MyTalk

Sofware di comunicazione tra utenti senza requisiti di installazione

clockworkTeam7@gmail.com

Piano di Qualifica

v 1.0

Informazioni sul documento

Nome documento	Piano di Qualifica
Versione documento	v 1.0
Data creazione	11/12/2012
Data ultima modifica	18/12/2012
Uso documento	Esterno
Redazione	Palmisano Maria Antonietta
Verifica	Furlan Valentino
Approvazione	Zohouri Haghian Pardis
Lista distribuzione	gruppo *Clockwork* Zucchetti SPA Prof. Tullio Vardanega

Sommario

Questo documento si prefigge di regolamentare le operazioni di verifica e validazione del gruppo *Clockwork* nello svolgimento del progetto **MyTalk**.

Diario delle modifiche

Autore	Modifica	Data	Versione
Zohouri Haghian Pardis	approvazione del documento	18/12/2012	v 1.0
Furlan Valentino	Verifica e correzioni grammaticali e sintattiche	18/12/2012	v 0.11
Palmisano Maria Antonietta	Stesura Resoconto attività di verifica	17/12/2012	v 0.10
Palmisano Maria Antonietta	Stesura sezione Pocedure di controllo di qualità di processo	17/12/2012	v 0.9
Palmisano Maria Antonietta	Stesura sezione Comunicazione e risoluzione di anomalie, stesura sezione Trattamento delle discrepanze	16/12/2012	v 0.8
Palmisano Maria Antonietta	Completata stesura sezione Strumenti e Tecniche, stesura sezione Misure e Metriche	16/12/2012	v 0.7
Palmisano Maria Antonietta	Stesura preliminare sezione Strumenti e Tecniche	14/12/2012	v 0.6
Palmisano Maria Antonietta	Stesura sezione Qualità	13/12/2012	v 0.5
Palmisano Maria Antonietta	Completata stesura sezione Risorse necessarie e Risorse disponibili	13/12/2012	v 0.4
Palmisano Maria Antonietta	Stesura sezione Responsabilità, stesura preliminare sezione Risorse necessarie e Risorse disponibili	12/12/2012	v 0.3
Palmisano Maria Antonietta	Stesura sezione Organizzazione, stesura sezione pianificazione strategica generale	11/12/2012	v 0.2
Palmisano Maria Antonietta	Creazione documento, stesura sezione Introduzione	11/12/2012	v 0.1

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento si prefigge di illustrare la strategia complessiva di verifica e validazione del gruppo *Clockwork* per la garanzia della qualità del progetto **MyTalk** e dei processi attuati per la sua produzione. Al fine di evidenziare e correggere anomalie, difetti e incongruenze, verranno presentati i vari criteri di qualità adottati, le metriche correlate e le conseguenti soglie che permetteranno di discriminarne l'effettiva qualità.

1.2 Scopo del prodotto

Il prodotto denominato **MyTalk** si propone di fornire un software per un sistema di comunicazione audio e video tra utenti. Lo scopo del progetto è poter comunicare con altri utenti tramite il browser, utilizzando solo componenti standard, senza dover installare plugin o programmi esterni. L'utilizzatore dovrà poter chiamare un altro utente, iniziare la comunicazione sia audio che video, svolgere la chiamata e terminare la chiamata ottenendo delle statistiche sull'attività.

1.3 Glossario

I termini tecnici o di uso non comune sono presenti nel documento allegato `Glossario_v1.0.pdf`. Tali riferimenti vengono evidenziati tramite sottolineatura alla prima occorrenza del termine nel documento.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato d'Appalto: **MyTalk**, v 1.1, rilasciato dal proponente Zucchetti SPA, reperibile all'indirizzo: `http://www.math.unipd.it/~tullio/IS-1/2012/Progetto/C1.pdf`
- Analisi dei Requisiti (allegato `Analisi_dei_Requisiti _v1.0.pdf`)
- Norme di Progetto (allegato `Norme_di_Progetto_v1.0.pdf`)
- Verbale incontro proponente 18/12/2012 (allegato `Verbale20121218.pdf`)

