Piano di Qualifica

Informazioni sul documento	
Nome file:	piano_di_qualifica.3.0.pdf
Versione:	3.0
Data creazione:	2012-12-10
Data ultima modifica:	2013-03-16
Stato:	Approvato
Uso:	Esterno
Lista di distribuzione:	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Dott. Gregorio Piccoli
Redattori:	Andrea Rizzi
	Stefano Farronato
	Elena Zecchinato
Approvato da:	Diego Beraldin
Verificatori:	Riccardo Tresoldi

Storia delle modifiche

Versione	Descrizione intervento	Redattore	Ruolo	Data
3.0	Approvazione documento	Diego Beraldin	Responsabile	2013-03-16
2.8	Correzione errori riscontrati in fase di verifica	Stefano Farronato	Progettista	2013-03-16
2.8	Verifica lessico ortografica del documenti	Riccardo Tresoldi	Progettista	2013-02-16
2.7	Aggiornata sezione "Dettaglio dell'esito delle revisioni" e "Appendice"	Elena Zecchinato	Progettista	2013-03-16
2.6	Aggiornata sezione "Dettaglio dell'esito delle revisioni" e "Appendice"	Elena Zecchinato	Progettista	2013-03-15
2.5	Aggiunta sezione "Autovalutazione"	Andrea Rizzi	Progettista	2013-03-15
2.4	Aggiornate e corrette definizioni analisi statica e dinamica	Stefano Farronato	Progettista	2013-03-15
2.3	Riscritta sottosezione "Metriche"	Stefano Farronato	Progettista	2013-03-14
2.2	Corretta sezione "Metriche e metodi" in "Tecniche, metodi e metriche"	Elena Zecchinato	Progettista	2013-03-14
2.1	Corretta sezione "Qualità di processo"	Stefano Farronato	Progettista	2013-03-14
2.0	Approvazione del documento	Andrea Meneghinello	Responsabile	2013-01-19
1.8	Verifica finale del documento	Andrea Rizzi	Verificatore	2013-01-18
1.7	Correzione errori riscontrati in fase di verifica	Riccardo Tresoldi	Progettista	2013-01-18
1.6	Verifica lessico ortografica del documento	Elena Zecchinato	Verificatore	2013-01-18
1.5	Aggiornamento capitolo "Prossima versione piano di qualifica" e riscritta sezione "software"	Stefano Farronato	Progettista	2013-01-17
1.4	Creazione e organizzazione capitolo "Appendice" e sezioni interne	Riccardo Tresoldi	Progettista	2013-01-16
1.3	Correzione capitolo riguardante "Misure, Metriche e metodi"	Riccardo Tresoldi	Progettista	2013-01-15
1.3	Stesura capitolo "Resoconto esito delle revisioni" e sezioni interne	Riccardo Tresoldi	Progettista	2013-01-14

1.2	Correzione capitolo “Visione generale strategia di verifica” modificando strutturalmente la parte relativa a “Organizzazione, pianificazione strategica, pianificazione temporale e responsabilità”	Stefano Farronato	Progettista	2013-01-14
1.1	Stesura sezione “qualità di processo”	Stefano Farronato	Progettista	2013-01-13
1.0	Approvazione documento	Elena Zecchinato	Responsabile	2012-12-18
0.12	Correzione errori segnalati dal verificatore	Stefano Farronato	Analista	2012-12-18
0.11	Verifica Capitoli 5,6,7,8,9	Diego Beraldin	Verificatore	2012-12-17
0.10	Verifica Capitoli 1,2,3,4	Diego Beraldin	Verificatore	2012-12-16
0.9	Correzione paragrafo 4.1 e paragrafo 6.1	Riccardo Tresoldi	Analista	2012-12-15
0.8	Correzione paragrafo 6.3 e stesura capitolo ”Riferimenti”	Stefano Farronato	Analista	2012-12-14
0.7	Impostazione capitoli rimanenti: “Resoconto delle attività di verifica”, “Prossima versione Piano di Qualifica”. Correzione paragrafo 6.2.1.	Riccardo Tresoldi	Analista	2012-12-13
0.6	Correzione capitolo “Visione generale della strategia di verifica” nella “risorse necessarie e disponibili”	Riccardo Tresoldi	Analista	2012-12-12
0.5	Stesura capitolo “Gestione amministrativa della revisione”	Stefano Farronato	Analista	2012-12-12
0.4	Stesura capitolo “Strumenti, tecniche e metodi ”, conclusione capitolo “Obiettivi di qualità”	Riccardo Tresoldi	Analista	2012-12-11
0.3	Stesura capitolo “Qualità”	Stefano Farronato	Analista	2012-12-11
0.2	Stesura capitolo “Visione generale della strategia di verifica”	Stefano Farronato	Analista	2012-12-10
0.1	Stesura scheletro documento, introduzione	Stefano Farronato	Analista	2012-12-10

Indice

1 Introduzione

1.1 Scopo del prodotto

Con il progetto "MyTalk" si intende un sistema software di comunicazione tra utenti mediante browser senza la necessità di installazione di plugin e/o software esterni. L'utilizzatore avrà la possibilità di interagire con un altro utente tramite una comunicazione audio - audio/video - testuale e, inoltre, ottenere delle statistiche sull'attività in tempo reale.

1.2 Scopo del documento

Il seguente documento ha lo scopo di presentare la strategia di verifica e di validazione complessiva che utilizzeranno i componenti del team di lavoro di Software Synthesis a scopo di garantire la qualità richiesta nello svolgimento del progetto MyTalk regolarmente accettato dall'azienda appaltatrice Zucchetti S.r.l.

Durante lo svolgimento del suddetto progetto sarà possibile l'insorgere di modifiche a tale documento, dettate da eventuali scelte progettuali o da esplicite richieste del committente stesso.

1.3 Glossario

Al fine di evitare incomprensioni dovute all'uso di termini tecnici nei documenti, viene redatto e allegato il documento *glossario.3.0.pdf* dove vengono definiti e descritti tutti i termini marcati con una sottolineatura.

2 Riferimenti

2.1 Normativi

norme_di_progetto.3.0.pdf allegato;

Riferimenti a specifici estratti del testo *Software Engineering (8th edition) Ian Sommerville, Pearson Education / Addison-Wesley* quali:

ISO/IEC 9126:2001, Software engineering - Product quality - Part 1: Quality model;

ISO/IEC 12207, Software Life Cycle Processes.

2.2 Informativi

analisi_dei_requisiti.3.0.pdf;

piano_di_progetto.3.0.pdf;

glossario.3.0.pdf;

test_e_misurazioni.1.0.pdf;

Capitolato d'appalto: MyTalk, v 1.0, redatto e rilasciato dal proponente Zucchetti S.r.l. reperibile all'indirizzo: `http://www.math.unipd.it/~tullio/IS-1/2012/Progetto/C1.pdf`;

Riferimenti a specifici estratti del testo *Software Engineering (8th edition) Ian Sommerville, Pearson Education / Addison-Wesley* quali:

Software Engineering - Part 5: Verification and Validation, Part 6: Management.

3 Obiettivi di qualità

3.1 Qualità di processo

Per valutare la qualità di processo il team intende considerare il modello SPY. La qualità di processo non può essere trascurata in quanto la qualità del prodotto finale risulta essere una sua conseguenza.

