore di accettazione alto per poi cercare di ridurlo in modo incrementale. L'obbiettivo fissato è quello di raggiungere valori compresi tra 0 e 20. Difficoltà particolari verranno gestite dal responsabile di progetto;
- **Numero di livelli di annidamento:** metrica per misurare il livello di annidamento dei metodi. Un numero elevato comporta eccessiva complessità del codice e ne riduce il livello di astrazione.

  Range-ottimale [1-4], range-accettazione [1-6].

3 Gestione amministrativa della revisione

3.1 Gestione delle anomalie e delle discrepanze

La fase di verifica porta alla ricerca di eventuali difetti, i quali possono essere errori logici oppure anomalie presenti nel codice. Il *Verificatore* ha il compito di esaminare scrupolosamente il codice del prodotto e sottolineare le eventuali anomalie e problemi per essere risolti successivamente.
Per anomalia si intende una deviazione del prodotto dalla sue aspettative, quindi, causa un malfunzionamento del sistema. Si possono dividere in:

- **Computational Error:** differenza tra un valore calcolato, osservato o misurato e il suo valore teoricamente corretto;
- **Error:** azione umana che produce un risultato errato;
- **Defect:** imperfezione o carenza all'interno di un prodotto, il quale non soddisfa le sue esigenze o specifiche prefissate e necessità di essere sistemato;
- **Fault:** difetto all'interno del codice sorgente. Può essere inteso come la codifica di un errore umano nel codice sorgente;
- **Failure:** evento nel quale un sistema, o un suo componente, non esegue una funzione richiesta entro limiti specificati. Si verifica quando, sotto specifiche condizioni, viene rilevato un *Fault*.

La gestione di queste anomalie assume, quindi, un ruolo di primaria importanza all'interno del progetto e dovranno essere gestite in maniera più rapida possibile.
Per discrepanza invece, si intende una divergenza tra il prodotto sviluppato e quello atteso. La presenza di una discrepanza fa si che il prodotto funzioni senza che si verifichino *Failure* ma, rende il prodotto realizzato errato rispetto alle attese. Una discrepanza sarà, quindi, trattata come un'anomalia di bassa priorità.
Ogni qualvolta il *Verificatore* incontrerà un'anomalia, dovrà seguire una procedura standard sia per quanto riguarda le anomalie che riguardano i documenti, sia quelle relative al software. Il *Verificatore* dovrà quindi aprire un ticket$_{\mathbf{G}}$ seguendo le regole riportate nel *Norme di Progetto v1.0.0*.
Il *Responsabile di Progetto* avrà il compito di approvare il ticket$_{\mathbf{G}}$; se approvato, il membro a cui è stato assegnato tale ticket dovrà impegnarsi per risolvere entro i termini stabiliti.

3.2 Procedure di controllo di qualità di processo

Il gruppo, per il controllo della qualità di processo, ha deciso di utilizzare il ciclo PDCA$_{\mathbf{G}}$ che ci consente di ottenere un miglioramento continuo dei processi.
Il PDCA$_{\mathbf{G}}$, noto anche come ciclo di Deming, è un metodo di gestione iterativo a quattro fasi per il controllo e il miglioramento continuo dei processi.
Le quattro fasi che lo compongono sono:

- **Plan:** stabilisce gli obiettivi e i processi necessari per ottenere risultati uguali a quelli attesi;
- **Do:** fase composta dall'attuazione del piano, dall'esecuzione del processo e dalla creazione del prodotto. Termina con una raccolta dei dati e creazione di grafici sul risultato di quanto ottenuto;

- **Check:** si confrontano i risultati ottenuti dalla fase precedente con i risultati stabiliti durante la pianificazione per verificare la presenza di differenze;
- **Act:** si effettuano correzioni laddove sono presenti differenze tra i risultati ottenuti e quelli previsti. Si determinano le cause delle discrepanze e dove c++$_G$'è bisogno di applicare delle modifiche per ottenere un miglioramento del processo e di conseguenza del prodotto.

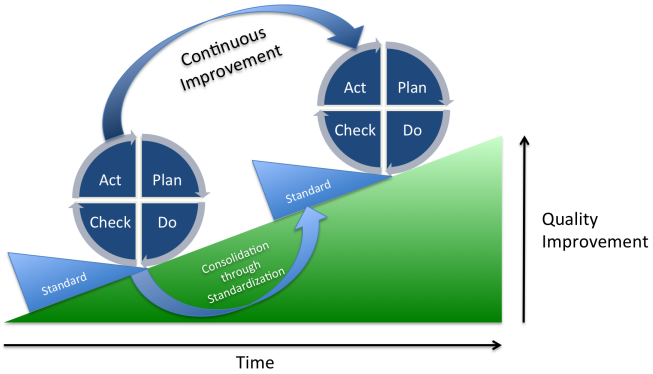

Figura 1: Ciclo di miglioramento della qualità PDCA

3.3 Procedure di controllo di qualità di prodotto

Per garantire la qualità del prodotto software si fa affidamento a due modalità di controllo:

- **Software Quality Assurance (SQA):** è un insieme di attività che assicurano che il software sviluppato sia conforme alle specifiche di qualità standard o definite dal gruppo. L'SQA è un processo appartenente al ciclo di vita del software che controlla regolarmente e preventivamente il software sviluppato per assicurare il rispetto degli obbiettivi di qualità prefissati;
- **Verifica e Validazione$_G$ (V&V):** è un processo che controlla che il sistema software soddisfi le specifiche e che raggiunga appieno il suo scopo.
  Per verifica si intende il processo attuo a valutare che il software in una determinata fase di sviluppo soddisfi le condizioni imposte all'inizio di tale fase. Per validazione$_G$ si intende il processo attuo a valutare se al termine del processo di sviluppo questo soddisfi i requisiti specificati. In altre parole, la verifica garantisce che il software è stato creato correttamente, mentre la validazione assicura che si è creato il giusto prodotto.