1.4.2 Informativi

- Software Engineering - Part 5: Verification and Validation, Part 6: Management - Ian Sommerville - 8th ed. (2006)
- SWEBOK - Chapter 11: Software Quality: `http://www.computer.org/portal/web/swebok/html/ch11`
- ISO/IEC 12207 - Standard per l'organizzazione dei processi

- ISO/IEC 9126:2001 - Standard per la valutazione della qualità del software

2 Visione generale delle strategie di verifica

2.1 Organizzazione

Il processo di verifica verrà istanziato quando il prodotto di un processo raggiungerà uno stato ritenuto sostanzialmente diverso da quello precedente: grazie al registro delle modifiche sarà possibile operare una verifica circoscritta ai soli cambiamenti. Se, eventualmente, saranno riscontrati problemi o anomalie nel corretto funzionamento di un prodotto, essi verranno trattati dal processo di risoluzione dei problemi.
Il gruppo *Clockwork* ha deciso di adottare per lo sviluppo del progetto un ciclo di vita incrementale (vedi Piano di Progetto). In base a tale scelta, il processo di verifica che verrà adottato opererà durante le diverse fasi del progetto nel seguente modo:

- **Analisi dei requisiti:** ogni documento necessario alla revisione dei requisiti (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 i plugin Hunspell ed Enchant mentre il controllo 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 4.1 dove si possono visionare quelli adottati dal gruppo *Clockwork* 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:** alla fase di collaudo (RA), il gruppo *Clockwork* garantisce il corretto funzionamento del prodotto realizzato. Se verranno riscontrati difetti o caratteristiche non attinenti alle richieste del cliente, ogni modifica che verrà effettuata per eliminare tali incongruenze rispetto al prodotto atteso sarà a carico del fornitore

2.2 Pianificazione strategica generale

Una verifica continua sui processi e sui prodotti, attuata in modo preventivo e non retrospettivo, può garantire un alto livello di qualità al fine di rispettare le aspettative del cliente e il pieno soddisfacimento dei requisiti richiesti dal Capitolato d'Appalto, minimizzando i rischi di fallimento prematuro del progetto e riducendo i costi di correzione degli eventuali errori.
Il Responsabile di Progetto si impegna a definire le attività di verifica finalizzate al collaudo di sistema, e le relative scadenze, assegnando tali attività ai verificatori in modo che ciascuno dei singoli elementi software, documentali o di altro genere, vengano verificati durante i processi realizzativi ed entro le scadenze previste.

I processi di revisione adottati saranno di due tipi:

- Revisioni formali condotte dal cliente e con effetto sanzionatorio (corrispondenti all'Audit Process di ISO/IEC 12207)
  - Revisione dei Requisiti (RR): 09/01/2013
  - Revisione di Accettazione (RA): da destinarsi, ipotizzata al 21/03/2013
- Revisioni informali di revisione interna (di progresso) con il coinvolgimento del cliente (corrispondenti al Joint Review Process di ISO/IEC 12207)
  - Revisione di Progettazione (RP): 06/02/2013
  - Revisione di Qualifica (RQ): 05/03/2013

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 necessarie

La verifica di qualità del prodotto e dei processi impone l'utilizzo di risorse umane e tecnologiche.

2.4.1 Risorse umane

I ruoli necessari alla garanzia di qualità sono i seguenti:

- **Responsabile di Progetto:** controlla la qualità dei processi interni, assegnando le attività di verifica ai ruoli preposti, ed è responsabile della corretta realizzazione nel prodotto nei confronti del committente. Si preoccupa inoltre di approvare o respingere ogni proposta di azione correttiva o migliorativa fornita dai verificatori decidendone anche l'assegnazione (vedi sezione 5)
- **Amministratore di Progetto:** definisce le metodologie e le norme delle attività di verifica, compresa la distribuzione dei resoconti relativi ai test eseguiti, e la gestione e risoluzione di anomalie e discrepanze (vedi sezione 5.1)
- **Programmatore:** a sua discrezione può applicare procedure di debugging per verificare il codice che ha prodotto; inoltre si impegna a risolvere tutte le anomalie evidenziate dai verificatori e dai test effettuati sul codice da lui prodotto
- **Verificatore:** applica processi di verifica e validazione su ogni prodotto e su ogni processo, e tiene traccia dei risultati ottenuti[1]. Tali attività riassumeranno gli esiti delle analisi delle misurazioni, individuando eventuali problematiche che verranno presentate al Responsabile di Progetto per essere risolte (vedi sezione 5)

2.4.2 Risorse software

- Framework utilizzati per i test di unità inerenti al linguaggio di programmazione scelto
- Strumenti per automatizzare i test e produrre resoconti sulle eventuali anomalie riscontrate
- Strumenti aggiuntivi per permettere una semplice e automatizzata analisi statica del codice in modo da poterne ricavare il maggior numero di informazioni possibile
- Software utile alla comunicazione fra i membri del gruppo

2.4.3 Risorse hardware

- Computer con ambiente di lavoro adatto allo sviluppo del progetto e per la fase di testing dello stesso
- Luogo di lavoro dove potersi incontrare e lavorare insieme quando necessario

[1] Il tracciamento verrà effettuato esclusivamente all'interno dei processi che lo richiedono (es: la verifica della documentazione non richiede tracciamento).

2.5 Risorse disponibili

2.5.1 Risorse software

- Github e correlati: risorse messe a disposizione dal servizio web Github per gestire commenti o problemi a codice, documentazione e ticket
- Hunspell ed Enchant per il controllo ortografico dei documenti
- Eclipse con relativi strumenti già inclusi o aggiunti tramite plugin
- Firebug Lite per l'analisi dinamica
- Yasca per l'analisi statica del codice HTML
- Chromedriver e Firebug Lite plugin per Google Chrome per l'analisi dinamica
- JUnit per i test di unità
- Strumenti W3C per la validazione dell'applicativo web
- Gruppo Facebook per poter comunicare fra membri del gruppo
- Speed Tracer per il controllo dell'efficienza dell'applicativo web

2.5.2 Risorse hardware

- Computer personali, portatili e fissi, di ogni membro del gruppo
- Aule studio del dipartimento di Matematica Pura ed Applicata dell'Università degli Studi di Padova
- Computer dei laboratori di informatica del dipartimento di Matematica Pura ed Applicata dell'Università degli Studi di Padova

3 Qualità

Prendendo come riferimento lo standard ISO/IEC 9126, il gruppo *Clockwork* si impegna a garantire le seguenti qualità per il prodotto **MyTalk**

- **Funzionalità:** il software deve soddisfare i compiti prestabiliti, poter interagire con altri sistemi, aderire a standard e convenzioni nell'ambiente in cui viene applicato, gestire i dati in modo sicuro, negando l'accesso a entità non autorizzate
- **Affidabilità:** il software deve poter evitare che si verifichino malfunzionamenti, poter gestire situazioni non attese o usi scorretti, poter essere riutilizzabile nel caso in cui avvengano errori critici, sia affidabile in conformità all'ambiente di applicazione
- **Efficienza:** l'esecuzione non deve allocare risorse in modo spropositato e i suoi tempi di risposta devono essere adeguati alle aspettative dell'utente
- **Usabilità:** l'esecuzione delle funzioni offerte dal software devono poter essere intuibili anche all'utente non esperto, o non difficilmente apprese
- **Manutenibilità:** la modifica del codice del software deve poter essere quanto più facile possibile, grazie a un codice comprensibile
- **Portabilità:** il software deve poter funzionare in ambienti di lavoro diversi

3.1 Altre qualità

Avranno inoltre un peso notevole nell'ideazione e realizzazione di **MyTalk** i seguenti punti:

- **Semplicità:** realizzazione del prodotto nella maniera più semplice possibile, ma non semplicistica
- **Incapsulamento:** il codice deve avere visibilità minima e permettere un utilizzo dall'esterno solamente mediante interfacce; ciò aumenta manutenibilità e possibilità di riuso del codice
- **Coesione:** le funzionalità che concorrono allo stesso fine devono risiedere nello stesso componente; favorisce semplicità, manutenibilità, riusabilità e riduce l'indice di dipendenza (vedi sezione 4.3)

4 Strumenti, Tecniche e Metodi

4.1 Strumenti

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

- **Hunspell ($\geqslant$ 1.3.2-4build1):** strumento per la correzione grammaticale dei documenti redatti. L'utilizzo di Hunspell avverrà tramite apposito plugin per LyX (`http://hunspell.sourceforge.net/`)
- **Enchant ($\geqslant$ 1.6.0-7build1):** strumento per la correzione grammaticale dei documenti redatti. L'utilizzo di Enchant avverrà tramite apposito plugin per LyX (`http://abisource.com/projects/enchant/`)
- **Yasca ($\geqslant$ 2.1):** programma open source che permette l'analisi statica del codice HTML `http://www.scovetta.com/yasca.html`
- **Chromedriver ($\geqslant$ 23.0.1240.0):** plugin per Google Chrome esegue test direttamente sul browser (`http://code.google.com/p/chromedriver`)
- **Firebug Lite ($\geqslant$ 1.4):** plugin per Google Chrome per l'analisi dinamica `http://getfirebug.com/releases/lite/chrome`
- **Eclipse ($\geqslant$ 3.8.0):** IDE multi-linguaggio e multi-piattaforma che fornisce, nella versione base, alcune funzionalità utili di debugging, come l'esecuzione del codice step-by-step, l'impostazione di breakpoint, visualizzazione dei valori di variabili e strutture dati durante l'esecuzione, sospensione e riavvio di thread in esecuzione, ecc. (`http://www.eclipse.org/`)
- **FindBugs ($\geqslant$ 2.0.0):** plugin per Eclipse, in grado di individuare ed evidenziare staticamente, nel codice sorgente, alcuni tra i più comuni errori nella scrittura di codice Java (come l'uso di worst practice). Ogni errore riscontrato verrà analizzato e discusso per valutarne l'eventuale modifica correttiva (`http://findbugs.sourceforge.net/`)
- **EclEmma ($\geqslant$ 2.2.0):** plugin per Eclipse, in grado di verificare automaticamente la copertura del codice, sia durante la normale esecuzione, sia durante l'analisi dinamica tramite test. I test devono coprire il codice sorgente per intero (`http://www.eclemma.org/`)
- **Metrics Plugin for Eclipse ($\geqslant$ 1.3.6):** plugin per Eclipse che fornisce informazioni riguardanti le misure statiche del codice (vedi sezione 4.3 per chiarimenti) come:
  - 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/`)

- Strumenti di validazione W3C:
  - **Markup Validation Service:** per effettuare test su tutte le pagine navigabili del sistema e verificare l'aderenza allo standard HTML5 (`http://validator.w3.org/`)
  - **CSS Validation Service:** per effettuare test sui fogli di stile di tutte le pagine del sistema e verificare l'aderenza allo standard CSS versione 3 (`http://jigsaw.w3.org/css-validator/`)
- **Speed Tracer (⩾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/`)
- **JUnit (⩾4.11):** framework per effettuare test di unità per il linguaggio di programmazione Java (`http://www.junit.org/`)

Per tutti gli altri strumenti utilizzati, non strettamente legati alla verifica, vedere le Norme di Progetto.

