
TEAM
COMMITTED

Università degli Studi di Padova

# Piano di Qualifica V3.0

# Informazioni sul documento

| | |
|---|---|
| **Nome documento** | Piano di Qualifica |
| **Data documento** | 2012/06/16 |
| **Redattori** | • Giacomo Quadrio<br>• Lorenzo Braghetto<br>• Gabriele Facchin<br>• Massimo Dalla Pietà<br>• Alessandro Cornaglia<br>• Marco Begolo |
| **Verificatori** | • Massimo Dalla Pietà<br>• Gabriele Facchin<br>• Lorenzo Braghetto |
| **Approvazione** | • Giorgio Maggiolo<br>• Giacomo Quadrio |
| **Uso documento** | Esterno |
| **Lista distribuzione** | • *Team Committed*<br>• *Prof. Tullio Vardanega*<br>• *Prof. 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.* | Giacomo Quadrio | 2012/06/16 | V3.0 |
| *Verificato. In attesa di approvazione.* | Lorenzo Braghetto | 2012/06/15 | V2.3 |
| *Conclusa redazione programma di test e modifiche richieste. In attesa di verifica* | Marco Begolo | 2012/06/14 | V2.2 |
| *Inserito programma di test* | Alessandro Cornaglia | 2012/02/15 | V2.1 |
| *Approvato.* | Giorgio Maggiolo | 2012/01/23 | V2.0 |
| *Verificato. In attesa di approvazione* | Gabriele Facchin | 2012/01/21 | V1.8 |
| *Inserito capitolo 5: Pianificazione ed esecuzione del collaudo - Test di sistema* | Giacomo Quadrio | 2012/01/20 | V1.7 |
| *Inserita sezione 7.2: Revisione di progettazione* | Lorenzo Braghetto | 2012/01/20 | V1.6 |
| *Inserito capitolo 5: Pianificazione ed esecuzione del collaudo - Test di integrazione* | Lorenzo Braghetto | 2012/01/19 | V1.5 |
| *Inserito capitolo 2.5.3: Strategie per la verifica* | Giacomo Quadrio | 2012/01/18 | V1.4 |
| *Ampliate tecniche inspection del codice* | Lorenzo Braghetto | 2012/01/17 | V1.3 |
| *Ampliate tecniche inspection dei documenti* | Lorenzo Braghetto | 2012/01/17 | V1.2 |
| *Ampliato e corretto capitolo 5.1* | Giacomo Quadrio | 2012/01/16 | V1.1 |
| *Approvato.* | Giorgio Maggiolo | 2011/12/16 | V1.0 |
| *Piccole modifiche, adeguato alle Norme di Progetto. Revisione completata.* | Massimo Dalla Pietà | 2011/12/16 | V0.11 |
| *Completato Resoconto attività di verifica capitolo 5* | Massimo Dalla Pietà | 2011/12/15 | V0.10 |
| *Completato Gestione amministrativa della revisione capitolo 4* | Massimo Dalla Pietà | 2011/12/14 | V0.9 |
| *Completato Obbiettivi di qualità capitolo 3* | Gabriele Facchin | 2011/12/14 | V0.8 |
| *Completato Obbiettivi di qualità capitolo 3, sottocapitolo 3.1* | Gabriele Facchin | 2011/12/13 | V0.7 |
| *Completato Visione generale delle stretegie di verifica capitolo 2* | Gabriele Facchin | 2011/12/13 | V0.6 |
| *Completato Strumenti e Tecniche capitolo 2 sottocapitolo 2.5* | Lorenzo Braghetto | 2011/12/12 | V0.5 |
| *Completato Organizzazione capitolo 2 sottocapitolo 2.1* | Giacomo Quadrio | 2011/12/10 | V0.4 |
| *Completato Risorse capitolo 2 sottocapitolo 2.4* | Lorenzo Braghetto | 2011/12/10 | V0.4 |
| *Iniziata stesura Gestione amministrativa della revisione capitolo 4* | Lorenzo Braghetto | 2011/12/09 | V0.3 |
| *Inizio stesura Visione generale delle strategie di verifica capitolo 2* | Giacomo Quadrio | 2011/12/09 | V0.2 |
| *Inizio creazione del documento. Creato lo schema base e completata stesura - Introduzione capitolo 1* | Quadrio Giacomo | 2011/12/09 | 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-v3.0.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 `Analisi_Dei_Requisiti_v3.0.pdf`)
- **Norme di progetto** (Allegato `Norme_Di_Progetto_v3.0.pdf`)
- **Verbale incontro proponente 2011/12/05** (Allegato `Verbale05.12.2011.pdf`)
- **Verbale incontro proponente 2012/01/26** (Allegato `Verbale20120126.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** (`http://webstore.iec.ch/preview/info_isoiec14598-1%7Bed1.0%7Den.pdf`)

# 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 - V2.0* 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 ed i verbali degli incontri. Si constaterà poi che ogni caso d'uso abbia almeno un requisito corrispondente, effettuando un controllo delle apposite tabelle di tracciamento. 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; ciò verrà effettuato tramite un'accurata analisi delle tabelle di tracciamento. 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, come elencato nella sezione 2.5.1 dove si possono visionare quelli adottati da Team Committed con una relativa

descrizione della loro funzione. 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 incongruenze 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
- **BlueJ** con i relativi strumenti già inclusi
- **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, quelli che seguiranno saranno gli strumenti utili alle fasi di verifica e vengono presi in considerazione per la loro utilità in questa fase.

- **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/`)
- **BlueJ** ($\geqslant$ 3.0.7): editor realizzato in Java dall'università di Kent per la scrittura di classi Java, fornisce un'ottima interfaccia grafica generale per le classi, tramite UML e un comodo strumento per creare oggetti, utilizzare i metodi della classe di quell'oggetto o di quelle a lui connesse, ottimo per effettuare i test di unità e di integrazione (`http://www.bluej.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. Ogni attività svolta verrà registrata e documentata.