Il modello preso come riferimento, la cui applicazione è illustrata in figura 1, prevede la valutazione del soddisfacimento dei requisiti di qualità al fine di individuare le aree in cui è possibile il miglioramento e le relative modifiche da apportare ai processi.

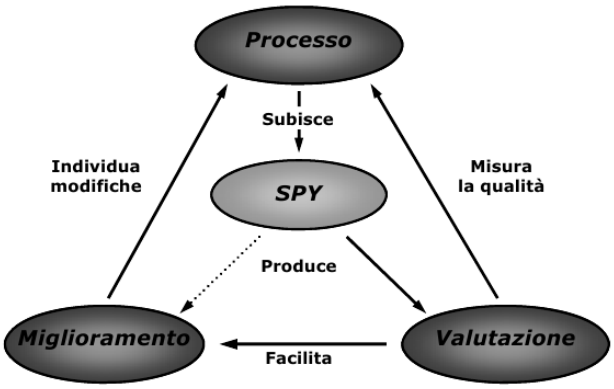

Figura 1: Il modello SPY per la valutazione della qualità di processo.

La qualità di processo è parte fondamentale del processo per garantire la qualità del prodotto, e viene controllata ponendo come riferimento una serie di attributi specifici:

Process performance: un processo raggiunge gli obiettivi prefissati trasformando input identificabili in output identificabili;

Performance managment: un processo è pianificato e controllato allo scopo di produrre risultati coerenti e corrispondenti agli obbiettivi per cui è stato attuato;

Work product management: un processo è pianificato e controllato allo scopo di produrre risultati documentabili, controllati e verificabili;

Process definition: l'attuazione di un processo si basa su un approccio stardardizzato;

Process resource: ogni processo ha a disposizione le adeguate risorse per essere attuato;

Process measurement: le misure e i risultati rilevati durante l'attuazione di un processo vengono usati per assicurarsi che lo stesso avvenga per raggiungere efficacemente gli obiettivi prefissati;

Process control: ogni processo è controllato mediante la raccolta, l'analisi e l'utilizzo delle misure di processo e di prodotto, al fine di valutare ed eventualmente correggere le sue modalità di attuazione;

Process change: il controllo sulle modifiche di gestione, attuazione o definizione di un processo è costante;

Continuous improvement: le modifiche e gli accorgimenti su un processo sono identificati e implementati allo scopo di assicurare il continuo miglioramento al fine di raggiungere più efficacemente gli obbiettivi rilevati.

Il modello fornisce inoltre una scala di valutazione sul grado di maturità dei processi, composta da cinque livelli e riportata in figura 2. Il livello raggiunto dai fornitori viene utilizzato come "biglietto da visita" nella candidatura ai bandi. Questo significa che quanto più elevato è il livello raggiunto dal fornitore tanto migliore è la presentazione che offre di se stesso sullo specifico processo.

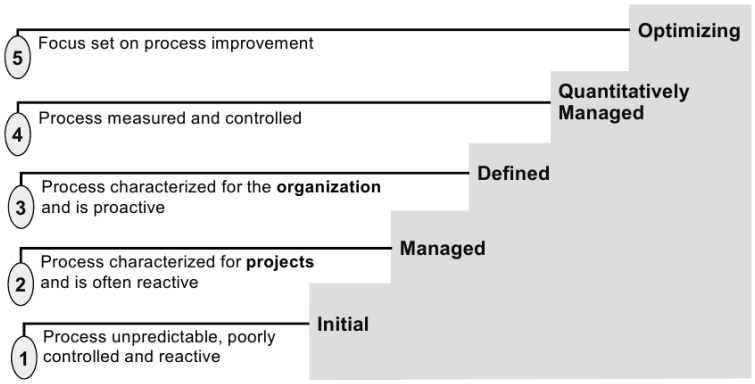

Figura 2: I livelli della classificazione CMMI.

- **Livello 0**: processo non implementato o incapace di raggiungere gli obiettivi prefissati, non vi è inoltre evidenza di alcun approccio sistematico agli attributi definiti;
- **Livello 1**: Il raggiungimento di questo livello è dimostrato attraverso l'attributo specifico *Process performance* in quanto il processo è messo in atto e raggiunge gli obbiettivi prefissati, tuttavia non è evidente nessun approccio sistematico per nessuno degli attributi definiti.
- **Livello 2**: ll raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di *Performance management* e *Work product management*. Il processo è gestito e sulla base di obbiettivi definiti è pertanto attuato, pianificato, tracciato, verificato ed eventualmente corretto.
- **Livello 3**: ll raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di *process definition* e *process resource*. Il processo è definito e sulla base di procedure definite sui principi del *software engineering* è pertanto attuato, pianificato e controllato.
- **Livello 4**: ll raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di *process measurement* e *process control*. Il processo risulta ora predicibile, è stabilizzato ed attuato e rispetta i risultati attesi nonchè le performance e le risorse impiegate.
- **Livello 5**: ll raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di *process change* e *continuous improvement*. Il processo si presta all'ottimizzazione, è predicibile e in grado di adattarsi al raggiungimento di obbiettivi specifici e rilevanti a livello organizzativo.

La qualità dei processi, come già accennato ad inizio sezione, è una condizione necessaria per arrivare ad un prodotto di qualità. Il team si pone l'obiettivo di garantire e ottimizzare la qualità dei propri processi mediante l'utilizzo di strumenti necessari per garantirla, pertanto si aspetta di trarre da tali considerazioni:

- un ottimizzazione dell'uso delle risorse;
- una migliore gestione economica, contenendo al massimo i costi;
- una maggiore possibilità di confronto con delle *best practice*;
- maggiore affidabilità nei tempi di stima rispetto alle consegne preventivate;
- una migliore stima dei rischi e la loro gestione;

Inoltre ogni processo sarà definito in modo dettagliato e valutato l'output prodotto. Infine verranno attuate procedure di misurazione, valutazione ed eventuale modifica sul processo per perseguire il suo miglioramento mediante una costante collaborazione tra il verificatore e lo sviluppatore del processo stesso.

3.2 Qualità di prodotto software

Al fine di garantire un elevato standard qualitativo, sia per ovvia scelta del team di sviluppo, sia per implicita richiesta da parte del committente stesso, Software Synthesis ha preso come riferimento lo standard ISO/9126:2001.

Funzionalità Il prodotto MyTalk deve soddisfare nelle sue funzionalità tutti i requisiti obbligatori individuati nell'attività di analisi, garantendone il funzionamento e l'aderenza alle richieste specifiche del committente. Tutto questo sarà svolto nel modo meno oneroso sia dal punto di vista economico che di sfruttamento delle risorse disponibili.

Per valutare il grado di funzionalità raggiunto dal prodotto si valuterà la quantità di requisiti che sono stati correttamente implementati all'interno del software finale. La soglia minima di soddisfacimento risulta essere l'assoluta copertura dei requisiti obbligatori imposti dal committente, tuttavia è parso chiaro che la fantasia del team in questa specifica area sarà ben valutata.

Portabilità Il prodotto finale dovrà per vincoli di capitolato essere pianamente usufruibile mediante browser Chrome, prodotto da Google, su tutti i sistemi operativi sui quali questo browser risulta compatibile.

