TEAM
COMMITTED

Universitá degli Studi di Padova

Piano di Qualifica V1.0

Informazioni sul documento

Nome documento	Piano di Qualifica
Versione documento	V1.0
Data documento	16/12/2011
Redattori	• Giacomo Quadrio • Lorenzo Braghetto • Gabriele Facchin • Massimo Dalla Pieta'
Verificatori	Gabriele Facchin
Approvazione	Giorgio Maggiolo
Uso documento	Esterno
Lista distribuzione	• *Team Committed* • *Prof. Tullio Vardanega* • *Dr. Amir Baldissera*

Sommario

Questo documento vuol definire le norme volte alla regolamentazione delle fasi di verifica che il gruppo *Team Committed* ha deciso di adottare.

Diario delle modifiche

Modifica	Autore	Data	Versione
Approvato.	Giorgio Maggiolo	16/12/2011	V1.0
Piccole modifiche, adeguato alle Norme di Progetto	Massimo Dalla Pietà	16/12/2011	V0.92
Completato Resoconto attività di verifica capitolo 5	Massimo Dalla Pietà	15/12/2011	V0.91
Completato Gestione amministrativa della revisione capitolo 4	Massimo Dalla Pietà	14/12/2011	V0.9
Completato Obbiettivi di qualità capitolo 3	Gabriele Facchin	14/12/2011	V0.8
Completato Obbiettivi di qualità capitolo 3, sottocapitolo 3.1	Gabriele Facchin	13/12/2011	V0.7
Completato Visione generale delle strategie di verifica capitolo 2	Gabiele Facchin	13/12/2011	V0.6
Completato Strumenti e Tecniche capitolo 2 sottocapitolo 2.5	Lorenzo Braghetto	12/12/2011	V0.5
Completato Organizzazione capitolo 2 sottocapitolo 2.1	Giacomo Quadrio	10/12/2011	V0.4
Completato Risorse capitolo 2 sottocapitolo 2.4	Lorenzo Braghetto	10/12/2011	V0.4
Iniziata stesura Gestione amministrativa della revisione capitolo 4	Lorenzo Braghetto	09/12/2011	V0.3
Inizio stesura Visione generale delle strategie di verifica capitolo 2	Giacomo Quadrio	09/12/2011	V0.2
Inizio creazione del documento. Creato lo schema base e completata stesura - Introduzione capitolo 1	Quadrio Giacomo	09/12/2011	V0.1

Indice

1 Introduzione

1.1 Scopo del documento

Tale documento ha lo scopo di illustrare le strategie adottate per la verifica e validazione del lavoro svolto dal *Team Committed* relativamente al progetto *SafetyGame* e ai processi relativi alla sua produzione.

1.2 Scopo del prodotto

Il prodotto *SafetyGame* si propone di offrire una piattaforma alle aziende per poter rendere alcuni contesti critici come l'apprendimento di norme relative alla sicurezza sul lavoro, parte integrante delle attività lavorative standard. Ad oggi infatti, le tematiche sono affrontate tramite attività di formazione statiche, basate su insegnamenti in aula che poco motivano ed interessano i partecipanti. *SafetyGame* si prefigge quindi di aumentare l'interesse dei dipendenti di un'azienda tramite l'utilizzo di sfide e compiti da portare a termine che si andranno ad integrare con le attività lavorative reali. Tale scopo si raggiungerà tramite la somministrazione di sfide attraverso apposite applicazioni desktop, browser e mobile.

1.3 Glossario

Tutti i termini marcati da apposita sottolineatura saranno approfonditi e descritti all'interno del file `Glossario-v1.0.pdf` al fine di evitare ambiguità.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato d'appalto: Piattaforma mobile di apprendimento comportamentale, rilasciato dal proponente Mentis s.r.l. reperibile all'indirizzo `http://www.math.unipd.it/~tullio/IS-1/2011/Progetto/C3.pdf`
- Analisi dei requisiti (Allegato AnalisiDeiRequisiti_v1.0.0.pdf)
- Norme di progetto (Allegato NormeDiProgetto_v1.0.0.pdf)
- Verbale incontro proponente 2011-12-05 (Allegato Verbale05.12.2011.pdf)

1.4.2 Informativi

- ISO/IEC 15504:1998 (`www2.cnipa.gov.it/site/_contentfiles/00310300/310320_15504.pdf`)
- ISO/IEC 9126 (`http://it.wikipedia.org/wiki/ISO/IEC_9126`)
- ISO 14598