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

- **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. Ogni problema rilevato verrà discusso fra i verificatori e gli sviluppatori (tutti quelli che hanno realizzato la parte di prodotto in questione)
- **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. Riguardo l'Inspection dei documenti seguiremo questi punti:
  - Latex:
    - Impostazione generale del documento affinchè aderisca alle Norme di Progetto
    - Controllo della sillabazione automatica
    - Controllo punteggiatura e leggibilità del testo
    - Presenza di una e una sola spaziatura dopo la punteggiatura
    - Posizionamento corretto delle immagini all'interno della pagina
    - Accentazioni corrette (scelta corretta tra acuti e gravi)

    - Lettere maiuscole accentate
    - Termini definiti nel glossario ma non sottolineati nel documento
    - Termini sottolineati nel documento ma non presenti nel glossario
    - Sottolineatura di tutti i termini che sono stati sottolineati almeno una volta all'interno del documento
  - UML:
    - Mantenimento della consistenza tra la nomenclatura dei grafici UML e la nomenclatura adottata nelle sezioni in cui i grafici saranno inseriti
    - Controllo ortografico dettagliato, data l'impossibilità di automatizzare tali controlli sui grafici
    - Attenzione alla direzione delle frecce nei diagrammi
    - Attenzione alla perfetta orizzontalità o verticalità di frecce orizzontali o verticali
    - Controllo del nome del sistema
    - Posizione degli include e degli extend
    - Eliminazione di caratteri extra, non previsti dalle norme, creati da Bouml

Per quanto riguarda invece l'inspection del codice, seguiremo alcuni punti guida:

- Le variabili di programma devono essere inizializzate prima che il loro valore sia utilizzato.
- Tutte le costanti devono avere un nome.
- Il limite superiore degli array deve essere uguale alla loro dimensione o alla dimensione-1.
- Controllare la presenza di eventuali buffer overflow.
- Verificare che la condizione delle istruzioni condizionali sia corretta.
- Ogni ciclo deve essere ultimato.
- Le istruzioni composte devono essere correttamente messe fra parentesi.
- Le variabili di input devono essere tutte utilizzate.
- I break devono essere correttamente utilizzati se richiesti.
- Le variabili di input devono essere tutte utilizzate.
- Le variabili di output devono possedere un valore prima che vengano restituite.
- Verificare gli effetti degli input imprevisti.
- Le chiamate a funzione e a metodo devono possedere il corretto numero di parametri.

- Il tipo dei parametri formali e reali deve corrispondere.
- Corretta disposizione dei parametri.
- Verificare che siano state prese in considerazione tutte le possibili condizioni di errore.

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 Strategie per la verifica

Le strategie per effettuare operazioni di verifica prevedono quattro fasi principali che verranno di seguito descritte:

- **Verifica di accettabilità:** per l'accettazione, Team Committed si prefiggerà di testare che le funzionalità illustrate all'interno dei requisiti siano coperte dagli appositi test di unità ed integrazione oltre che da prove funzionali sul sistema.
- **Verifica del disegno architetturale:** Dopo aver realizzato lo schema architetturale del prodotto verrà ferificato che siano rispettati alcuni precisi principi di seguito elencati:
  - Layering: le classi devono rispettare una struttura gerarchica
  - Packaging: le classi devono essere accorpate con una certa logica
  - Coesione: una unità è progettata in modo tale da rispondere unicamente ai requisiti ad essa associati nel tracciamento dei requisiti
  - Accoppiamento ridotto: le classi saranno progettate in modo tale che siano il più possibile indipendenti

  Questa fase di verifica verrà eseguita anche grazie all'ausilio di apposite metriche descritte in "Misure e Metriche" ( 2.5.4)
- **Verifica del codice:** La verifica del codica ha diversi obiettivi, il primo può essere identificato nel trovare, segnalare ed isolare quelle parti di codice che non sono conformi alle norme stabilite in precedenza nel documento Norme di progetto, attraverso una fase di analisi statica. Un altro obiettivo può essere l'identificazione di quelle parti di codice soggette ad errori di programmazione mediante analisi dinamica.
- **Verifica di performance e criticità:** le performance e la criticità sono elementi che dipendono fortemente dall'infrastruttura di rete e dai server su cui risiederà l'applicazione. Team Committed si impegna quindi ad effettuare le dovute misurazioni riguardanti il carico di lavoro e la stabilità del sistema nel caso di picchi di richieste, questo per verificare la robustezza e reattività del sistema.

### 2.5.4 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, poichè 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.

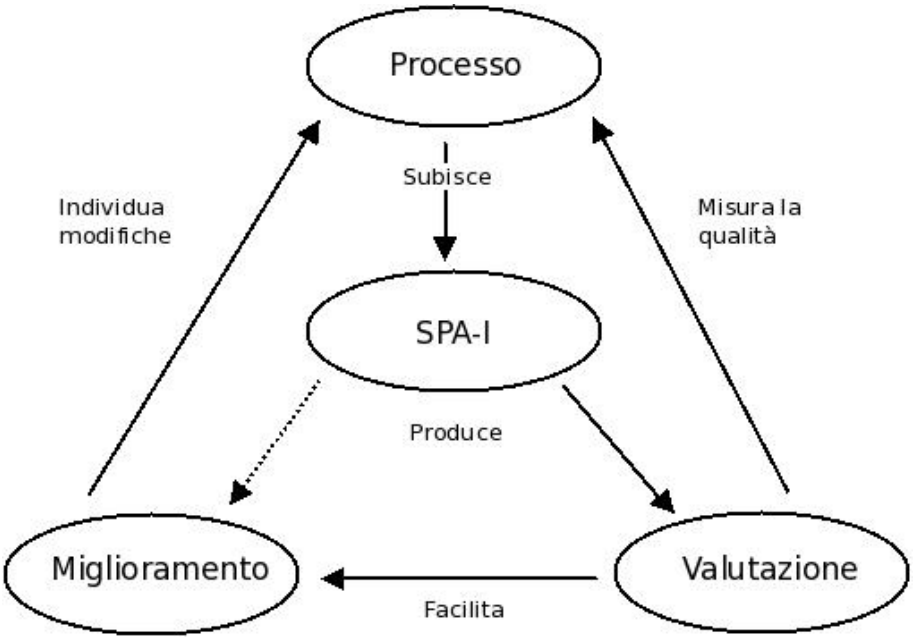


Figure 1: *Schema SPA-I*

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. Per ogni caratteristica generale sarà poi specificato quale sarà il grado minimo di raggiungimento per definire un determinato obiettivo di qualita conseguito. In conclusione, il prodotto si potrà considerare completo e di qualità solo quando tutte le caratteristiche saranno soddisfatte almeno ad un grado minimo.

### 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

**Quantificazione del raggiungimento dell'obbiettivo di qualità:** 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

**Quantificazione del raggiungimento dell'obbiettivo di qualità:** 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

**Quantificazione del raggiungimento dell'obbiettivo di qualità:** 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

**Quantificazione del raggiungimento dell'obbiettivo di qualità:** 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

**Quantificazione del raggiungimento dell'obbiettivo di qualità:** 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

**Quantificazione del raggiungimento dell'obbiettivo di qualità:** 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 - V2.0`.

**Risoluzione di un ticket** Le modalità di risoluzione e chiusura di un ticket sono descritte nelle `Norme di Progetto - V2.0`, 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 Pianificazione ed esecuzione del collaudo

## 5.1 Collaudo

Per il collaudo sono previste due modalità di esecuzione:

- **Alpha - Test**: è un collaudo effettuato internamente e consiste nella verifica finale del traciamento dei requisiti attraverso le varie fasi dello sviluppo. Verrà inoltre effettuata una prova di esecuzione che ha come unico scopo di verificare il corretto funzionamento del prodotto nelle sue varie funzioni.
- **Beta - Test**: collaudo di tipo esterno. I test in questo caso sono affidati a persone esterne allo sviluppo del sistema, che ne testeranno la funzionalità e la fruizione.