A dimostrazione di tale soddisfacimento ci si affida alla dimostrazione del superamento della validazione del codice del *front-end web* e dell'assoluta coerenza con le specifiche WebRTC, proposta evolutiva di HTML5. All'approvazione del superamento di tale requisito si procederà con lo stesso metodo testando browser alternativi a quello imposto al fine di rendere MyTalk usufruibile da un bacino d'utenza più vasto possibile.

Usabilità Il prodotto deve risultare facile ed intuitivo da parte dell'utenza che dispone di una conoscenza medio-bassa del web e dell'informatica in generale.

Utenti che hanno familiarità con programmi per la gestione di chiamate mediante VOIP non dovranno trovare alcuna difficoltà o iniziale disorientamento nell'utilizzo di MyTalk.

Data l'aleatorietà di tale qualità di prodotto e la non "oggettività" nella misurazione di tale caratteristica si cercherà semplicemente di raggiungere tale risultato basandosi su esperienze personali o brevi test su specifici utenti selezionati.

Affidabilità L'applicazione deve riuscire a stabilire e mantenere stabile una comunicazione tra due o più utenti, senza mostrare problemi di natura tecnica se non imputabili alla qualità della connessione di cui dispongono gli utenti stessi. Deve dimostrarsi altresì robusta nella sua struttura e facile da ripristinare in caso di errori di varia natura.

Al fine di garantire queste caratteristiche verrà utilizzata come unità di misura la quantità di interazioni tra utenti con esito positivo, tenendo conto di tutti i parametri che concorrono ad una corretta comunicazione (qualità audio, video, messaggi testuali correttamente inviati/ricevuti, etc.).

Efficienza MyTalk si pone come obbiettivo oltre alla corretta ed appagante esperienza comunicativa, anche di non risultare particolarmente esosa dal punto di vista hardware, sia dal punto di vista puramente componentistico dell'unità dalla quale si accede al prodotto, sia dal punto di vista dell'uso di banda a disposizione della rete.

Verranno pertanto monitorate in fase di test sia le percentuali d'utilizzo di memoria e processore della macchina, sia la quantità di kb/s trasmessi e ricevuti durante l'esecuzione del programma. I test verranno eseguiti su varie tipologie di hardware e linee di diverse velocità, al fine di rendere il software usufruibile dalla più vasta fetta d'utenza possibile.

I test risulteranno superati se nei momenti di massimo consumo di risorse il programma riuscirà a garantire un utilizzo fluido e una discreta navigabilità nel web dalla macchina soggetta al test.

Manutenibilità Il capitolato specifica esplicitamente che la modifica e la manutenibilità del software sono una caratteristica fondamentale dell'intero progetto, questa necessità nasce dal costante utilizzo di linguaggi non ancora qualificati come standard, pertanto soggetti a continua evoluzione.

3.2.1 Ciclo di Deming

Per riscontrare le caratteristiche sopra elencate il team ha deciso di adottare il ciclo di Deming (PDCA).

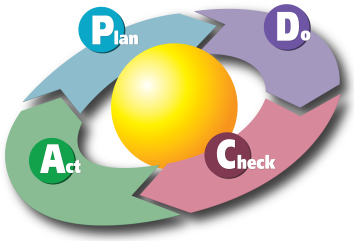

Figura 3: Il ciclo di Deming.

Tale tecnica è costituita dalle seguenti fasi:

PLAN	analizzare il problema e stabilire le attività, le tempistiche e i soggetti coinvolti;
DO	attuare dettagliatamente le attività pianificate;
CHECK	verificare a consuntivo quanto preventivato nella pianificazione;
ACT	nel caso in cui il controllo abbia rilevato discrepanze tra preventivo e consuntivo attuare misure correttive.

Gli obiettivi di qualità prefissati possono essere raggiunti tramite una adeguata pianificazione delle attività, che deve essere rispettata attentamente per poter permettere l'autovalutazione ed eventualmente l'adozione di misure correttive volte al miglioramento continuo e all'attuazione delle *best practices*. Solo in tal modo è infatti possibile ottenere un prodotto di qualità.

3.3 Gestione della qualità

La qualità di un software non si limita all'assenza di problematiche durante il suo funzionamento anche se è evidente che la maggior parte delle attività inerenti al suo controllo hanno come fine ultimo l'identificazione e l'eliminazione di ogni possibile caratteristica che si discosti dal comportamento atteso del programma o dai requisiti che il programma stesso deve soddisfare.

Al fine di rendere ancor più chiara la comprensione di tali problematiche ed evitare che termini all'apparenza simili assumano significati impropri precisiamo che un **malfunzionamento** è un funzionamento di un programma (o di una sua parte) diverso da quanto previsto dalla sua specifica, un **difetto** è la causa di un malfunzionamento, mentre un **errore** è l'origine di un difetto. Tali termini oltre ad avere un significato diverso hanno quindi anche una dipendenza causa-effetto gli uni dagli altri, con una dipendenza temporale ben definita.

Terminiamo il capitolo evidenziando gli aspetti fondamentali per contenere i problemi descritti in precedenza e accertare la qualità del prodotto e dei processi: *Quality Assurance*, Verifica e validazione. Nel primo caso si intende l'insieme della attività svolte a garantire il soddisfacimento degli obbiettivi di qualità prefissati, mentre la verifica deve garantire che avvengano i controlli affinché il prodotto ottenuto al termine di una fase sia coerente con quanto precedente a tale fase. La Validazione è infine un'attività di controllo che confronta il risultato di una fase del processo di sviluppo del prodotto con i requisiti del prodotto stesso.

Verifica e validazione sono attività che coesistono con quelle tradizionali, e la loro integrazione deve essere svolta nel modo più efficace possibile al fine di renderle concretamente produttive in quanto la scoperta di un errore durante lo sviluppo garantisce un guadagno significativo per la qualità del prodotto.

4 Visione generale della strategia di verifica

4.1 Organizzazione, pianificazione strategica, pianificazione temporale e responsabilità

L'attività di verifica inizierà quando il prodotto di un processo raggiungerà uno stadio che si potrà definire diverso da quello precedente a seguito della conclusione di un processo di ciclo di vita.

La verifica di tali cambiamenti sarà operata in modo mirato e circoscritta, grazie al registro delle modifiche che verrà compilato durante la stesura del documento stesso. Al termine della fase di verifica, i documenti saranno consegnati al responsabile di progetto, che provvederà ad approvarli.

Il team nel *brainstorming* successivo all'analisi generale del progetto MyTalk ha deciso di adottare un ciclo di vita incrementale (specificato nel documento *piano_di_progetto.3.0.pdf*).