2 Visione generale delle strategie di verifica

2.1 Organizzazione

Per poter effettuare un corretto processo di verifica si è scelto di effettuare le dovute operazioni di controllo quando il prodotto sotto analisi avrà raggiunto uno stadio tale per cui risulta presentare sostanziali modifiche rispetto alla precedente versione. Lo svolgimento di tale fase sarà agevolato grazie all'apposito registro delle modifiche. Inoltre, per garantire una migliore verifica si è adottato il "**Broken Window Theory**" secondo il quale, non appena un errore viene rilevato, questo andrà segnalato e corretto il prima possibile onde evitarne la propagazione incontrollata.

Il ciclo di vita scelto per lo sviluppo del progetto è un ciclo di vita incrementale (vedi `Piano di Progetto - V1.0` per approfondimenti) e di conseguenza le operazioni di verifica verranno realizzate in modo tale da intervenire in maniera coerente nelle varie fasi del progetto come illustrato di seguito:

- **Analisi dei requisiti**: ogni documento necessario alla RR, quando verrà completato, entrerà nella dedicata fase di revisione per controllare la presenza di eventuali irregolarità lessico/grammaticali e nei contenuti esposti. Nel dettaglio, il controllo ortografico verrà effettuato con gli strumenti messi a disposizione da LyX, per la precisione tramite il plugin **Aspell**, mentre il controllo lessicale, grammaticale e sintattico da un'accurata rilettura del testo. I contenuti verranno invece controllati in modo tale da verificare la copertura delle richieste del proponente e questo tramite un'accurata rilettura e confronto con il capitolato d'appalto e che ogni requisito abbia il suo caso d'uso corrispondente. Verranno verificati inoltre i contenuti grafici e tabellari e la conformità dei documenti alle Norme di Progetto stabilite. Se durante la verifica saranno state rilevate irregolarità queste verranno segnalate tramite un apposito ticket dal verificatore e corrette dal redattore. Il processo di verifica si concluderà quindi con la validazione del documento da parte del verificatore e l'approvazione da parte del responsabile per la presentazione al committente

- **Progettazione**: il processo di verifica riguardante la fase di Progettazione consisterà nel verificare che tutti i requisiti descritti durante la fase di Analisi dei Requisiti rientrino nei componenti individuati. Qualora dalla verifica sorgano incongruenze o mancanze, queste verranno segnalate tramite ticket e successivamente risolte

- **Realizzazione**: la verifica in questa fase verrà effettuata sia da parte dei programmatori stessi che utilizzando appositi e specifici strumenti di verifica automatizzata del codice. La presenza di errori verrà segnalata da un apposito ticket che verrà preso in carico dai programmatori e chiuso una volta risolto il problema

- **Validazione**: il team si impegna a fornire al collaudo una versione funzionante e possibilmente completa del prodotto. Eventuali difetti o incon-

gruenze saranno corretti ed eliminati da Team Committed

2.2 Pianificazione strategica e temporale

La strategia di verifica adottata prevede, come scritto in sezione 2.1, una verifica ogni qualvolta il prodotto da noi realizzato abbia raggiunto uno stadio tale che si differenzi dalla sua precedente versione. Questa, unita alla tecnica "*Broken Window Theory*", cerca di garantire un alto livello di qualità al fine di raggiungere il totale soddisfacimento dei requisiti richiesti dal capitolato d'appalto oltre che del cliente. Il *Responsabile di Progetto* avrà quindi il compito di coordinare e definire le attività volte alla verifica del materiale prodotto, sia esso software, documenti o materiale d'altro genere, oltre che a far rispettare le scadenze previste.

2.3 Responsabilità

Al *Responsabile di Progetto* saranno affidate le responsabilità riguardanti tutte le attività di verifica e validazione, ponendosi quindi come garante del corretto svolgimento delle attività volte alla qualità del materiale prodotto nei confronti del committente. L'*Amministratore di Progetto* si occuperà invece di assicurare che l'ambiente in cui tutte le attività di realizzazione del prodotto si svolgeranno sia adeguato a tale scopo.