## 5.2 Test di Sistema

Tale sezione viene adibita alla pianificazione ed illustrazione dei test di sistema che verranno atuati prima del rilascio del prodotto ultimato.

Lo scopo dell'eseguire tali test è dimostrate che l'applicazione prodotta soddisfa le richieste del cliente, fornendo quindi esempi di test da attuare per confermare che i requisiti individuati nel documento `Analisi Dei Requisiti - V2.0` sono stati soddisfatti.

### 5.2.1 Ambito Generale

| Cod. requisito | Cod. verifica | Modalità verifica |
|---|---|---|
| RFOB 1 | TS-RFOB 1 | Prova dinamica: il server su cui risiede il sistema dovrà dimostrarsi funzionante rispondendo alle richieste dei vari componenti |
| RFOB 1.1 | TS-RFOB 1.1 | Prova dinamica: viene verificato testando la possibilità di connettersi al servizio da differenti dispositivi |
| RFD 2 | TS-RFD 2 | Prova dinamica: viene verificato se, al raggiungimento di determinati obiettivi di Gamification, il sistema assegnerà effettivamente dei badge |
| RFD 2.1 | TS-RFD 2 | Prova dinamica: viene verificato se, al raggiungimento di determinati obiettivi di Gamification, il sistema assegnerà effettivamente dei badge |

| | | |
|---|---|---|
| RFD 2.1.1 | TS-RFD 2 | Prova dinamica: viene verificato se, al raggiungimento di determinati obiettivi di Gamification, il sistema assegnerà effettivamente dei badge |
| RPOB 1 | TS-RPOB 1 | Prova dinamica: viene verificato che, in caso di intenso traffico nella rete, il sistema sia comunque in grado di garantire una corretta gestione dei dati evitandone la perdita |
| RPOB 2 | TS-RPOB 2 | Prova dimanima: viene verificato testando se, con l'utilizzo di un database test di domande risiedente nel medesimo server, il sistema funzioni correttamente ed in egual modo che con l'utilizzo di un database test di domande risiedente in un server remoto |
| RQOP 3.1 | TS-RQOP 3.1 | Prova dinamica: viene verificato se, al passaggio del mouse sopra determinati campi o sezioni del servizio, questo sia in grado di restituire appositi consigli sull'utilizzo dello stesso |
| RQOP 3.2 | TS-RQOP 3.2 | Prova dinamica: viene verificato se, da ogni schermata, il sistema renderà possibile accedere ad un'apposita area di aiuto |
| RVOP 1.1 | TS-RVOP 1.1 | Prova dinamica: viene verificato testando l'applicativo desktop su sistemi operativi di diverso tipo |

### 5.2.2 Ambito Utente

| **Cod. requisito** | **Cod. verifica** | **Modalità verifica** |
|---|---|---|
| RFOB 3 | TS-RFOB 3 | Prova dinamica: viene verifiato se il sistema effettivamente tiene traccia, nel tempo, di tutte le azioni di tutti gli Utilizzatori |
| RFOB 3.1 | TS-RFOB 3.1 | Prova dinamica: viene verifiato se il sistema effettivamente registra in diversi file di log, divisi per macro tipologia di azione, tutte le azioni di ogni Utilizzatore |

| | | |
|---|---|---|
| RFOB 4 | TS-RFOB 4 | Prova dinamica: viene verificato se il sistema offrirà effettivamente la possibilità all'Utente che ha smarrito la password di connettersi all'apposita pagina per richiederne il recupero |
| RFOB 4.1 | TS-RFOB 4.1 | Prova dinamica: viene verificato che, se la procedura di richiesta modifica password è stata eseguita correttamente, il sistema invierà effettivamente una mail contenente una password temporanea generata randomicamente |
| RFOB 4.1.1 | TS-RFOB 4.1.1 | Prova dinamica: viene verificato che al primo login dopo il recupero password, il sistema chieda effettivamente che questa venga modificata |
| RFOB 11 | TS-RFOB 11 | Prova dinamica: viene verificato che il sistema invii una mail di conferma qual'ora i dati personali del dell'Utente siano stati modificati |
| RFOB 11.1 | TS-RFOB 11.1 | Prova dinamica: viene verificato che il sistema invii una mail di conferma qual'ora venisse modificata la password di accesso |

### 5.2.3 Ambito Dipendente

| **Cod. requisito** | **Cod. verifica** | **Modalità verifica** |
|---|---|---|
| RFOB 5 | TS-RFOB 5a | Prova dinamica: viene verificato se a fronte dell'inserimento di un indirizzo email o password non validi il sistema risponda in maniera negativa |
| RFOB 5 | TS-RFOB 5b | Prova dinamica: viene verificato se a fronte dell'inserimento di un indirizzo email e password validi il sistema risponda in maniera positiva |
| RFOB 5.1 | TS-RFOB 5.1 | Prova dinamica: viene verificato se il sistema permetterà effettivamente di effettuare il login tramite l'apposita applicazione web, desktop o android |

| | | |
|---|---|---|
| RFOB 6 | TS-RFOB 6 | Prova dinamica: viene verificato che il sistema sia in grado di fornire, grazie ad un timer, periodiche domande precaricate al Dipendente |
| RFOB 7 | TS-RFOB 7 | Prova dinamica: viene verificato che il sistema sia in grado di fornire delle domande al Dipendente qual'ora le richiedesse |
| RFOB 7.1 | TS-RFOB 7.1 | Prova dinamica: viene verificato che il sistema limiti, grazie ad un timer, il numero di domande che il Dipendente può richiedere |
| RFOB 10 | TS-RFOB 10a | Prova dinamica: viene verificato che il sistema sia in grado di offrire la possibilità al Dipendente di accedere all'area di modifica dei propri dati personali |
| RFOB 10 | TS-RFOB 10b | Prova dinamica: viene verificato che il sistema risponda negativamente alla modifica delle informazioni qual'ora fossero stati inseriti dati non corretti |
| RFOB 10 | TS-RFOB 10c | Prova dinamica: viene verificato che il sistema risponda positivamente alla modifica delle informazioni qual'ora fossero stati inseriti dati corretti |
| RFOB 11.2 | TS-RFOB 11.2 | Prova dinamica: viene verificato che il sistema invii una mail di conferma qual'ora venisse modificato l'indirizzo email |
| RFOB 12 | TS-RFOB 12 | Prova dinamica: viene verificato se un Dipendente autenticato è in grado di accedere ai propri dati personali |
| RFOB 13 | TS-RFOB 13a | Prova dinamica: viene verificato se il sistema risponderà positivamente alla scelta di un Utente di rispondere ad una domanda proposta |
| RFOB 13 | TS-RFOB 13b | Prova dinamica: viene verificato se il sistema risponderà positivamente alla scelta di un Utente di rinviare una domanda proposta |
| RFOB 14.1 | TS-RFOB 14.1 | Prova dinamica: viene verificato se il sistema è in grado di riconoscere ed offrire informazioni sulla tipologia di domanda (nuova, posticipata, sbagliata) |

| | | |
|---|---|---|
| RFOB 15 | TS-RFOB 15 | Prova dinamica: viene verificato se un Dipendente sarà in grado di visualizzare lo storico dei propri punteggi |
| RFOB 16 | TS-RFOB 16 | Prova dinamica: viene verificato se un Dipendente sarà in grado di poter terminare la propria sessione |
| RFOB 19.2.3 | TS-RFOB 19.2.3 | Prova dinamica: viene verificato se un Dipendente potrà modificare la sua password al primo accesso dopo la registrazione |