4.2 Tecniche

4.2.1 Analisi statica

Verrà applicata durante l'intero ciclo di vita del software, in due tipologie diverse:

- **Walkthrough:** lettura critica del codice e/o dei documenti senza l'assunzione di presupposti e a largo spettro. Ogni difetto riscontrato viene discusso tra i verificatori e gli sviluppatori[2] affinché siano evitate incomprensioni, per poi procedere alla modifica concordata. Ogni attività svolta deve essere registrata. Sebbene richieda risorse considerevoli, il walkthrough sarà indispensabile per individuare tutti gli errori e le anomalie, ed ogni problematica individuata verrà aggiunta alla lista di controllo, al fine di ridurre ulteriormente l'utilizzo di walkthrough nelle fasi successive, affinando la fase di inspection. Per la fase di redazione dei documenti si procederà iniziando con il walkthrough, per poi passare successivamente all'inspection, grazie all'aggiunta di nuovi punti alla lista di controllo, come quelli presentati poco sotto. Per la fase di codifica si inizierà con il walkthrough per poi passare, in una fase intermedia, all'inspection, grazie all'arricchimento della lista di controllo

[2] Identificati in tutti coloro che hanno realizzato il determinato prodotto sottoposto a verifica.

- **Inspection:** lettura mirata del codice e/o dei documenti, guidata da una lista di controllo definita gradualmente, sia grazie all'esperienza personale, sia attraverso l'arricchimento derivante dall'attività di walkthrough, tale da raccogliere le tipologie di errori più frequenti che possono essere commessi durante un determinato processo e sono stati quindi individuati come critici. Per quanto riguarda l'inspection dei documenti, la lista di controllo conterrà i seguenti punti:
  - LaTeX:
    * Impostazione generale del documento affinché 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
  - 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

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

Ogni difetto riscontrato viene corretto e ogni fase dell'inspection viene documentata con dei rapporti sulle attività svolte.
L'inspection verrà preferita al walkthrough in quanto richiede meno attenzione ed è meno collaborativa (permettendo di essere effettuata in meno tempo e con meno risorse). All'inizio della codificazione non potremo affidarci ad una lista di controllo completa e dettagliata, e quindi si rimanderà l'inspection ad una fase intermedia nella quale la lista di controllo sarà sufficientemente ricca da poter essere affidabile e in grado di fornire vantaggi superiori al semplice walkthrough.

Metodi di Analisi Statica:

Per quanto riguarda questa fase, le metodologie di analisi statica messe a disposizione sono svariate, sarà opportuno attuare l'analisi più adatta in base ad ogni situazione che si deve affrontare.

- **Analisi del flusso di controllo:** verificherà, analizzando i vari flussi possibili, che il codice proceda correttamente e quindi rispetti la sequenza specificata. Accerteremo che il codice sia ben strutturato, che non esistano parti di codice che non vengono mai raggiunte o che non terminano. Il tutto tramite la tecnica di call-tree analysis (albero delle chiamate), sfruttando la capacità dei compilatori di calcolarlo, e verificando che le variabili di controllo delle iterazioni non vengano modificate. Il flusso di controllo sarà opportunamente analizzato in quelle parti di codice in cui, per esempio, vengono effettuate diverse chiamate di metodi a catena o vi siano variabili di controllo delle iterazioni

- **Analisi di flusso dei dati:** assicura che il flusso di dati non faccia uso di variabili non ancora inizializzate o prive di valore in modo da evitare possibili errori al momento del loro utilizzo. Tali errori verranno facilmente evitati con una buona costruzione del codice da parte del programmatore, prestando particolare attenzione ad inizializzare quelle variabili alle quali non viene assegnato un valore in automatico dal compilatore. I verificatori avranno comunque il compito di effettuare tale analisi
- **Flusso d'informazione:** individuazione delle relazioni tra ingressi ed uscite tra i vari moduli, accertando che le uniche dipendenze consentite siano quelle previste dalla specifica dei moduli. Questo tipo di analisi può essere effettuata a tre livelli: singolo modulo, più moduli tra loro correlati, intero sistema. In base ai moduli che otterremo si deciderà quali livelli di analisi utilizzare
- **Verifica formale del codice:** tramite l'esplorazione di tutte le esecuzioni possibili si verifica la correttezza del codice rispetto alla specifica dei requisiti

4.2.2 Analisi dinamica

Verrà effettuata verifica e validazione di ogni prodotto dell'attività di Realizzazione attraverso l'esecuzione di test. Ogni singolo test deve essere ripetibile, cioè deve essere effettuato dato lo stesso input, lo stesso ambiente di esecuzione e deve fornire gli stessi risultati, permettendo in tal modo il più facile riconoscimento dell'origine di eventuali difetti riscontrati.