Coerentemente a tale scelta, il processo di verifica adottato opererà nelle diverse fasi del progetto nel modo seguente:

4.1.1 Analisi dei Requisiti

Quando un documento uscirà dalla fase di redazione, verrà preso in esame ed effettuata una fase di revisione definitiva prima di essere presentato ufficialmente alla RR:

- Verrà presa in esame la correttezza grammaticale, che al contrario di quella ortografica può essere verificata solo mediante un accurata rilettura da parte del verificatore designato.
- Verrà presa in esame la correttezza lessicale mediante un'accurata rilettura da parte del verificatore designato.
- Verrà presa in esame la correttezza dei contenuti e la coerenza rispetto al documento mediante un accurata rilettura da parte del verificatore designato.
- Verrà effettuata la verifica che ogni tabella o figura sia corretta nel suo contenuto e disponga della rispettiva didascalia.
- Verrà presa in esame la correttezza rispetto alle norme di progetto redatte, utilizzando gli strumenti più appropriati per la verifica.
- Verrà presa in esame la corrispondenza tra ogni requisito e i casi d'uso, consultando e controllando le apposite tabelle di tracciamento, verificando inoltre la corretta gestione di entrambi mediante SynthesisRequirementManager, l'applicativo web creato appositamente da Software Synthesis.
- Verrà considerata la corrispondenza fra i requisiti elencati e quanto enunciato nel testo del capitolato, in modo da assicurare che tutti i requisiti espliciti siano stati correttamente recepiti.
- Verrà presa in esame la chiarezza espositiva nell'enunciazione dei requisiti al fine di ridurre al minimo le ambiguità nonché la granularità dei requisiti individuati, perché avere requisiti atomici ne rende più facile il tracciamento e la verifica del soddisfacimento.

4.1.2 Progettazione

Progettazione ad alto livello

Il processo di verifica garantirà la rintracciabilità nei componenti individuati durante la fase di Progettazione architetturale di ogni singolo requisito descritto nel documento di analisi

dei requisiti (versione 3.0). A tale scopo lo strumento che si è scelto di utilizzare è ancora SynthesisRequirementManager, nello specifico tramite il modulo relativo ai *configuration item*.

Tali verifiche verranno svolte al termine della stesura del documento di specifica tecnica e dell'aggiornamento dei documenti relativi al piano di progetto, piano di qualifica, norme di progetto, analisi dei requisiti.

Al fine di garantire un'elevata qualità nell'architettura software elaborata durante la progettazione di alto livello, la verifica dovrà valutare la presenza delle seguenti proprietà:

- **necessità** ogni componente soddisfa almeno uno fra i requisiti emersi in fase di analisi;
- **sufficienza** l'architettura nel suo complesso è in grado di soddisfare tutti i requisiti;
- **modularità** è chiaramente definita la suddivisione delle varie componenti e il ruolo di ciascuna di esse;
- **flessibilità** permette modifiche a basso costo al variare dei requisiti;
- **affidabilità** consente un utilizzo corretto durante il funzionamento;
- **semplicità** ogni parte contiene tutto e solo ciò che è necessario al proprio funzionamento;
- **incapsulamento** le parti del sistema mantengono private le informazioni relative al proprio stato interno e permettono di accedervi solo attraverso le operazioni dell'interfaccia pubblica che espongono verso l'esterno;
- **coesione** le componenti che stanno assieme condividono lo stesso obiettivo;
- **basso accoppiamento** eventuali modifiche a una componente avranno bassa o nulla incidenza sul resto dell'architettura grazie al controllo delle relazioni di dipendenza.

Infine, i diagrammi UML riportati nel documento di specifica tecnica saranno sottoposti a verifica rispetto a quanto stabilito nella sezione 3.4 del documento *norme_di_progetto.3.0.pdf*.

Progettazione di dettaglio

Per ogni unità architetturale prodotta durante la fase di progettazione di dettaglio verrà effettuato un test di unità al fine di verificarne il comportamento dell'unità stessa rispetto a quello atteso in fase d'esecuzione. Tali test sono definiti da:

- un oggetto su cui viene eseguito e gli obbiettivi del il test;
- la strategia e le risorse software utilizzate nella prova;
- il piano d'esecuzione del test stesso.

Allo scopo di ridurre al minimo la presenza di errori interni alle singole unità sono stati programmati un numero di test sufficiente ad assicurare *statement coverage* e *branch coverage*, la prima ha lo scopo di assicurare che tutte le linee di comando di ciascun modulo siano state eseguite almeno una volta, mentre la seconda assicura che tutti i possibili rami del flusso di controllo vengono attraversati almeno una volta.

Effettuati i test di unità si procederà con l'unione delle unità stesse in modo da poter effettuare i relativi test d'integrazione per verificare come le varie componenti del sistema interagiscono tra loro, evidenziando eventuali difetti di progettazione, e dipendenze (o incompatibilità) tra i diversi componenti architetturali.

La pianificazione dei vari test da eseguire verrà fornita dal team Software Synthesis una volta terminata la fase di progettazione di dettaglio.

4.1.3 Codifica

I programmatori svolgeranno le attività di codifica del prodotto e i test di unità per la verifica del codice realizzato nel modo più automatizzato possibile.

I verificatori inoltre controlleranno la presenza di eventuali discrepanze rispetto alle norme pattuite in merito al processo di codifica, nonché l'identificazione di parti di codice soggette ad errori di programmazione. I modelli presi in considerazione per eseguire tali verifiche sono due: verifica funzionale e verifica strutturale.

La verifica funzionale (*black-box*) è una tecnica che procederà con l'accertamento delle funzionalità di un unità controllando esclusivamente che i risultati in uscita rispecchino il valore atteso in ogni condizione d'utilizzo che l'unità dispone. Questo tipo di modello consente test efficaci sull'intero sistema o su unità di dimensioni rilevanti.

La verifica strutturale (*white-box*) al contrario è una tecnica in cui l'ispezione viene svolta a livello di codice sorgente, richiamando esplicitamente tutti i metodi presenti nell'unità al fine di verificarne, oltre al corretto funzionamento, anche la correttezza logica. Questo modello si predilige per controlli sui singoli moduli che costituiscono il progetto, controllando come già citato la struttura stessa del codice.

Specifichiamo infine che per effettuare un controllo approfondito del codice si utilizza il termine "analisi", che si specializza essenzialmente in due tipologie: analisi statica e dinamica.

L'analisi statica non comporta l'esecuzione del codice in quanto la verifica avviene tramite lettura da parte di un verificatore (tramite *inspection* e *walkthrough*, descritti più dettagliatamente nella sezione 5.1.1) o mediante opportuni strumenti software di verifica statica. Gli errori riscontrati mediante tali metodologie dal verificatore non vengono corretti dal soggetto stesso, ma vengono riferiti al progettista o al programmatore al quale comunque rimane il compito di individuazione e correzione degli errori individuati durante il suo lavoro sul codice stesso. L'analisi dinamica richiede l'esecuzione del programma in un ambiente controllato e con dati in ingresso verificati. Verranno registrati tutti i dati relativi all'esecuzione e inerenti alla quantificazione della qualità del software preso in esame, inoltre tali dati saranno confrontati con i requisiti stilati.

4.1.4 Validazione

Alla fase di collaudo, il team garantirà il corretto funzionamento del prodotto MyTalk relativo alle funzionalità,l'usabilità, l'efficienza, l'affidabilità, e la portabilità. Tali garanzie vengono confermate dopo un'attività interna di test denominata Alfa-test. All'esito positivo di tale attività avverrà il collaudo vero e proprio (Beta-test) alla presenza del proponente, successivi difetti riscontrati o eventuali caratteristiche non coerenti alle richieste di quest'ultimo saranno soggette a modifica e correzione al fine di eliminare tali incongruenze.

4.2 Risorse necessarie e disponibili

L'utilizzo di risorse umane e tecnologiche è fondamentale per la verifica di qualità del prodotto e dei processi.