#### 5.2.4 Ambito Amministratore Azienda

| **Cod. requisito** | **Cod. verifica** | **Modalità verifica** |
|---|---|---|
| RFOB 17 | TS-RFOB 17a | Prova dinamica: viene verificato se il sistema risponde positivamente all'inserimento di username e password validi nella sezione dedicata al login di un Amministratore Azienda |
| RFOB 17 | TS-RFOB 17b | Prova dinamica: viene verificato se il sistema risponde negativamente all'inserimento di username e password non validi nella sezione dedicata al login di un Amministratore Azienda |
| RFOB 18 | TS-RFOB 18 | Prova dinamica: viene verificato se il sistema è in grado di offrire all'Amministratore Azienda Autenticato la possibilità di visionare la pagina di gestione dipendenti |
| RFOB 19 | TS-RFOB 19a | Prova dinamica: viene verificato se il sistema risponderà positivamente all'inserimento, da parte di un Amministratore Azienda Autenticato, di dati corretti relativi ad un nuovo Dipendente |
| RFOB 19 | TS-RFOB 19b | Prova dinamica: viene verificato se il sistema risponderà negativamente all'inserimento, da parte di un Amministratore Azienda Autenticato, di dati non corretti relativi ad un nuovo Dipendente |

| | | |
|---|---|---|
| RFOB 19.2 | TS-RFOB 19.2 | Prova dinamica: viene verificato se il sistema, dopo l'inserimento di un nuovo Dipendente da parte dell'Amministratore Azienda Autenticato, invierà una mail contenente i dati del nuovo account all'indirizzo specificato |
| RFOB 19.2.1 | TS-RFOB 19.2.1 | Prova dinamica: viene verificato se il sistema inserirà nella mail inviata l'username del nuovo accout Dipendente |
| RFOB 19.2.2 | TS-RFOB 19.2.2 | Prova dinamica: viene verificato se il sistema inserirà nella mail inviata la password del nuovo accout Dipendente |
| RFOB 20 | TS-RFOB 20 | Prova dinamica: viene verificato se il sistema sarà in grado di far eliminare all'Amministratore Azienda Autenticato un account Dipendente |
| RFOB 21 | TS-RFOB 21a | Prova dinamica: viene verificato che il sistema sia in grado di permettere all'Amministratore Azienda Autenticato di accedere alla pagina di modifica degli account dei Dipendenti |
| RFOB 21 | TS-RFOB 21b | Prova dinamica: viene verificato che il sistema risponda positivamente alla modifica dei dati di un Dipendente da parte dell'Amministratore Azienda Autenticato con informazioni valide |
| RFOB 21 | TS-RFOB 21c | Prova dinamica: viene verificato che il sistema risponda negativamente alla modifica dei dati di un Dipendente da parte dell'Amministratore Azienda Autenticato con informazioni non valide |
| RFOB 22 | TS-RFOB 22 | Prova dinamica: viene verificato che il sistema permetta ad un Amministratore Azienda Autenticato di visionare le informazioni di ogni Dipendente registrato |
| RFOB 23 | TS-RFOB 23 | Prova dinamica: viene verificato che il sistema permetta da un Amministratore Azienda Autenticato di visionare le informazioni relative ai punteggi di ogni Dipendente registrato |

| | | |
|---|---|---|
| RFOB 24 | TS-RFOB 24 | Prova dinamica: viene verificato che il sistema permetta all'Amministratore Azienda Autenticato di modificare i trofei di ogni singolo Dipendente |
| RFOB 25 | TS-RFOB 25 | Prova dinamica: viene verificato che il sistema permetta all'Amministratore Azienda Autenticato di terminare la propria sessione |

### 5.2.5 Ambito Amministratore Sicurezza

| **Cod. requisito** | **Cod. verifica** | **Modalità verifica** |
|---|---|---|
| RFOB 26 | TS-RFOB 26a | Prova dinamica: viene verificato se il sistema risponde positivamente all'inserimento di username e password validi nella sezione dedicata al login di un Amministratore Sicurezza |
| RFOB 26 | TS-RFOB 26b | Prova dinamica: viene verificato se il sistema risponde negativamente all'inserimento di username e password non validi nella sezione dedicata al login di un Amministratore Sicurezza |
| RFOB 27 | TS-RFOB 27 | Prova dinamica: viene verificato che il sistema offra la possibilità ad un Amministratore Sicurezza Autenticato di consultare le domande dell'azienda |
| RFOB 28 | TS-RFOB 28 | Prova dinamica: viene verificato che il sistema offra la possibilità ad un Amministratore Sicurezza Autenticato di accedere all'interfaccia per la modifica delle domande di un'azienda |
| RFOB 28.1 | TS-RFOB 28.1 | Prova dinamica: viene verificato che il sistema offra la possibilità all'Amministratore Sicurezza Autenticato di aggiungere nuove domande da sottoporre ai dipendenti da un'apposito database generale |

| | | |
|---|---|---|
| RFOB 28.2 | TS-RFOB 28.2 | Prova dinamica: viene verificato che il sistema offra la possibilità all'Amministratore Sicurezza Autenticato di eliminare domande dalla lista di quelle che possono essere somministrate ai Dipendenti |
| RFOP 29 | TS-RFOP 29 | Prova dinamica: viene verificato che il sistema offra la possibilità all'Amministratore Sicurezza Autenticato di poter modificare gli obiettivi di Gamification |
| RFOP 29.1 | TS-RFOP 29.1 | Prova dinamica: viene verificato che il sistema esegua correttamente la modifica dei badge assegnabili ai dipendenti al raggiungimento di obiettivi di Gamification |
| RFOB 30 | TS-RFOB 30 | Prova dinamica: viene verificato che il sistema esegua permetta all'Amministratore Sicurezza Autenticato di terminare correttamente la propria sessione |