2.4 Risorse

La gestione della qualità prevede l'utilizzo di alcune risorse

2.4.1 Risorse Necessarie

Risorse Umane

I ruoli necessari per garantire un'adeguata qualità sono i seguenti:

- **Responsabile** è il referente e responsabile nei confronti del committente, supervisiona i processi interni e si preoccupa di valutare le proposte di modifica correttiva o migliorativa dei Verificatori
- **Amministratore** definisce i piani per la gestione della qualità, come i processi di verifica, test e individuazione e risoluzione delle anomalie e discrepanze
- **Verificatore** esegue le attività di verifica sul prodotto a seconda delle norme redatte, ne riassumerà gli esiti e in caso di discrepanze presenterà il problema al Responsabile
- **Programmatore** è incentivato a eseguire attività di debugging quando ne riterrà necessario ed è coinvolto dai verificatori nella risoluzione dei ticket

Risorse Software

- **Software per la verifica dei requisiti richiesti**
- **Software di gestione e commento di ticket, codice e documentazione**
- **Software necessario alla corretta configurazione dell'ambiente di sviluppo**
- **Software utile alla comunicazione fra i membri del team**

Risorse Hardware

- **Computer su cui eseguire il software necessario all'ambiente di sviluppo**
- **Luogo fisico per incontri fra i membri del team**

2.4.2 Risorse Disponibili

Risorse Software

- **Github e correlati:** risorse messe a disposizione dal servizio web Github per gestire commenti o problemi a codice, documentazione e ticket
- **Aspell** per il controllo ortografico dei documenti
- **Eclipse** con relativi strumenti già inclusi o aggiunti tramite plugin
- **Strumenti W3C** per validazione web
- **Gruppo Facebook** per poter comunicare fra membri del team

Risorse Hardware

- **Computer personali, portatili e fissi**
- **Computer messi a disposizione nei laboratori dal Dipartimento di Matematica Pura ed Applicata dell'Università di Padova**
- **Aule studio del Dipartimento di Matematica Pura ed Applicata dell'Università di Padova**

2.5 Strumenti, tecniche e metodi

2.5.1 Strumenti

Il gruppo potrà avvalersi dei seguenti strumenti per effettuare i processi di verifica:

- **Aspell** ($\geqslant$ 0.60.6): strumento per la correzione grammaticale dei documenti redatti. L'utilizzo di Aspell avverrà tramite apposito plugin per LyX (`http://aspell.net/`)

- **Eclipse** ($\geqslant$ 3.7.1): IDE multi-linguaggio e multi-piattaforma che mette a disposizione alcune funzionalità di debugging quali l'esecuzione step-by-step, l'impostazione di beaking point ecc (`http://www.eclipse.org/`)
- **FindBugs** ($\geqslant$ 2.0.0): strumento utilizzato nell'analisi statica e volto ad individuare errori nella scrittura di codice Java (`http://findbugs/sourceforge.net/`)
- **EclEmma** ($\geqslant$ 1.5.3): strumento in grado di determinare la copertura del codice Java prodotto sia durante le fasi di esecuzione che di testing e disponibile sotto forma di plugin per *Eclipse* (`http://www.eclemma.org/`)
- **Android Lint** ($\geqslant$ 16.0.0)**:** strumento utilizzato per individuare i principali errori di programmazione all'interno di applicazioni Android (`http://tools.android.com/tips/lint`)
- **Metrics** ($\geqslant$ 1.3.6): plugin per *Eclipse* che permette di effettuare analisi metrica del codice prodotto. Fornisce informazioni relative a misure statiche del codice:
  - **Complessità ciclomatica**
  - **Peso delle classi**
  - **Numero di parametri**
  - **Numero di campi dati per classe**
  - **Numero livelli di annidamento**
  - **Indice di utilità - Indice di dipendenza**

  (`http://metrics.sourceforge.net`)
- **JUnit** ($\geqslant$ 4.10): Framework utilizzato per effettuare test di unità per il linguaggio Java (`http://www.junit.org/`)
- **Selenium IDE** ($\geqslant$ 1.4.1): plugin per *Firefox* utilizzato per registrare ed eseguire test tramite browser (`http://seleniumhq.org/`)
- **ApacheBrench** ($\geqslant$ 2.0): strumento a linea di comando utilizzato per misurare l'efficienza di un server web ed in grado di simulare situazioni di sovraccarico della rete (`http://www.apache.org/`)
- **SpeedTracer** ($\geqslant$ 2.4): plugin per *Google Chrome* che permette di verificare l'efficienza di un'applicazione web durante la sua esecuzione (`http://code.google.com/intl/it-IT/webtoolkit/speedtracer/`)
- **Strumenti di validazione W3C**:
  - **Markup Validation Service**: per effettuare test sulle pagine di cui l'applicativo web è costituita e verificarne l'aderenza agli standard HTML5 (`http://validator.w3.org/`)

– **CSS Validation Service**: per effettuare test sui fogli di stile utilizzati nell'applicativo web e verificarne l'aderenza allo standard CSS 2.1 (`http://jigsaw.w3.org/css-validator/`)

2.5.2 Tecniche

Analisti statica L'analisi statica consisterà nella verifica critica del codice, controllandone requisiti e la corretta progettazione.