4.2.1 Umane

Software Synthesis si è imposta, per garantire un elevato standard qualitativo, che all'interno del team siano ricoperti, nel corso del progetto, i seguenti ruoli:

- Responsabile: responsabile della corretta realizzazione del prodotto secondo gli standard e le richieste commissionate, designato all'allocazione corretta delle risorse umane ai rispettivi compiti e stimolarne il coordinamento. Controlla inoltre la qualità dei processi interni mediante le attività di verifica da lui predisposte. Infine ha la facoltà di approvare o meno ogni proposta di correzione (migliorativa o di modifica generica) avanzata.
- Amministratore: stila le metodologie e definisce le norme per la verifica dei processi, gestisce inoltre i risultati relativi ai test eseguiti e il processo di gestione e correzione delle anomalie e delle discrepanze.
- Verificatore: applica i processi di verifica e validazione approvati dall'amministratore e predisposte dal responsabile tenendo traccia del suo lavoro. Ogni discrepanza o anomalia incontrerà durante tale attività verrà presentata tramite *ticketing* (vedi capitolo 5 del documento *norme_di_progetto.3.0.pdf*).
- Programmatore: durante il suo lavoro ha l'obbligo di risolvere le anomalie evidenziate tramite *ticketing* dai verificatori o dai test effettuati sul codice da lui stesso prodotto.

4.2.2 Software

A livello software risulteranno necessarie, le seguenti risorse:

SynthesisRequirementManager, il programma basato su interfaccia web sviluppato internamente al team per il tracciamento e la gestione dei requisiti avente lo scopo di standardizzare e rendere quanto più automatizzata possibile tale fase del progetto ed evitare quindi l'insorgere di eventuali sviste dovute all'intervento umano.

Sistema di controllo versione (in particolare `git`) al fine di permettere il lavoro in parallelo da parte dei membri del gruppo e la gestione dello storico delle versioni; il gruppo prevede altresì di appoggiarsi a un *repository* remoto per permettere la sincronizzazione fra i diversi utenti (cfr. sez. 3.1 del documento *norme_di_progetto.3.0.pdf*).

Editor di diagrammi, da impiegarsi tanto per la produzione dei diagrammi UML richiesti per l'attività di progettazione quanto per la creazione dei diagrammi di Gantt necessari alla pianificazione delle attività di progetto.

Sistema di *ticketing*, per gestire la suddivisione e l'assegnazione delle attività ai vari componenti del gruppo in maniera ordinata e controllabile.

Ambiente di sviluppo integrato (IDE), al fine di individuare e correggere già durante l'attività di codifica gli eventuali errori presenti nel codice nonché di renderne più agevole la ristrutturazione (*refactoring*).

Analizzatori statici, vale a dire strumenti per permettere l'automatizzazione nell'analisi del codice prodotto, ai fini di ricavarne il maggior numero possibile di informazioni. Le risorse da utilizzare a tale scopo sono descritte con maggiori dettagli nella sezione 8.1 delle norme di progetto (versione 2.0).

Testing framework, che risulteranno utili ai fini dei test di unità legati al linguaggio di programmazione scelto per standardizzare (e automatizzare) i test sul prodotto finale nonché produrre dei resoconti appropriati sulle eventuali anomalie riscontrate.

***Editor* di documenti**, secondo quanto documentato nella sezione 4.1 delle norme di progetto (versione 2.0), in particolare un editor di testo specializzato per LaTeXcon evidenziazione della sintassi, autocompletamento dei comandi e correzione ortografica integrata.

Gli specifici programmi e *tool* per la gestione di tali servizi sono descritti in dettaglio nel documento *norme_di_progetto.3.0.pdf* allegato.

4.2.3 Hardware

Software Synthesis ha a disposizione oltre al materiale personale di ogni componente del team (computer portatili e fissi) le strutture fisiche ed informatiche messe a disposizione dalla facoltà di informatica dell'Università degli Studi di Padova, quali laboratori didattici e le aule studio allocate negli stabili della Facoltà stessa.

5 Tecniche, metodi e metriche

5.1 Tecniche

Durante lo svolgimento del progetto il team utilizzerà le tecniche di analisi statica e dinamica per effettuare verifiche su software e documentazione.

5.1.1 Analisi statica

L'analisi statica è un tipo di controllo basato sulla non esecuzione del codice, ma in senso lato può essere applicato a qualsiasi tipo di prodotto anche non propriamente eseguibile (ad es. la documentazione di progetto). Sono previste, in particolare, due forme di analisi statica: il controllo manuale (detto altrimenti *desk check*) e il controllo assistito da strumenti automatici. Per quanto concerne il *desk check*, cioè il controllo realizzato unicamente da parte di un agente umano, sono previsti due metodi formali:

- **walkthrough**: implica un esame ad ampio spettro del prodotto da verificare, che è preso in considerazione nella sua totalità in modo indiscriminato e senza alcuna assunzione previa sulla natura, la posizione e la frequenza degli errori da rilevare. Si tratta notoriamente di una tecnica molto onerosa in termini sia di tempo che di sforzo e può essere essa stessa per sua natura soggetta ad errori (in particolar modo falsi negativi). Tuttavia, almeno nelle fasi iniziali del lavoro, è l'unica scelta praticabile a causa della relativa inesperienza dei membri del gruppo nella realizzazione di prodotti complessi e articolati (sia software che documentazione). Allo scopo di ridurre il costo determinato dalla ripetizione di tale attività nell'arco di tutto il ciclo di vita è previsto che durante l'analisi in *walkthrough* sia stilata una lista di controllo relativa agli errori più frequenti e ai contesti in cui è più probabile che si producano errori in modo da collezionare una base di esperienza comune e consolidata destinata ad alimentare le attività di ispezione.

- **inspection**: prevede un controllo mirato avente obiettivi specifici, ristretti e stabiliti a priori *prima* che la verifica abbia luogo. Si tratta di un'attività meno dispendiosa in termini di risorse perché non presuppone l'analisi esaustiva del prodotto ma è focalizzata su determinate categorie di errori frequenti, enunciate in una lista di controllo (*checklist*) redatta sulla base dell'esperienza personale e delle attività di *walkthrough* precedentemente poste in essere.

5.1.2 Analisi dinamica

I controlli dinamici, altrimenti definiti test, prevedono l'esecuzione del software in un ambiente controllato e con dati di input specificatamente pensati per testarne le funzionalità e l'aderenza ai requisiti mettendo in luce l'eventuale presenza di malfunzionamenti dovuti alla presenza di difetti.

Caratteristica fondamentale dei test è la loro **ripetibilità**, cioè dato lo stesso set di dati in ingresso e nello stesso contesto di esecuzione, l'output deve essere deterministico e univocamente determinato. Tale proprietà, unitamente all'auspicabile utilizzo di un *logger* che ha il compito di registrare le fasi dell'esecuzione del test, consente di individuare e riconoscere in maniera più agevole i difetti presenti nel prodotto.