#### 5.2.6 Ambito Amministratore Installatore

| **Cod. requisito** | **Cod. verifica** | **Modalità verifica** |
|---|---|---|
| RFOB 31 | TS-RFOB 31 | Prova dinamica: viene verificato che sia possibile eseguire una corretta installazione del sistema in alcuni dispositivi fissi di test |
| RFOB 32 | TS-RFOB 32 | Prova dinamica: viene verificato che sia possibile eseguire una corretta installazione del sistema in alcuni dispositivi mobili di test |
| RFOB 33 | TS-RFOB 33 | Prova dinamica: viene verificato che sia possibile una corretta installazione del servizio web in un server test |
| RFOB 34 | TS-RFOB 34 | Prova dinamica: viene verificato che sia possibile una corretta creazione di un database test per garantire il corretto funzionamento del sistema |

| | | |
|---|---|---|
| RFOP 35 | TS-RFOP 35a | Prova dinamica: viene verificato che il sistema risponda positivamente all'inserimento di una domanda con dati corretti |
| RFOP 35 | TS-RFOP 35b | Prova dinamica: viene verificato che il sistema negativamente all'inserimento di una domanda con dati non corretti |
| RFOP 35 | TS-RFOP 35c | Prova dinamica: viene verificato che il sistema risponda positivamente alla modifica di una domanda con l'inserimento di dati corretti |
| RFOP 35 | TS-RFOP 35d | Prova dinamica: viene verificato che il sistema risponda negativamente alla modifica di una domanda con l'inserimento di dati non corretti |
| RFOP 35 | TS-RFOP 35e | Prova dinamica: viene verificato che il sistema permetta all'Amministratore Installatore di eliminare correttamente una domanda |

## 5.3 Test di integrazione

In questa sezione verrà descritta la pianificazione dei test di verifica e collaudo che verranno eseguiti per le varie parti del sistema. Grazie a questi test si vuole verificare il corretto funzionamento del software, i test sono divisi fra test intra-componenti e inter-componenti.

### 5.3.1 Test intra-componenti

I componenti verranno testati e collaudati seguendo una strategia incrementale attraverso il principio di bottom-up.

Verranno prima collaudati i componenti con minori dipendenze funzionali, per poi proseguire con l'albero delle dipendenze. Nella fattispecie prima sarà necessario avere funzionanti tutti i database e la gestione degli stessi, poi si potrà preseguire con la restante parte back-end e la parte front-end.

Per ogni componente verranno effettuate prove dalle quali ci si può aspettare un esito positivo o un esito negativo.

| **Componente** | **Funzionalità da verificare** |
|---|---|

| | |
|---|---|
| CDESK1 (login Dipendente) | • Login tramite username e password<br>• Recupero password dimenticata |
| CDESK2 (notifica nuova domanda) | • Notifica nuova domanda secondo i tempi impostati<br>• Rifiuto o accettazione nuova domanda<br>• Apertura browser se viene accettata la domanda |
| CWEB1 (login) | • Login tramite username e password<br>• Riconoscimento tipologia Utente<br>• Recupero password dimenticata |
| CWEB2 (gestione dati Utente) | • Visualizzazione dati Utente loggato<br>• Modifica degli stessi |
| CWEB3 (amministrazione utenti) | • Inserimento nuovi utenti<br>• Modifica utenti<br>• Eliminazione utenti |
| CWEB4 (rispondere ad una domanda) | • Visualizzazione della domanda<br>• Gestione risposta, aggiornamento punteggi<br>• Visualizzazione esito risposta |

| | |
|---|---|
| CWEB5 (amministrazione domande) | • Inserimento e rimozione delle domande relative ad un'azienda dall'insieme delle domande globali |
| CWEB6 (visualizzazione punteggi) | • Visualizzazione propri punteggi aggiornati |
| CWEB7 (gestione punteggi) | • Visualizzazione punteggi globali |
| CWEB8 (notifica nuova domanda) | • Notifica nuova domanda<br>• Rifiuto o accettazione nuova domanda |
| CMOB1 (login) | • Login tramite username e password<br>• Recupero password dimenticata |
| CMOB2 (gestione dati Utente) | • Visualizzazione dati Utente loggato<br>• Modifica degli stessi |
| CMOB3 (rispondere nuova domanda) | • Visualizzazione della domanda<br>• Gestione risposta, aggiornamento punteggi<br>• Visualizzazione esito risposta |
| CMOB4 (visualizzazione punteggi) | • Visualizzazione punteggi aggiornati |
| CMOB5 (notifica nuova domanda) | • Visualizzazione nuova domanda |

| | |
|---|---|
| CBACK1 (login) | • Login tramite username e password<br>• Recupero password dimenticata |
| CBACK2 (amministrazione utenti) | • Visualizzazione dati utenti<br>• Modifica degli stessi |
| CBACK3 (gestione domande e risposte) | • Inserimento nuove domande<br>• Modifica domande<br>• Eliminazione domande |
| CBACK4 (amministrazione punteggi) | • Visualizzazione punteggi globali<br>• Inserimento e modifica criteri di attribuzione punteggi e badge |

### 5.3.2 Test inter-componenti

Man mano che il collaudo prosegue verranno aggiunti test e prove sui componenti in maniera incrementale. Ad ogni nuovo collaudo quindi verrà testato il curretto funzionamento inter-componenti tenedono conto delle loro dipendenze. Il sistema software in ogni caso non prevede grosse dipendenze fra i vari componenti front-end e back-end, il grosso del lavoro inter-componenti relativo alle dipendenze andrà rivolto al corretto funzionamento del database e delle relative richieste.

## 5.4 Test di unità

I test di unità sono programmati per quelle classi del progetto che si occupano di soddisfare requisiti funzionali. Sono però escluse dai test le classi dei package che si occupano di gestire la parte grafica del sistema. I test, affinché possano risultare utili e permettano di aumentare la qualità del prodotto, devono soddisfare alcune proprietà:

- **Automatizzazione**: i test devono poter essere eseguiti automaticamente.

- **Approfondimento**: i test devono essere esaustivi ed accurati, in modo da verificare il comportamento di quelle parti del progetto che potrebbero creare errori.
- **Indipendenza**: devono essere il più possibile indipendenti dall'ambiente di esecuzione delle altre classi di progetto.
- **Ininfluenza**: i test non devono influenzare in alcun modo il codice sorgente, quindi non è accettabile che il codice sorgente venga modificato per facilitare o consentire il testing.

Di seguito verranno ora elencati i test più significativi che serviranno a verificare il corretto funzionamento delle singole unità di codice.