Metodi di Analisi Dinamica:

- **Test di unità:** test che vengono applicati alle singole unità di sistema al fine di verificarne la presenza di malfunzionamenti
- **Test di integrazione:** rappresenta il test effettuato su più unità software messe in comunicazione tra di loro per verificare che le loro interazioni siano quelle previste
- **Test di sistema:** rappresenta il test effettuato su tutte le unità software messe in comunicazione tra di loro per verificare che il prodotto nel suo insieme abbia le funzionalità e le caratteristiche previste
- **Test di regressione:** tutte le volte che si applica una modifica ad una parte del codice precedentemente testato e dimostrato funzionante, devono essere svolti tutti i test di unità e integrazione ad essa relativa, eventualmente aggiungendo nuovi test se necessario, al fine di non pregiudicare le funzionalità già verificate. Se due test risulteranno simili, il meno efficace tra i due dovrà essere eliminato
- **Test di accettazione:** collaudo controllato dal committente. Se il collaudo viene superato in modo positivo, il sistema viene rilasciato e la commessa si conclude

4.3 Misure e metriche

Le misure rilevate dal processo di verifica, che permettono di ottenere informazioni quantitative relative ai prodotti di ogni processo produttivo (sia esso progettuale o di codifica), devono essere basate su metriche stabilite a priori quando è possibile ritenere di avere sufficienti conoscenze dell'argomento trattato. Qualora ci fossero metriche, riguardanti determinate misure, ancora incerte e approssimative, si cercherà di provvedere a stabilirle precisamente durante il corso della progettazione e realizzazione del prodotto, attraverso l'analisi di ciò che sarà stato realizzato fino a quel momento e grazie anche all'apprendimento graduale del gruppo.

Le misure e le metriche suddette sono le seguenti:

- **Complessità ciclomatica:** rappresenta la complessità di un metodo, basata sulla misurazione del numero di cammini linearmente indipendenti che attraversano il grafo di flusso di controllo, nel quale i nodi rappresentano gruppi indivisibili di istruzioni e gli archi connettono due nodi se le istruzioni di un nodo possono essere eseguite immediatamente dopo le istruzioni dell'altro nodo. Un valore elevato di complessità riduce la manutenibilità e le possibilità di riuso del metodo: se dovesse risultare tale, parte delle sue funzionalità deve essere demandando ad altri metodi da richiamare. Particolare attenzione va posta sulla stima di questo valore: il costrutto switch, ad esempio, moltiplica i cammini linearmente indipendenti e quindi può, molto facilmente, comportare una misurazione di complessità ciclomatica molto elevata (e comunque oltre i limiti imposti). Tuttavia potrebbe verificarsi l'eventualità che, al fine di garantire una velocità di esecuzione maggiore per un certo metodo, si decida di assumere limiti di complessità ciclomatica più laschi. I valori di complessità ciclomatica misurati verranno trattati, dunque, con le dovute considerazioni. Il valore ideale di complessità ciclomatica massima posto come obiettivo è dieci

- **Peso delle classi:** è la somma totale della complessità ciclomatica di tutti i metodi appartenenti ad una determinata classe. Il valore deve essere ragionevole, e dove venisse ritenuto troppo alto, si procederà ad aumentare il numero di metodi della classe in considerazione, oppure verrà affidata parte dei compiti ad una o più nuove classi coese

- **Numero di parametri:** indica il numero di parametri formali per metodo. Un numero molto elevato può essere ridotto, e quindi migliorato, grazie alla creazione di ulteriori classi che contengano più parametri tra loro correlati, aumentando la manutenibilità e l'astrazione del codice. La finalità è quella di mantenere un numero massimo di parametri pari a dieci