Questo tipo di analisi è usualmente fatta con due tecniche complementari:

1. **Walkthrough**: consiste in un'ispezione generale del codice o dei documenti senza prerequisiti iniziali e pianificazione. È utile nelle fasi iniziali di verifica quando va considerato il software nel suo intero senza sapere quali saranno le parti più critiche

2. **Inspection**: consiste in una verifica mirata del codice o dei documenti, è possibile solo quando si è raggiunto un buon grado di conoscenze e si è già consapevoli degli errori più comuni o delle aree più critiche

Chiaramente la tecnica dell'*inspection* è più efficace, ma sarà possibile attuarla solo dopo aver appreso la consapevolezza iniziale del proprio codice e delle proprie capacità.

Metodi di Analisi Statica:

- **Analisi del flusso di controllo**: si accerta che il codice segua il flusso aspettato, che non si possa entrare in porzioni di codice che possano non terminare, che non esista codice non raggiungibile

- **Analisi del flusso dei dati**: si accerta che il software non acceda mai a variabili non inizializzate o scriva inutilmente più volte prima di usare una variabile

- **Analisi del flusso di informazione**: verifica che gli input e gli output di ogni unità di codice o di più unità rientrino nelle specifiche del programma

Analisi dinamica L'analisi dinamica consisterà nella verifica dei componenti del software o del sistema in generale. Verranno effettuati test mirati e ripetuti in diversi contesti. Il programmatore, durante i test, dovrà essere cosciente dello stato dell'ambiente, degli input e dei risultati in ogni momento dell'esecuzione. Questi risultati saranno utili solo nel caso vengano trovati errori da correggere, nel caso non vengano trovati, non significa necessariamente che non ve ne siano.

Metodi di Analisi Dinamica

- **Test di unità**: test che si effettuano per ogni unità del software con il massimo grado di parallelismo

- **Test di integrazione:** verifica dei componenti formati dall'integrazione delle varie unità che hanno passato il test di unità
- **Test di sistema e di collaudo**: verifica che il sistema in cui andrà installato il software rispetti i requisiti richiesti, o che il software riesca ad adattarsi correttamente al contesto dell'azienda proponente. Il collaudo sarà sul software installato, finito il quale avverrà il rilascio del prodotto
- **Test di regressione**: nel caso di una modifica ad un singolo componente, andranno effettuati nuovamente tutti i test di unità e, se necessario, di integrazione riferiti a quel componente

2.5.3 Misure e Metriche

Le misure e le metriche che adotteremo per la verifica della qualità del software si ispireranno alle indicazioni dello standard *ISO-14598*. Sono misure spesso focalizzate al miglioramento della capacità di prevedere e contenere il costo del software, Il *Team Committed* le utilizzerà per valutare la qualità del prodotto sia nel processo di progettazione che di codifica.

- **Complessità ciclomatica:** La complessità ciclomatica di un metodo è il numero di cammini linearmente indipendenti attraverso il codice sorgente. Per esempio, se il codice sorgente non contiene if o for, allora il livello di complessità sarà 1, poiché esiste un solo cammino
- **Misure nella progettazione:** permettono di stimare il costo di un software e la sua qualità.
  - **Numero di classi, coesione tra di esse e peso**. Il peso di una classe è identificato dalla somma della complessità ciclomatica di tutti i metodi appartenenti alla classe
  - **Complessità di flusso**: misura la quantità di informazioni in entrata ed uscita da una funzione (fan in e fan out)
- **Misure sul codice**
  - **Linee di codice**
  - **Misure di coesione funzionale**: misurano le istanze di definizione e utilizzo di variabili e costanti (1994). Un numero elevato di parametri può essere abbassato, e quindi migliorato, raggruppando parametri tra loro correlati in classi diverse, aumentando manutenibilità e astrazione del codice
  - **Livello di copertura di istruzioni, dei rami, dei percorsi base.** Una non buona copertura del codice è indice di scarsa qualità dello stesso, inoltre vanno evitati eccessivi annidamenti dei metodi

3 Obbiettivi di qualità

Il *Team Committed* ha ritenuto importante fissare, di comune accordo, alcuni obbiettivi di qualità da perseguire e raggiungere sia nei processi di realizzazione che del prodotto in se, questo per garantire una migliore e più efficace soddisfazione dei requisiti richiesti nel capitolato d'appalto.