In base al loro ambito di applicazione, i test possono essere suddivisi in:

- test di unità aventi come oggetto le singole unità e, oltre al modulo da verificare e ai dati d'esempio, possono coinvolgere anche componenti attive (driver) o passive (stub) che siano in grado di simulare le parti del sistema non ancora disponibili al momento in cui il test viene eseguito;

- test di integrazione atti a verificare la corretta interazione e integrazione fra le componenti che costituiscono le parti del sistema e hanno come risultato una *build*, vale a dire un sottosistema funzionante che può essere eseguito in modo indipendente;
- test di sistema, volti a testare il rispetto dei requisiti software individuati in fase di analisi dei requisiti da parte dell'intero sistema;
- test di regressione destinati a rilevare il caso indesiderabile in cui una modifica locale destabilizza il resto del sistema, si tratta del numero minimo di test necessario per scongiurare tale eventualità senza per questo dover ripetere in toto i test di unità e di integrazione;
- test di accettazione, o collaudo, realizzato sotto la supervisione del committente per verificare l'aderenza del prodotto ai requisiti utente di più alto livello.

5.2 Metodi

Il processo di misurazione attraverso il quale è possibile valutare quantitativamente l'aderenza del prodotto agli standard di qualità è basato su un ciclo iterativo simile al ciclo di Deming. In particolare, in una fase preliminare prevede che sia stabilita l'importanza e l'ambito di applicazione della misurazione, la selezione di metriche da adottare come si è fatto nelle precedenti sezioni e la pianificazione del momento nel ciclo di sviluppo in cui le misurazioni dovranno essere effettuate.

La parte operativa, cioè la quantificazione dei valori delle metriche di qualità, avviene lungo tutto l'arco del ciclo di sviluppo, con particolare attenzione alle fasi di progettazione di dettaglio e di test effettuati contestualmente alla codifica e in fase di accettazione/collaudo.

Al fine di evitare che la verifica della qualità sia un onere troppo gravoso in termini di risorse umane e di tempo, anch'esso deve essere sottoposto a misurazione in quanto attività di progetto (cioè la parte *check* di PDCA): il tempo di lavorazione impiegato per attività di verifica e QA dovrà dunque essere registrato in ogni momento, ed è prerogativa del responsabile di progetto adottare misure correttive qualora i tempi dovessero risultare eccessivi al punto da compromettere il rispetto delle scadenze stabilite nel piano di progetto.

Il team ha quindi, dopo aver analizzato il problema che andrà ad affrontare, optato per utilizzare le seguenti tipologie di analisi statica:

- **Analisi di flusso di controllo:** il prodotto sviluppato avrà caratteristiche di affidabilità e robustezza, riducendo al minimo gli sprechi. Questa analisi ha il compito di verificare che il codice sia eseguito nella giusta sequenza, correttamente strutturato, e non siano presenti parti di codice non raggiungibili o iterazioni aventi un limite superiore non definito. A tale fine sarà necessario prestare particolare attenzione alle decisioni interne all'unità di codice e verificare che le condizioni risultino totalmente gestite.
- **Analisi di flusso dei dati:** al fine di garantire che in ogni stato in cui si trova il prodotto si riesca ad accedere correttamente ai dati richiesti dall'utente e che nessun cammino d'esecuzione acceda a variabili prive di valore. Questa tipologia di analisi può rilevare anomalie quali scritture successive su una variabile senza letture intermedie o la presenza di variabili *write-only*. Al fine di evitare complicazioni inutili, per questo tipo di analisi si opterà per sconsigliare l'uso di variabili globali in fase di progettazione e codifica.
- **Analisi di flusso di informazione:** per controllare il grado di accoppiamento, determinando le dipendenze tra ingressi e uscite che risultano dall'esecuzione di un'unità di codice e consentendo solo quelle previste dalla specifica degli specifici moduli.
- **Verifica formale del codice:** garantisce la correttezza del codice sorgente sviluppato rispetto alla specifica formale dei requisiti redatta nell'attività di analisi.

5.3 Metriche

Per effettuare delle misure efficacemente è necessario definire un'unità di misura, una scala metrica e il metodo per rilevare tali misure.

Tali caratteristiche sono definite dettagliatamente dallo standard ISO/IEC 9126, e le metriche descritte possono essere catalogate secondo due macro categorie: metriche elementari e derivate. Le prime sono in grado di rappresentare istantaneamente un dato fenomeno, le metriche derivate al contrario sono ricavate da operazioni su altre misurazioni.

Sempre in riferimento alla capacità di immediatezza di misurazione sulle caratteristiche del prodotto si possono distinguere metriche dirette, che sono rilevabili sul software senza considerare l'influenza di fattori esterni (come l'ambiente nel quale è eseguito il software), e metriche indirette che combinano nelle misurazioni sul prodotto anche misure relative a elementi ambientali.

5.3.1 Metriche per il progetto

Relativamente all'andamento e alla gestione del progetto il team ha stabilito di affidarsi a una serie di indicatori al fine di ottenere una stima delle caratteristiche che per loro natura non possono essere misurate in maniera esatta. L'uso di tali indicatori è stato ritenuto opportuno al fine di valutare l'insorgenza di fattori di criticità e controllare il rispetto dei vincoli in termini di risorse (tempi/costi) di progetto.

Gli indicatori selezionati come candidati per l'utilizzo sono, in particolare:

BCWS (*Budgeted Cost of Work Scheduled*) pari al costo preventivato in termini di ore per realizzare i *task* di progetto fino alla data corrente;

ACWP (*Actual Cost of Work Performed*) dato dal costo a consuntivo in termini di ore effettivamente impiegato per lo svolgimento dei *task* di progetto alla data corrente;

BCWP (*Budgeted Cost of Work Performed*) determinato dal valore delle attività terminate alla data corrente;

BV (*Budget Variance*) definito come

$$BV \triangleq BCWS - ACWP$$

ovvero lo scostamento fra quanto preventivato e quanto sostenuto in termini di costi alla data corrente;

CV (*Cost Variance*) definito come

$$BV \triangleq BCWP - ACWP$$

dato dallo scostamento fra il valore di quanto effettivamente prodotto dalle attività di progetto e il costo sostenuto.

5.3.2 Metriche per la verifica del codice

Si riporta un elenco delle principali metriche, in riferimento alle quali il team di sviluppo si ripropone di valutare in modo univoco e quantificabile la qualità del prodotto relativamente alla parte di codifica:

- numero di righe di codice esclusi commenti e annotazioni, considerato nella sua totalità (TLOC, *Total lines of code*) o piuttosto come corpo dei soli metodi (MLOC, *Method lines of code*);

- numero di metodi (NOM, *Number of Methods*) e numero di campi dati di ciascuna classe (NOF, *Number of Fields*);
- profondità di una classe nell'albero di derivazione (DIT, *Depth of Inheritance Tree*);
- numero di metodi ridefiniti (NORM, *Number of OverRidden Methods*)
- indice di specializzazione (IS), definito come

$$IS \triangleq \frac{DIT \times NORM}{NOM}$$

- complessità ciclomatica, basata sul concetto che un metodo può essere rappresentato come un grafo, per calcolare tale complessità vengono sommati tutti i cammini linearmente indipendenti di tale metodo: maggiore è il numero di cammini, maggiore sarà di conseguenza la complessità ciclomatica e la possibilità che in tale codice ci siano errori; tale metrica è definita come

$$v(G) = E - N + P$$

  dove E è il numero di archi, N il numero di nodi e P le componenti connesse da ogni arco e risulta essere molto utile per determinare il numero di test da effettuare su un determinato metodo per coprire tutti i cammini possibili;
- peso della classe (WMC, *Weighted Methods per Class*) definito come la somma della complessità ciclomatica di tutti i metodi membri di una classe;
- mancanza di coesione dei metodi di una classe (LCOM, *Lack of Cohesion of Methods*), un indice che misura quanto i metodi di una classe fanno riferimento ai campi dati della stessa, definito se $m(A)$ è il numero di metodi che riferiscono il campo dati A come

$$LCOM \triangleq \frac{\frac{1}{NOF}\left(\sum_{A \text{ attributo}} m(A)\right) - NOM}{1 - NOM}$$