| **Codice test** | **Descrizione** |
|---|---|
| TU-CBACK 1.1 | Verificare se, all'inserimento di valori non consentiti all'interno dei campi dati per l'autenticazione, verrà riportato un errore |
| TU-CBACK 1.2 | Verifica se, all'inserimento di valori corretti all'interno dei campi dati per l'autenticazione, si otterrà come risposta uno degli output attesi |
| TU-CBACK 1.3 | Verifica se, all'inserimento di valori non consentiti all'interno dei campi dati per il recupero della password, verrà riportato un errore |
| TU-CBACK 1.4 | Verifica se, all'inserimento di valori corretti all'interno dei campi dati per il recupero della password, si otterrà come risposta uno degli output attesi |
| TU-CBACK 2.1 | Verifica se, all'inserimento di dati non consentiti all'interno dei campi dati per la visualizzazione delle informazioni di un utente, verrà riportato un errore |
| TU-CBACK 2.2 | Verifica se, all'inserimento di dati corretti all'interno dei campi dati per la visualizzazione delle informazioni di un utente, si otterrà come risposta uno degli output attesi |
| TU-CBACK 2.3 | Verificare se, all'inserimento di valori non consentiti all'interno dei campi dati per la creazione di un utente, verrà riportato un errore |
| TU-CBACK 2.4 | Verificare se, all'inserimento di valori corretti all'interno dei campi dati per la creazione di un utente, si otterrà come risposta uno degli output attesi |
| TU-CBACK 2.5 | Verifica che il salvataggio dei dati relativi ad un nuovo utente avvenga correttamente |
| TU-CBACK 2.6 | Verificare se, all'inserimento di valori non consentiti all'interno dei campi dati per la modifica delle informazioni utente, verrà riportato un errore |

| | |
|---|---|
| TU-CBACK 2.7 | Verificare se, all'inserimento di valori corretti all'interno dei campi dati per la modifica delle informazioni utente, si otterrà come risposta uno degli output attesi |
| TU-CBACK 2.8 | Verifica che il salvataggio dei dati modificati di un utente avvenga correttamente |
| TU-CBACK 3.1 | Verifica se, all'inserimento di valori non consentiti all'interno dei campi dati per la creazione delle domande, si otterrà come risposta un errore |
| TU-CBACK 3.2 | Verifica se, all'inserimento di valori corretti all'interno dei campi dati per la creazione delle domande, si otterrà come risposta uno degli output attesi |
| TU-CBACK 3.3 | Verifica che il salvataggio di una nuova domanda avvenga correttamente |
| TU-CBACK 3.4 | Verifica se, all'inserimento di valori non consentiti all'interno dei campi dati per la modifica delle domande, si otterrà come risposta un errore |
| TU-CBACK 3.5 | Verifica se, all'inserimento di valori corretti all'interno dei campi dati per la modifica delle domande, si otterrà come risposta uno degli output attesi |
| TU-CBACK 3.6 | Verifica che il salvataggio dei dati modificati di una domanda avvenga correttamente |
| TU-CBACK 3.7 | Verifica se, all'inserimento di valori non consentiti all'interno dei campi dati per la risposta ad una domanda, si otterrà come risposta un errore |
| TU-CBACK 3.8 | Verifica se, all'inserimento di valori corretti all'interno dei campi dati per la risposta ad una domanda, si otterrà come risposta uno degli output attesi |
| TU-CBACK 3.9 | Verifica che il salvataggio di una risposta ad una domanda avvenga correttamente |
| TU-CBACK 4.1 | Verifica se, all'inserimento di dati non consentiti all'interno dei campi dati per la visualizzazione dei punteggi, verrà riportato un errore |
| TU-CBACK 4.2 | Verifica se, all'inserimento di dati corretti all'interno dei campi dati per la visualizzazione dei punteggi, si otterrà come risposta uno degli output attesi |
| TU-CBACK 4.3 | Verifica se, all'inserimento di dati non consentiti all'interno dei campi dati per la creazione di nuovi trofei, verrà riportato un errore |

| | |
|---|---|
| TU-CBACK 4.4 | Verifica se, all'inserimento di dati corretti all'interno dei campi dati per la creazione di nuovi trofei, si otterrà come risposta uno degli output attesi |
| TU-CBACK 4.5 | Verifica se, all'inserimento di dati non consentiti all'interno dei campi dati per la modifica di trofei, verrà riportato un errore |
| TU-CBACK 4.6 | Verifica se, all'inserimento di dati corretti all'interno dei campi dati per la modifica di trofei, si otterrà come risposta uno degli output attesi |
| TU-CBACK 4.7 | Verifica se si otterrà una risposta corretta ad una richiesta di visualizzazione punteggi. |

## 5.5 Analisi metrica del codice

Si forniscono in tabella una sintesi riguardante le misure statiche del codice effettuate sulle metriche e calcolate utilizzanto il plug-in per Eclipse Metrics.
Si evidenzieranno in rosso quei valori che non rispettano le massime misure consentite.

| **Metrica** | **Media** | **Massimo** |
|---|---|---|
| Complessità ciclomatica | 1,342 | 13 |
| Numero di parametri | 0.903 | 11 |
| Numero di campi dati per classe | 3.34 | 23 |
| Numero di livelli di annidamento | 1,179 | 5 |
| Indice di unità | 6.75 | 32 |
| Indice di dipendenza | 4.875 | 15 |

Nonostante alcuni elementi della colonna massimo superino i valori prefissati è necessario analizzare adeguatamente questi dati prima di trarre conclusioni affrettate. La complessità ciclomatica raggiunge il valore 13 soltanto per la classe GestioneBadgeD in quanto effettua dei controlli molto importanti relativamente alla gestione dei badge di un dipendente. Di conseguenza tale dato può essere considerato tutto sommato accettabile ed infatti, escludendo la complessità ciclomatica di GestioneBadgeD, il valore massimo diviene 8.
Il numero di parametri è a sua volta accettabile anche se sopra la soglia massima consentita in relazione al fatto che questo valore è condizionato dai parametri necessari ai metodi del package condivisi per eseguire il loro compito.
Le altre metriche hanno invece evidenziato valori più che accettabili.

## 5.6 Descrizione dei Test effettuati

Di seguito saranno definiti i test di verifica effettuati sulle componenti del back-end. I test implementati e descritti sono quindi dei test che cercano di verificare singolarmente il comportamento delle funzioni di una componente, ossia quella specifica alla quale il test fa riferimento.

### 5.6.1 Test Package com.safetyGame.back.access

**SqlDAODipendentiTest**

**Classe verificata**: SqlDAODipendenti
**Descrizione**: Questa classe controlla se la classe in test è in grado di interagire correttamente con il database SQL. Nel dettaglio viene controllato se è possibile aggiungere un nuovo dipendente al DB aziendale, visualizzare le informazioni di un dipendente, modificare le informazioni di un dipendente, modificare i ¨ di un dipendente, eliminare un dipendente, ottenere l'elenco dei dipendenti, ottenere le informazioni di un amministratore e modificare le informazioni di un amministratore. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

**SqlDAODomandeTest**

**Classe verificata**: SqlDAODomande
**Descrizione**: Questa classe controlla se la classe in test è in grado di interagire correttamente con il database SQL. Nel dettaglio viene controllato se è possibile aggiungere una nuova domanda, eliminare una domanda, ottenere una domanda specifica, ottenere i campi di una domanda specifica, ottenere la lista di domande di un'azienda, segnalare una domanda come risposta e segnalare una domanda come posticipata. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

**SqlDAOLoginTest**

**Classe verificata**: SqlDAOLogin
**Descrizione**: Questa classe controlla se la classe in test è in grado di interagire correttamente con il database SQL. Nel dettaglio viene controllato se è possibile effettuare il login di un dipendente o di un amministratore. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

**SqlDAOPunteggiTest**

**Classe verifica:** SqlDAOPunteggi
**Descrizione:** Questa classe controlla se la classe in test è in grado di interagire correttamente con il database SQL. Nel dettaglio viene controllato se è possibile

ottenere le statistiche dei punteggi di uno specifico utente e se è possibile ottenere le statistiche globali degli utenti. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito:** nessun errore riscontrato.