A Standard di qualità

A.1 Standard ISO/IEC 9126

Lo standard ISO$_G$/IEC$_G$ 9126 è uno standard creato per delineare delle normative utili a descrivere un modello di qualità del software. Lo standard propone un approccio in cui viene posta attenzione al miglioramento dell'organizzazione e dei processi di una società di software, in modo da migliorare di conseguenza la qualità del prodotto software.
Lo standard ISO$_G$/IEC$_G$ 9126 è suddiviso in quattro parti:

- **Modello di qualità:** la prima parte dello standard classifica il modello di qualità in sei caratteristiche generali e in varie sotto caratteristiche misurabili tramite l'utilizzo di metriche.
  Le sei caratteristiche generali e le relative sotto caratteristiche sono:
  - **Funzionalità:**
    - **Appropriatezza;**
    - **Accuratezza;**
    - **Interoperabilità;**
    - **Conformità;**
    - **Sicurezza.**
  - **Affidabilità:**
    - **Maturità;**
    - **Tolleranza agli errori;**
    - **Recuperabilità;**
    - **Aderenza.**
  - **Usabilità:**
    - **Comprensibilità;**
    - **Apprendibilità;**
    - **Operabilità;**
    - **Attrattiva;**
    - **Conformità.**
  - **Efficienza:**
    - **Comportamento rispetto al tempo;**
    - **Utilizzo delle risorse;**
    - **Conformità.**
  - **Manutenibilità:**
    - **Analizzabilità;**
    - **Modificabilità;**
    - **Stabilità;**
    - **Testabilità.**
  - **Portabilità:**
    - **Adattabilità;**
    - **Installabilità;**
    - **Conformità;**

    - **Sostituibilità.**

- **Qualità esterne:** le metriche esterne applicabili al software, e quindi rilevabili tramite l'analisi dinamica, misurano i comportamento del prodotto sulla base dei test, dall'operatività e dall'osservazione durante la sua esecuzione;
- **Qualità interne:** le metriche interne, misurabili attraverso l'analisi statica, sono utili per prevedere il livello della qualità esterna ed in uso, poiché i suoi attributi interni influiscono su quelli esterni ed in uso. Permettono così di individuare anomalie prima che queste ultime possano influenzare la qualità del prodotto finale;
- **Qualità in uso:** la qualità in uso, raggiungibile solo dopo aver ottenuto la qualità interna ed esterna, fornisce metriche per misurare il grado di utilizzabilità del prodotto da parte dell'utente finale.

A.2 Standard ISO/IEC 15504

Lo standard ISO$_{\mathbf{G}}$/IEC$_{\mathbf{G}}$ 15504, conosciuto anche come SPICE$_{\mathbf{G}}$, è un insieme di documenti tecnici per lo sviluppo di processi software, utili a valutare la dimensione dei processi tramite l'utilizzo di specifiche metriche. É derivato dallo standard ISO/IEC 12207 e da modelli di maturità quali Bootstrap, Trillium e il CMM.
Lo standard definisce la dimensione del processo e la suddivide nelle seguenti cinque categorie:

- **Custormer/Supplier;**
- **Engineering;**
- **Support;**
- **Management;**
- **Organization.**

Per ogni processo, viene definito un livello di capacità dei processi definito da una scala di sei livelli e da nove attributi suddivisi nei vari livelli:

- **Level 5. Optimizing process;**
    - Process Innovation;
    - Process Optimization.
- **Level 4. Predictable process;**
    - Process Measurement;
    - Process Control.
- **Level 3. Established process;**
    - Process Definition;
    - Process Deployment.
- **Level 2. Managed process;**
    - Performance Management;

  - Work Product Management.

- **Level 1. Performed process;**
  - Process Performance.
- **Level 0. Incomplete process.**

Ogni attributo è misurabile tramite l'utilizzo di una scala di valutazione divisa in quattro punti:

- **Not achieved (0-15%);**
- **Partially achieved (15-50%);**
- **Largely achieved (50-85%);**
- **Fully achieved (85-100%).**

Lo standard fornisce una guida per l'effettuazione di una valutazione formata da:

- Processo di valutazione;
- Modello per la valutazione;
- Strumenti per la valutazione.

Lo standard infine, stabilisce che per una corretta valutazione i verificatori debbano avere un buon livello di competenza e di esperienza.

B Resoconto delle attività di verifica

B.1 Dettaglio delle verifiche tramite analisi

B.1.1 Verifica della documentazione

Sono qui sotto riportati i risultati dei test di leggibilità effettuati sulla documentazione tramite l'utilizzo dell'indice Gulpease.

Nome documento	**Valore indice**	**Esito**
Analisi dei Requisiti v1.0.0	56	Superato
Glossario v1.0.0	50	Superato
Norme di Progetto v1.0.0	54	Superato
Piano di Progetto v1.0.0	55	Superato
Piano di Qualifica v1.0.0	55	Superato
Studio di Fattibilità v1.0.0	48	Superato

Tabella 1: Risultati indice Gulpease

C Pianificazione ed esecuzione del collaudo

Questa sezione sarà redatta quando il prodotto software sarà pronto per il collaudo.