- **Numero di campi dati per classe:** un numero elevato di attributi interni ad una classe potrebbe palesare la necessità di incapsulamento di parte di essi in nuove classi coese, che forniscano inoltre metodi utili al loro utilizzo

- **Numero di variabili locali:** denota il numero di variabili locali interne a ciascun metodo. Come per il numero di campi dati per classe, potrebbe essere necessario incapsulare alcune di queste variabili in nuove classi coese qualora il numero sia troppo elevato
- **Numero di livelli di annidamento:** rappresenta il numero di livelli di annidamento dei metodi. Un valore elevato determina alta complessità e riduce il livello di astrazione del codice: per questo deve mantenersi a livelli bassi
- **Grado di accoppiamento:**
  - **Indice di utilità:** numero di classi esterne al package che dipendono da classi interne ad esso. Deve avere un valore ragionevole: se troppo basso indicherà che il package non fornisce molte funzionalità al suo esterno e quindi potrebbe essere scarsamente utile; se troppo alto indicherà che altre classi sono strettamente dipendenti dal package in questione, e quindi potrebbe accadere che eventuali modifiche ad esso comportino costi elevati di adattamento delle classi che vi dipendono, qualora non fosse stato progettato adeguatamente il sistema di interfacce
  - **Indice di dipendenza:** numero di classi interne al package che dipendono da classi esterne ad esso. In generale va sempre minimizzato, aumentando quindi le funzionalità proprie di un package, senza la necessità di affidarsi al servizio offerto da altre classi esterne

5 Gestione amministrativa della revisione

5.1 Comunicazione e risoluzione di anomalie

Un'**anomalia** consiste in una deviazione del prodotto dalle aspettative prefissate. Per la gestione e risoluzione di anomalie ci si affida allo strumento di ticketing messo a disposizione dal servizio Github nell'apposita sezione "Issue". Il Verificatore, per ogni anomalia che riscontrerà e non ancora individuata da altri verificatori, dovrà aprire un nuovo ticket. Per la struttura e risoluzione di un ticket vedere `Norme_di_Progetto_v1.0`.pdf.

5.2 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.

5.3 Procedure di controllo di qualità di processo

L'organizzazione interna dei processi si basa sul principio PDCA, in grado di garantire un miglioramento continuo della qualità di tutti i processi (compreso quello di verifica stesso) e, conseguentemente, dei prodotti risultanti. I processi devono essere pianificati dettagliatamente rispetto ai requisiti richiesti e alle risorse disponibili, e quindi attuati secondo il piano. La verifica sui processi avviene attraverso l'analisi costante delle misurazioni sul prodotto di ciascun processo e sul processo stesso. Se l'analisi di tali misure evidenzia valori che si discostano, in modo peggiorativo, dai piani prefissati, questo denoterà la presenza di un problema: per ognuno di essi che verrà identificato, si provvederà a stabilirne le cause e le possibili soluzioni, intervenendo in modo correttivo sul processo ed, eventualmente, sul piano iniziale ad esso relativo. Le misurazioni sul processo consistono principalmente in:

- Tempo impiegato per essere completato
- Cicli iterativi interni al processo
- Attinenza ai piani stabiliti
- Soddisfazione dei requisiti richiesti

- Risorse utilizzate e/o consumate durante il processo

Se non vengono identificati problemi relativi ad un processo, è possibile intervenire comunque in modo migliorativo: tale miglioramento consiste evidentemente nel ridurre il numero di cicli iterativi, risorse e tempo utilizzati, comunque garantendo che l'esecuzione del processo sia fedele al piano e soddisfi i requisiti, aumentandone quindi la sua efficienza e determinando una variazione nel grado di efficacia non negativo.

6 Pianificazione ed esecuzione del collaudo

Questa sezione verrà redatta quando il prodotto **MyTalk** sarà prossimo al collaudo.

7 Appendice

7.1 Resoconto delle attività di verifica

7.1.1 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 4.2.1 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 Hunspell ed Enchant, 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 nella sezione 2.1.