**SqlDAOBadgeTest**

**Classe verifica**: SqlDAOBadge
**Descrizione**: Questa classe controlla se la classe in test è in grado di interagire correttamente con il database SQL. Nel dettaglio viene controllato se è possibile ottenere i badge dell'azienda, i badge personali di un dipendente o se è possibile assegnare un badge ad un dipendente. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

### 5.6.2 Test package com.safetyGame.back.condivisi

**DataOraTest**

**Classe verificata**: DataOra
**Descrizione**: Questa classe controlla se la classe in test è in grado di creare in maniera corretta un oggetto di tipo DataOra. Nel dettaglio viene controllato se è possibile ottenere un oggetto DataOra corretto, a seconda del diverso costruttore richiamato.
**Esito**: nessun errore riscontrato.

### 5.6.3 Test package com.safetyGame.back.controller

**GestioneLogTest**

**Classe verifica**: GestioneLog
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare correttamente le operazioni di registrazione dei log, secondo quanto definito in fase di analisi dei requisiti. Nel dettaglio viene controllato se è possibile creare un nuovo file di log qual'ora non esistesse, oppure aggiornarne uno già esistente al verificarsi di determinate operazioni quali il login di un utente od un amministratore, il logout, la ricezione di domande, la proposta di domande, il posticipo di una domanda, la risposta ad una domanda, la modifica delle informazioni di un utente, l'ottenimento di un badge da parte di un utente, l'aggiunta di un nuovo dipendente, la rimozione di un dipendente, l'aggiunta di una nuova domanda e la rimozione di una domanda. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: i test sulla classe GestioneLog hanno evidenziato che, a causa di alcune restrizioni del server TomCat, non è possibile per ora attivare le funzioni di log delle operazioni sopra elencate.

**GestioneBadgeASTest**

**Classe verifica:** GestioneBadgeAS
**Descrizione:** Classe che controlla se la classe in test è in grado di ottenere correttamente la lista di tutti i badge disponibili presso una data azienda. Questa operazione è stata effettuate definendo un input fittizio e verificandone l'output.
**Esito:** nessun errore riscontrato.

**GestioneRecuperoTest**

**Classe verifica**: GestioneRecupero
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare il reset della password di un dipendente. Nel dettaglio si andrà a testare che la classe sia in grado di resettare la password di un dipendente, di resettare la password di un amministratore e di inviare infine un'apposita mail contenente la nuova password da utilizzare per l'accesso al sistema. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

**GestioneDipendentiAATest**

**Classe verifica**: GestioneDipendentiAA
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare le operazioni di gestione dei dipendenti di un'azienda. Nel dettaglio si andrà a testare che la classe possa aggiungere un nuovo dipendente, rimuovere un dipendente, modificare le informazioni di un dipendente, modificare la password di un dipendente e visionare l'elenco dei dipendenti di un'azienda. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

**GestioneBadgeDTest**

**Classe verifica**: GestioneBadgeD
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare le operazioni di recupero dei badge personali di un dipendente. Nel dettaglio si andrà a testare che la classe esegua correttamente l'ottenimento di tutti i badge assegnati ad uno specifico utente e di assegnare un certo badge ad un utente. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

**GestioneDipendentiDTest**

**Classe verificata**: GestioneDipendentiD
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare le operazioni di interazione con il sistema Safety-Game. Nel dettaglio si andrà

a testare che la classe riesca ad effettuare correttamente il recupero delle informazioni di un dipendente e la modifica delle informazioni di un dipendente. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

#### GestioneDomandeASTest

**Classe verificata**: GestioneDomandeAS
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare le operazioni di interazione con il database delle domande da parte di un amministratore sicurezza. Nel dettaglio si andrà a testare che la classe riesca ad effettuare correttamente il recupero dell'elenco generale delle domande, l'aggiunta di una nuova domanda a quelle specifiche di un'azienda e la rimozione di una domanda. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato

#### GestioneLoginTest

**Classe verificata**: GestioneLogin
**Descrizione**: CLasse che controlla se la classe in test è in grado di effettuare le operazioni di login di un un utente all'interno del sistema. Nel dettaglio si andrà a verificare che utenti di tipo dipendente ed amministratore riescano ad effettuare il login ed il logout. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato

#### GestionePunteggiAATest

**Classe verificata**: GestionePunteggiAA
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare le operazioni riguardanti il recupero dei punteggi medi dell'azienda ed i punteggi di tutti i dipendenti oltre che alla modifica dei trofei di uno specifico utente. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

#### GestionePunteggiDTest

**Classe verificata**: GestionePunteggiD
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare le operazioni riguardanti la gestione dei punteggi da parte di un dipendente. Nel dettaglio si verificherà il funzionamento delle funzioni di ottenimento delle statistiche di uno specifico dipendente e le statistiche globali. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

**GestioneDatiTest**

**Classe verificata**: GestioneDati
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare le operazioni di gestione dei dati contenuti all'interno del sistema Safety-Game. La classe GestioneDati è un façade e si è quindi verificato che le sue sottoclassi svolgessero correttamente le operazioni ad esse assegnate. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

**GestioneDomandeDTest**

**Classe verifica**: GestioneDomande
**Descrizione**: Classe che controlla se la classe in test è in grado di effettuare le operazioni di gestione delle interazioni tra un dipendente ed una domanda del sistema. Nel dettaglio si andranno a verificare le funzioni di ottenimento di una domanda, di risposta ad una domanda e di posticipo di una domanda. Queste operazioni sono state effettuate definendo un input fittizio e verificandone l'output.
**Esito**: nessun errore riscontrato.

# 6 Dettaglio dell'esito delle revisioni

Ad ogni revisione che il grupo deve sostenere, dovrà essere presentata al committente l'apposita documentazione prodotta corredata da una presentazione orale di quanto realizzato allo scopo di valutare lo stato di avanzamento dei lavori. Il committente a questo punto produrrà una valutazione dettagliata del lavoro svolto, documento per documento. Gli eventuali errori ed inesattezze che sorgeranno da tale valutazione, così come le migliorie proposte, dovranno produrre una correzione e modifica dei documenti in modo tale da evitare il perpetuarsi di inesattezze nelle revisioni successive.