3.1 Qualità dei processi

La realizzazione di prodotti di buona qualità richiede che vi sia qualità anche nei processi necessari. Il *Team Committed* ha quindi deciso di adottare lo standard **ISO/IEC 15504:1998 SPICE** (*Software Process Improvement and Capability Determination*) che definisce il modello denominato **SPA-I** (*Software Process Assessment & Improvement*) che offre metodi per la valutazione ed aumento di qualità dei processi.

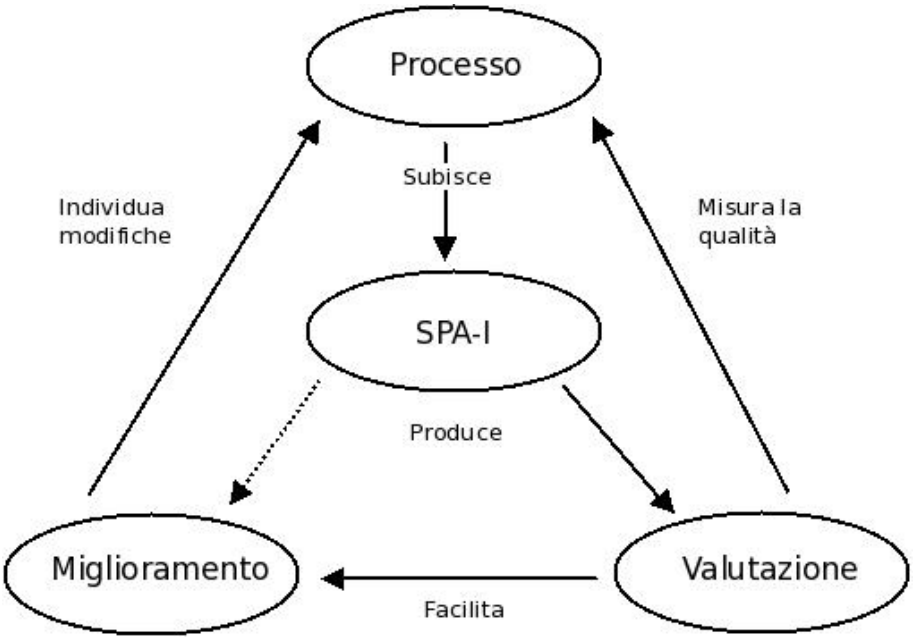

Figure 1: *Schema SPA-I*

Tutti i processi dovranno quindi essere sottoposti a valutazione in modo da verificarne la qualità ed eventualmente facilitarne il miglioramento. A tale scopo lo *SPICE* definisce nove attributi di processo per effettuare una migliore valutazione:

1. **Process performance**: un processo raggiunge i suoi obiettivi nel momento in cui trasforma dell'input identificabile in output identificabile

2. **Performance Management**: l'attuazione di un processo è pianificata e controllata al fine di generare risultati che rispondono agli obiettivi attesi
3. **Work Product Management**: l'attuazione di un processo è pianificata e controllata al fine di generare risultati che siano adeguatamente documentati, controllati e verificati
4. **Process Definition**: l'attuazione di un processo si basa su approcci standardizzati
5. **Process Resource**: il processo può contare sulle adeguate risorse umane, d'infrastrutture ecc. per essere attuato
6. **Process Measurement:** i risultati conseguiti e le misure rilevate durante l'attuazione di un processo sono utilizzati per assicurarsi che l'attuazione di tale processo supporti efficacemente il raggiungimento di obiettivi specifici
7. **Process Control**: un processo è controllato tramite la raccolta, analisi ed utilizzo delle misure di prodotto e di processo rilevate, con l'obbiettivo di correggere, se necessario, le sue modalità di attuazione
8. **Process Change**: le modifiche alla definizione, gestione e attuazione di un processo sono controllate
9. **Continuous Integration**: le modifiche ad un processo sono identificate ed implementate con lo scopo di assicurare il continuo miglioramento nel raggiungere gli obbiettivi rilevati per l'organizzazione