- indice di utilità (Ca, *Afferent Coupling*) definito come il numero di classi esterne al package che dipendono da una determinata classe;
- indice di dipendenza (Ce, *Efferent Coupling*), definito come il numero delle classi interne al package che dipendono da una classe;
- instabilità (I) definita come

$$I \triangleq \frac{Ce}{Ca + Ce}$$

- astrattezza (a livello di progetto) dato dal rapporto fra il numero di classi astratte/interfacce e il numero totale di tipi;
- la metrica relativa a metodi e procedure $SFIN - SFOUT$, dove SFIN (*structural fan-in*) è il numero di volte che tale metodo è invocato nel corpo di altri metodi e SFOUT (*structural fan-out*) è il numero di invocazioni di metodi esterni all'interno del metodo stesso (corrispondono a Ca e Ce rispettivamente);
- rapporto fra righe di commenti e righe di codice.

Altre metriche che saranno prese in considerazione sono la lunghezza dei metodi e il numero di parametri di ciascun metodo.

5.4 Misure

Il processo di misurazione del software è normato dallo standard ISO di riferimento: ISO 15939 *Software Engineering - Software measurement process* (2002).

Tale processo è descritto come ciclo iterativo che prevede un'attività di misurazione, la raccolta di *feedback* derivata dai risultati e un'eventuale impostazione di azioni correttive per migliorare il processo produttivo.

Nel ciclo di vita del software, è possibile misurare lo stato raggiunto dal prodotto nei suoi vari stadi di lavorazione al fine di giungere a due previsioni distinte:

- prevedere le caratteristiche del software in una fase del ciclo di vita diversa da quella in cui si sta effettuando la valutazione;
- stimare le caratteristiche del software prodotto nello stadio di sviluppo raggiunto in cui si sta effettuando la valutazione.

Vi sono diversi momenti, durante tutti il ciclo di vita del progetto, che richiedono una valutazione delle caratteristiche del software che si sta producendo:

- **Fase di progettazione:** prevedere eventuali problemi nel software in un periodo temporale successivo al suo rilascio e quanto sarà facile eseguire eventuali interventi di manutenzione, valutandone l'impatto che avranno nel contesto d'utilizzo.
- **Fase di collaudo:** confrontare che quanto prodotto sia coerente con le specifiche del committente, analizzando eventuali problemi che possono emergere e non sono stati considerati nell'attività di progettazione.
- **Periodo successivo alla data di rilascio:** misurare l'impatto del software sull'efficienza e l'efficacia del lavoro svolto dall'utente che ne usufruisce, eventualmente confrontando il prodotto con eventuali concorrenti simili e valutandone potenziali aree di miglioramento, pianificando aggiornamenti mirati e creando, eventualmente, una versione migliorata del prodotto stesso.

Indispensabili sono inoltre delle misurazioni effettuate sia su entità semilavorate che compongono il prodotto software che sugli stati finali delle stesse. Si possono dividere tali misure in interne, che hanno una stretta dipendenza con le attività di sviluppo del prodotto, ed esterne, che non dipendono dalle fasi del ciclo di vita.

Le misure interne che si possono adottare sono:

- **Nell'analisi dei requisiti:** utili per stimare il costo del software e le implicazioni sul ciclo di vita del software, in questa fase si può misurare il numero di requisiti e di casi d'uso evidenziati.
- **Nella progettazione:** stimano in modo più accurato il costo del software e alcuni aspetti inerenti alla qualità. In questa fase è possibile quantificare il numero di moduli in uno schema di progettazione e il loro livello di coesione ed accoppiamento.
- **Nel codice:** come per le misure nella fase di progettazione si hanno stime a livello economico e qualitativo, misurando il numero di linee di codice prodotto, la complessità ciclomatica dello stesso, di coesione funzionale ed eventuali misure *object oriented* che valutano l'ereditarietà, il polimorfismo, l'incapsulamento e altre caratteristiche correlate alla programmazione orientata agli oggetti.

Le misure esterne sono principalmente basate sul modello di qualità di Boehm, che si sviluppa mediante approccio *top-down*: a livello gerarchico nelle parti superiori compaiono gli attributi qualitativi dal punto di vista del committente, mentre nei livelli inferiori gli attributi dal punto di vista del fornitore.

Tali qualità sono organizzate in modo gerarchico ad albero, le cui foglie sono composte dalle metriche, ovvero gli elementi direttamente osservabili e misurabili, come l'indipendenza dalla piattaforma, l'accessibilità, l'efficienza e la leggibilità.

6 Gestione amministrativa della revisione

6.1 Gestione anomalie e incongruenze

Un anomalia è una deviazione dalle aspettative definite sul prodotto, per la loro gestione è stato pianificato l'utilizzo di un sistema di *ticketing*. Lo strumento scelto dal team è Codebase (`http://www.codebasehq.com/`) che permetterà ad ogni verificatore di aprire un *ticket* per ogni nuova anomalia riscontrata.

I *ticket* sono strutturati nel modo seguente:

- **Titolo:** comprende il nome del file da modificare e una descrizione stringata dell'errore;
- **Tipologia**: indica la tipologia del *ticket* aperto, nel caso di anomalie di carattere tecnico sarà impostato a Bug.
- **Categoria Errore:** individua macroscopicamente il tipo di errore preso in esame.
- **Stato:** tiene traccia dell'attuale stato del *ticket* ed è catalogato in cinque categorie:
  - **Nuovo:** *ticket* appena creato dal Verificatore, rappresenta lo stato iniziale di ogni nuova pratica.
  - **Accettato:** stato booleano che identifica l'approvazione da parte del Responsabile di Progetto e ne assegna la pratica ad uno specifico soggetto (vedi punto Responsabile Correzione).
  - **In gestione:** il soggetto assegnato si sta occupando del problema e della relativa correzione dell'anomalia riscontrata.
  - **Corretto:** l'anomalia è stata correttamente gestita e risolta, il Responsabile di Progetto può considerare il *ticket* chiuso.
  - **Non Valido:** l'anomalia proposta dal verificatore viene respinta dal Responsabile di Progetto in quanto non sussiste o semplicemente è già trattata in un altro *ticket*.
- Priorità: determina la priorità con cui dev'essere gestita l'anomalia, può essere di tre tipi

  Critica: l'anomalia risulta essere ad un livello che non permette l'esecuzione del prodotto;

  Alta: l'anomalia provoca diversi problemi nell'utilizzo del prodotto.