## 6.1 Revisione dei Requisiti

Il lavoro svolto per la fase di Revisione dei Requisiti è stato valutato positivamente, dimostrando che il materiale prodotto era sufficientemente adeguato. Sono state comunque segnalati alcuni errori e problematiche principalmente nei documenti di Analisi dei Requisiti, Piano di Progetto, Piano di Qualifica e Norme di Progetto. Di seguito verranno quindi riportate nel dettaglio le operazioni di correzione:

- **Analisi dei requisiti:** Sono stati completamente rivisti e modificati i diagrammi dei casi d'uso e le relative descrizioni, eliminando gli errori e le imprecisioni riscontrate. Sono stati poi di conseguenza eliminati, modificati o aggiunti alcuni requisiti ed è stato rieffettuato infine il tracciamento Casi d'uso - Requisiti.
- **Piano di Progetto:** Ridefinito l'inizio delle attività di verifica, ora correttamente distribuite lungo tutto l'arco del progetto. Unificate in un'unica sezione la Pianificazione ed il Preventivo economico.
- **Piano di Qualifica:** Approfondite le operazioni di inspection nei documenti e nel codice. Migliorata la descrizione dell'organizzazione delle strategie di verifica ed aggiunto uno specifico capitolo inerente alle strategie di verifica vere e proprie. Migliorata la descrizione della fase di verifica svolta nella fase di Revisione dei Requisiti.
- **Norme di Progetto:** Corretta la notazione della data. Dettagliato maggiormente il documento per quanto riguarda le procedure e gli strumenti utilizzati.

## 6.2 Revisione di Progettazione

Il lavoro svolto per la fase di Revisione di Progettazione è stato valutato positivamente, anche se alcuni elementi legati all'utilizzo e implementazione dei design pattern è stato valutato non sufficientemente adeguato. A tal proposito è stato infatti necessario rivedere in gran parte il documento di Specifica Tecnica. Sono stati poi modificati e rivisti anche i documenti di Norme di Progetto, Analisi der

Requisiti, Piano di Progetto e Piano di Qualifica. Di seguito verranno quindi riportate nel dettaglio le operazioni di correzione:

- **Norme di Progetto:** Corrette alcune inesattezze lessicali relative ad abbreviazioni utilizzate nel documento e ai termini con cui si sono indicati i diagrammi UML.
- **Analisi dei requisiti:** Corrette alcune inessattezze lessicali. Modificati alcuni diagrammi dei casi d'uso e relative descrizioni, eliminando gli errori e dettagliando maggiormente dove necessario. Sono stati poi di conseguenza eliminati, modificati o aggiunti alcuni requisiti ovvero alcuni di questi, considerati troppo ad alto livello, sono stati maggiormente dettagliati e resi più specifici.
- **Specifica Tecnica:** Sono stati rivisti i design pattern architetturali impiegati per definire la struttura dell'applicazione SafetyGame, in particolare si scelto di comune accordo di adattare la struttura ideata ad un design pattern di tipo Model View Presenter (MVP). Sono stati aggiunti poi nuovi design pattern creazionali (Singleton, Factory Method) e strutturali (Façade, Data Access Object) ritenuti utili al fine di migliorare il progetto. Riscritti integralmente i diagrammi dei package per adattarli ai nuovi design pattern. Rifatto il tracciamento Componenti - Requisiti e Requisiti - Componenti.
- **Piano di Progetto:** Corrette le indicazioni relative alle fasi di Verifica del progetto, completata l'analisi dello scostamento tra preventivo e consuntivo ed aggiornata infine l'analisi dei rischi alla luce del periodo trascorso.
- **Piano di Qualifica:** Specificati in maniera più chiara la quantificazione degli obbiettivi di qualita per il loro raggiungimento.

# 7 Resoconto attività di verifica

## 7.1 Revisione dei Requisiti

### 7.1.1 Verifica della documentazione

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 tramite walkthrough prima ed inspection poi, verificando che i documenti rispettino i punti individuati e riportati in tale sezione inerenti la formattazione del testo, effettuando le dovute procedure per la correzione degli errori rilevati. Nel dettaglio, i documenti sono stati revisionato effettuando 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.
Infine, per effettuare il doppio tracciamento, si sono controllate le apposite tabelle come descritto nel capitolo 2.1.

## 7.2 Revisione di Progettazione

### 7.2.1 Verifica della documentazione

Anche in tale fase è stata effettuata un'accurata analisi statica sui documenti in maniera simile a quanto avvenuto nella fase di Revisione dei Requisiti. Una particolare attenzione è stata però posta nella fase di verifica dei diagrammi UML in quanto la gran parte degli errori e delle imprecisioni rilevate nei documenti presentati durante la precedente revisione erano concentrati in tale sezione. Nei documenti di Specifica Tecnica e Piano di Qualifica è poi stato fondamentale effetuare un'adeguata ed attenta fase di tracciamento. Come in precedenza sono stati infine utilizzati i Ticket per segnalare eventuali irregolarità ed errori che sono stati poi presi in carico dal redattore e risolti.

## 7.3 Revisione di Qualifica

### 7.3.1 Verifica della documentazione

A causa degli esiti conseguiti nella Revisione di Progettazione, è stata posta particolare attenzione alla ristesura del documento di Specifica Tecnica. Si è innanzitutto valutata l'architettura prodotta ed i design pattern utilizzati, dopodiché è stata effettuata un'ulteriore fase di studio ed analisi di altri design pattern in modo da verificare se fosse necessario implementarne di nuovi. Tale fase ha in conclusione comportato una notevole riprogettazione dell'architettura generale del prodotto e dei componenti di cui è composto. E' quindi stata posta particolare attenzione all'implementazione più corretta ed efficiente possibile dei vecchi e nuovi design pattern per evitare incongruenze strutturali.
Nella stesura del documento di Definizione di Prodotto è stata effettuata una

verifica sulla correttezza grammaticale e la chiarezza dell'esposizione. Inoltre è stata verificata la presenza di apposite descrizioni per tutte le classi e metodi appartenenti al progetto ed è stato anche valutato il loro livello di chiarezza e descrizione, in modo che risultassero sufficientemente esaustive. Anche in questa fase è stata logicamente svolta un'accurata analisi statica dei documenti e dei diagrammi realizzati tramite accurate operazioni di inspection. Esse sono state svolte tenendo conto degli errori più spesso commessi e delle aree maggiormente a rischio.

### 7.3.2 Verifica del codice

I documenti non sono però stati i soldi ad essere sottoposti ad un'accurata fase di verifica: anche il codice prodotto infatti è stato adeguatamente verificato grazie agli strumenti a nostra disposizione. E' stata svolta un'accurata inspection su tutto il codice prodotto per la Revisione di Qualifica, andando a verificare che fossero stati adottati tutti i punti guida individuati nel paragrafo "2.5.2 Tecniche - Inspection del codice". Tali punti guida sono stati fondammentali per poter mantenere innanzitutto un'uniformità di stesura del codice per tutte le varie componenti di cui è composto e poi, cosa ben più importante, evitare il verificarsi di errori più o meno gravi. Nella fase di inspection è stato infatti rilevato più di qualche errore dato dalla non osservanza di questi punti che è però stato prontamente segnalato e corretto. Il resto delle attività di verifica e test sul codice sono state riportate all'interno del paragrafo 5 "Pianificazione ed esecuzione del collaudo" di questo stesso documento.