La norma stabilisce poi quattro differenti livelli di possesso di ciascuno degli attributi:

- **N - Non posseduto** (0 - 15% di possesso): non c'è evidenza oppure ce n'è poca del possesso di un attributo
- **P - Parzialmente posseduto** (16 - 50% di possesso): c'è evidenza di approccio sistematico al raggiungimento del possesso di un attributo e del raggiungimento di tale possesso, ma alcuni aspetti del possesso possono essere non prevedibili
- **L - Largamente posseduto** (51 - 85% di possesso): vi è evidenza di approccio sistematico al raggiungimento del possesso di un attributo e di un significativo livello di possesso di tale attributo, ma l'attuazione del processo può variare nelle diverse unita operative della organizzazione
- **F - (Fully) Pienamente posseduto** (86 - 100% di possesso): vi è evidenza di un totale e sistematico approccio e di un completo raggiungimento del possesso dell'attributo e non esistono significative differenze nel modo di attuare il processo tra le diverse unità operative

Vi sono poi vari livelli di maturità dei processi che sono dati dal diverso livello di possesso degli attributi:

- **Livello 0 - Processo incompleto**: il processo non è implementato o non raggiunge gli obiettivi. Non vi è evidenza di approcci sistematici agli attributi definiti
- **Livello 1 - Processo semplicemente attuato**: il processo viene messo in atto e raggiunge i suoi obiettivi. Non vi è evidenza di approcci sistematici agli attributi definiti. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "Process performance"
- **Livello 2 - Processo gestito**: il processo è attuato, ma anche pianificato, tracciato, verificato ed aggiustato se necessario, sulla base di obiettivi ben definiti. Il raggiungimento di questo livello è dimostrato attraverso il possessi degli attributi di "Performance management" e "Work product management"
- **Livello 3 - Processo definito**: il processo è attuato, pianificato e controllato sulla base di procedure ben definite, basate sui principi del software engineering. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "Process definition" e "Process resource"
- **Livello 4 - Processo predicibile**: il processo è stabilizzato ed è attuato all'interno di definiti limiti riguardo i risultati attesi, le performance, le risorse impiegate ecc. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "Process measurement" e "Process control"
- **Livello 5 - Processo ottimizzante**: il processo è predicibile ed in grado di adattarsi per raggiungere obiettivi specifici e rilevanti per l'organizzazione. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "Process change" e "Continuous integration"

L'applicazione dello standard *ISO/IEC 15504* porta a benefici sia agli sviluppatori del software che ai suoi utilizzatori o acquirenti. Per gli sviluppatori porta vantaggi nell'ottimizzazione dell'uso delle risorse, un contenimento dei costi, una maggiore tempestività di consegna del prodotto ultimato, migliore stima dei rischi e degli impegni e la possibilità di confrontarsi con delle best practice. Per gli utenti invece abbiamo una maggior facilità nella selezione dei fornitori, una migliore valutazione dei rischi di progetto, controllo dello stato di avanzamento in corso d'opera, riduzione dei costi di correzione degli errori ed un controllo dei rischi e delle varianti in corso d'opera.

3.2 Qualità del prodotto

Per quanto concerne la qualità del prodotto si è scelto di seguire alcune linee guida dettate dallo standard *ISO/IEC 9126*, classificato da sei caratteristiche generali e varie sotto-caratteristiche.

3.2.1 Funzionalità

Il prodotto software realizzato deve essere in grado di offrire apposite funzionalità che siano in grado di soddisfare le esigenze stabilite, necessarie per operare sotto condizioni specifiche.

- **Appropiatezza**: costituisce la capacità del software di offrire un insieme di funzioni per i compiti ed obbiettivi prefissati all'utente
- **Accuratezza**: capacità del software di fornire i risultati concordati o i precisi effetti richiesti
- **Interoperabilità**: capacità del prodotto software di interagire ed operare con specifici sistemi
- **Conformità**: capacità del prodotto software di aderire a standard, convenzioni e regolamentazioni rilevanti al settore operativo a cui vengono applicate
- **Sicurezza**: la capacità del prodotto software di proteggere informazioni e dati impedendo che persone o sistemi non autorizzati possano accedervi o modificarli, mentre garantire queste operazioni a utenti o sistemi autorizzati

La misurazione del raggiungimento di questo obiettivo si calcolerà verificando la quantità di requisiti soddisfatti che avranno un riscontro in elementi funzionanti nell'applicazione prodotta. La soglia di sufficienza sarà quindi data dal soddisfacimento di tutti i requisiti obbligatori previsti dal capitolato d'appalto.

3.2.2 Affidabilità

L'affidabilità misura la capacità di un prodotto software di mantenere un determinato livello di prestazioni se usato in determinate condizioni e per un certo periodo.

- **Maturità**: capacità del prodotto software di evitare che si verifichino errori, malfunzionamenti o siano prodotti risultati non coerenti
- **Tolleranza agli errori**: capacità di mantenere determinati livelli di prestazioni nonostante la presenza di errori, malfunzionamenti o usi scorretti del prodotto
- **Recuperabilità**: capacità di un prodotto di ripristinare il livello appropriato di prestazioni e di essere in grado di recuperare le informazioni rilevanti in seguito ad un malfunzionamento
- **Aderenza**: capacità di aderire a standard, regole e convenzioni inerenti l'affidabilità

La misurazione del raggiungimento di questo obiettivo si calcolerà confrontando il numero di esecuzioni totale con quelle andate a buon fine e che hanno mantenuto un livello di prestazioni tali da poter permettere l'utilizzo previsto del prodotto.

3.2.3 Efficienza

La capacità di fornire adeguate prestazioni in relazione alla quantità di risorse usate determina l'efficienza.

- **Comportamento rispetto al tempo**: capacità di fornire adeguati tempi di risposta, elaborazione e velocità, sotto condizioni determinate
- **Utilizzo delle risorse**: capacità di utilizzare in maniera adeguata la giusta quantità e tipologia di risorse
- **Conformità**: capacità di aderire a standard e specifiche sull'efficienza

La misurazione del raggiungimento di questo obiettivo sarà determinata dal tempo necessario per ottenere una risposta dal servizio sia in condizioni normali che in condizioni di sovraccarico.

3.2.4 Usabilità

La capacità di un prodotto software di essere capito, appreso e usato dall'utente in certe condizioni determina la sua usabilità.

- **Comprensibilità**: costituisce la facilità di comprensione dei concetti del prodotto, permettendo all'utente quindi di comprendere se il software è appropriato
- **Apprendibilità**: capacità di diminuire l'impegno richiesto agli utenti per imparare ad utilizzare l'applicazione
- **Operabilità**: capacità di porre gli utenti in condizioni tali da utilizzare il prodotto per i propri scopi e controllarne l'uso
- **Attrattiva**: capacità del software di essere piacevole per l'utente
- **Conformità**: capacità del software di aderire a standard o convenzioni relativi all'usabilità

La misurazione del raggiungimento di questo obiettivo sarà costituita dalla capacità dell'applicativo di adattarsi ai vari tipi di ambienti in cui esso verrà eseguito come ambienti desktop o dispositivi mobile. L'usabilità sarà poi ritenuta raggiunta fornendo un'interfaccia il più possibile chiara, semplice ed intuitiva e rendendo l'applicativo fruibile a quelle categorie di utenti affetti da disabilità come daltonici o ipovedenti.

3.2.5 Manutenibilità

La manutenibilità rappresenta la capacità del software di essere modificato, includendo correzioni, miglioramenti o adattamenti.

- **Facilità di analisi**: rappresenta la facilità con la quale è possibile analizzare il codice per localizzare un eventuale errore
- **Modificabilità**: capacità del prodotto software di permettere l'implementazione di una specificata modifica
- **Stabilità**: capacità del software di evitare effetti inaspettati derivanti da modifiche errate
- **Testabilità**: capacità del software di essere facilmente testato per validare le modifiche apportate

La misurazione del raggiungimento di questo obiettivo saranno legate al rispetto delle misure metriche descritte nel capitolo 2.5.3

3.2.6 Portabilità

La portabilità è la capacità di un software d'essere portato da un ambiente di lavoro ad un altro.

- **Adattabilità**: capacità del software di essere adattato per differenti ambienti operativi senza dover applicare modifiche diverse da quelle fornite per il software considerato
- **Installabilità**: capacità del software di essere installato in uno specifico ambiente
- **Conformità**: capacità del software di aderire a standard e convenzioni relative alla portabilità
- **Sostituibilità**: capacità di essere utilizzato al posto di un altro software per svolgere gli stessi compiti

L'applicazione desktop deve essere installabile in tutti i principali sistemi operativi disponibili (Mac OSX, Windows, GNU Linux). Il front-end deve essere compatibile e funzionante con tutti i browser che aderiscono al W3C e con i browser di dispositivi mobile di diverso tipo (iOS, Windows Phone, Android, Symbian). Infine l'applicazione Android deve essere compatibile con il maggior numero di distribuzioni possibile del sistema operativo.

4 Gestione amministrativa della revisione

4.1 Comunicazione e risoluzione di anomalie

Un'anomalia consiste in un comportamento non coerente con le aspettative di qualità prefissate.

Lo strumento scelto per la gestione delle anomalie è la sezione "*Issue*" messa a disposizione da Github. Coerentemente con l'organizzazione generale delle strategie di verifica, nuove anomalie potranno essere scoperte in due modi:

1. Al rilascio di ogni nuova versione con relativo registro delle modifiche, il Verificatore effettuerà una fase di verifica cercando eventuali anomalie.
2. Grazie all'approccio "*Broken Window Theory*", chiunque in qualunque momento è incentivato alla ricerca di possibili anomalie.

Appena un'anomalia viene trovata è opportuna una segnalazione tempestiva attraverso i ticket.

Struttura di un ticket

La struttura di un ticket dovrà seguire le seguenti specifiche:

- **Nome**
- **Descrizione**
- **Label**
- **Milestone**
- **Assegnee**

A riguardo è possibile leggere in dettaglio le specifiche di un ticket nelle `Norme di Progetto - V1.0`.

Risoluzione di un ticket Le modalità di risoluzione e chiusura di un ticket sono descritte nelle `Norme di Progetto - V1.0`, sono volte a garantire la massima capacità di coordinamento fra i vari membri del gruppo garantendo un'attenzione alla qualità generale.

4.2 Trattamento delle discrepanze

Una **discrepanza** è un tipo di anomalia non grave, non concerne il corretto funzionamento del prodotto, ma può riguardare un allontanamento dai requisiti attesi specificati nel capitolato d'appalto oppure una violazione delle *Norme di Progetto*.

Le modalità di ricerca e comunicazione delle discrepanze è del tutto simile alle modalità specificate per le anomalie; la risoluzione presenta invece modalità differenti.

Una volta creato il ticket sarà compito del verificatore riconoscere quale tipo di discrepanza si tratta e nel caso riguardi una violazione delle *Norme di Progetto* provvederà a comunicare il problema all'Amministratore che prenderà provvedimenti. Se invece il problema riguarda un allontanamento dai requisiti, una volta identificata l'origine della discrepanza. il Verificatore solleciterà l'Analista per valutare la gravità e i costi per risolverla.

4.3 Procedure di controllo di qualità di processo

Le procedure di gestione della qualità che il gruppo *Team Committed* userà si basano sul concetto di "**ciclo di Deming**" o **PDCA**, utili a garantire un continuo processo di miglioramento basandosi sulla comunicazione attiva da parte delle varie componenti del gruppo, una continua connessione fra le fasi di analisi, progettazione, produzione e verifica o collaudo. Lo scopo è un continuo miglioramento dei processi qualitativi unito ad una riduzione di costi e sprechi.

Il concetto è molto vicino al termine **Kaizen**, comunemente tradotto con "*miglioramento continuo*", si riferisce ai processi di miglioramento qualitativo che devono coinvolgere tutto il gruppo di persone al lavoro. Si basa sulla filosofia de "*L'energia viene dal basso*", ovvero sull'idea che il processo qualitativo non dipenda solo dall'organizzazione di un progetto, ma anche sul lavoro fatto sui singoli processi e prodotti.

5 Resoconto attività di verifica

5.1 Revisione dei requisiti

Nella fase di Revisione dei Requisiti è stata effettuata un'attività di verifica sui documenti prodotti. Nel dettaglio è stato effettuata un'analisi statica come indicato nella sezione 2.5.2, effettuando le dovute procedure per la correzione degli errori rilevati. Si è effettuato quindi il controllo ortografico tramite Aspell, il plugin per LyX, mentre il controllo grammaticale, sintattico e lessicale è avvenuto tramite un'accurata rilettura da parte dei revisori. Sono stati poi controllati anche i contenuti tabellari e grafici. La segnalazione di irregolarità è avvenuta tramite Ticket che sono state prese in carico successivamente dal redattore e risolte.