  Media: l'anomalia si presenta di moderata priorità

  Bassa: l'anomalia permette la corretta esecuzione del programma, la sua gestione ha una bassa priorità generale.
- Responsabile Correzione: selezionato dal Responsabile di Progetto, ha il compito di gestire e correggere l'anomalia.
- Note: comprende informazioni dettagliate redatte dal soggetto che ha riscontrato l'anomalia. Può comprendere la descrizione dell'errata esecuzione del prodotto a causa dell'anomalia e il comportamento corretto atteso nonché il tempo stimato per la correzione del problema stesso.
- Tag: indicano i termini inerenti all'anomalia per rendere la ricerca e l'identificazione più rapida ed efficace possibile.

Quando il progetto MyTalk giungerà al termine del processo produttivo sarà stilato un documento che analizzerà tutti i test effettuati e i relativi risultati che una volta analizzato dal Responsabile di Progetto potrà certificare la correttezza del prodotto finale.

Trattiamo ora le **discrepanze**, ovvero degli errori di coerenza tra il prodotto che si è realizzato e quello atteso. Tale errore si presenta in una forma di diversa gravità in base a quanto la discrepanza si distanzia dal risultato atteso, tuttavia verrà trattata come una normale anomalia e gestita tramite *ticketing*.

La discrepanza generalmente può essere identificata come un allontanamento dalle norme di progetto o dai requisiti specificati nella fase d'analisi, nel primo caso il Responsabile di Progetto notificherà all'Amministratore che prenderà le dovute contromisure; nel secondo si identificherà il requisito associato al problema e se ne valuterà la discrepanza e le relative modifiche correttive per renderlo coerente al requisito stesso.

6.2 Procedure di controllo di qualità di processo

La qualità del processo si basa su un ciclo continuo di miglioramento (fondato sul processo stesso e sull'uso ottimale delle risorse disponibili).

Per garantire questo miglioramento continuo, l'organizzazione dei processi si basa sul principio PDCA. La verifica che i processi rispettino la pianificazione rispetto ai requisiti stilati e alle risorse disponibili avviene attraverso l'analisi costante delle misurazioni sul prodotto del processo stesso.

Se i risultati di tali verifiche evidenziano valori che denotano ad un peggioramento qualificativo da quelli prefissati nella pianificazione, si provvederà ad identificare e a stabilire le possibili soluzioni al problema che li ha generati, intervenendo in modo correttivo sul processo.

Risulta evidente che è possibile intervenire in modo migliorativo su un processo a prescindere dalla rilevazione di problematiche su di esso. Tale miglioramento consiste generalmente nella riduzione di risorse temporali o fisiche utilizzate o ridurre i cicli iterativi garantendo l'esecuzione fedele del processo rispetto al piano, il mantenimento del suo grado di efficacia ma allo stesso tempo aumentandone l'efficienza.

7 Dettagli dell'esito delle revisioni

Ad ogni revisione che il team sostiene, il committente richiede la realizzazione di una presentazione in cui tutti i componenti del team stesso sono tenuti a partecipare per esporre lo stato di avanzamento del progetto e i punti salienti esposti nella documentazione consegnata al committente stesso nei giorni precedenti all'incontro.

Il committente valuterà complessivamente la qualità dei documenti e l'esposizione del gruppo, stabilendo un giudizio complessivo del lavoro svolto e stilando un breve elaborato in vengono sottolineate mancanze o particolari accorgimenti al fine di migliorare ciascun documento.

Il team ha quindi il compito di approntare prontamente le correzioni e le modifiche richieste, allineando i documenti segnalati alle aspettative del committente.

7.1 Revisione dei requisiti

In seguito alle osservazioni evidenziate durante la Revisione dei Requisiti, il team si è impegnato ad analizzare le tematiche e gli errori evidenziati dal committente, modificando i documenti relativi e realizzando una versione migliorata degli stessi.

Viene riportato in dettaglio le modifiche effettuate dal gruppo per risolvere le problematiche evidenziate:

- **Analisi dei requisiti**: Aggiunta una lista di distribuzione e un sommario al documento. Revisione dei requisiti ed espansione di alcuni di questi (e.g. RUFO1.0.0, RUFO2.0.0 e RUFO3.0.0). Riviste sez. 3.3 e 3.4 e vincoli generali che non sono stati riportati nel documento ma citati nel capitolato. Rivisitati e corretti i diagrammi dei casi d'uso evidenziati, eliminando gli errori e le imprecisioni riscontrate, completando o integrando la descrizione dove suggerito. Aggiunte didascalie esplicative a figure sprovviste. Inserito il tracciamento fra requisiti e casi d'uso.

- **Piano di progetto**: corretto il termine "fase" in quanto non designa attività, ma solo un segmento temporale continuo. Corrette nella forma e nel contenuto la tabella 14. Chiarita la pianificazione relativa alla *milestone* RP ed estesa la correzione anche alle altre attività del piano. Correzioni degli errori lessicali riscontrati.

- **Piano di qualifica**: rivisto completamente per renderlo coerente con le specifiche che tale documento dovrebbe contenere in quanto è stata riscontrata un errata interpretazione dello scopo del documento stesso. Aggiunta la parte relativa al resoconto delle attività di verifica svolte sulla documentazione presentata.

- **Norme di progetto**: corretti gli errori tipografici e terminologici. Aggiunte norme relative alla progettazione e regole per la gestione dei cambiamenti e per garantire l'assenza di conflitto d'interessi nello svolgimento della attività di verifica e di approvazione.

- **Studio di fattibilità**: approfondite le tematiche relative alla valutazione dei capitolati scartati.

7.2 Revisione di progettazione

Dalla Revisione di progettazione sono emerse delle osservazioni mirate al fine di migliorare sia la parte relativa alla filosofia progettuale descritta nella specifica tecnica che migliorativa nei confronti dei restanti documenti consegnati. Il team come nella precedente revisione si è impegnato ad integrare e correggere tali discrepanze evidenziate, procedendo nella sua attività di miglioramento della documentazione stessa.

- **Analisi dei requisiti**: rivista la lista di distribuzione, correzioni a dettagli nel testo di alcuni use-case (UC1.2.X), ulteriore controllo sull'atomicità di alcuni requisiti evidenziandone alcuni in cui era necessaria un'ulteriore scomposizione. Rivista inoltre la denominazione di alcuni requisiti in quanto dichiarati erroneamente come "qualitativi".
- **Piano di progetto**: aggiunto il paragrafo relativo al preventivo a finire come segnalato in fase di revisione, integrato inoltre il capitolo inerente ai rischi di progetto aggiornandoli a quanto riscontrato durante l'avanzamento del progetto. Integrate infine con maggiori dettagli esplicativi la descrizione di ogni attività svolta durante il ciclo di vita progettuale.
- **Piano di qualifica**: applicati gli obbiettivi di qualità elencati e non descritti nello specifico del progetto affrontato sia per quanto concerne la qualità del prodotto che per quella relativa al processo. Rivista la sezione relativa alle "Tecniche, metodi e metriche" estendendolo inoltre con una descrizione più dettagliata delle metriche utilizzate. Aggiunta infine un capitolo relativo alle considerazioni per il miglioramento dei processi e dei ruoli svolti dai soggetti del team.
- **Norme di progetto**: resi più procedurali i contenuti, integrando e modificando la struttura del documento con ulteriori dettagli normativi da applicare al progetto.
- **Specifica tecnica**: eliminato il riferimento errato alla "lentezza" del linguaggio Java in quanto non incidente nel progetto sviluppato, correzioni lessico-ortografiche e "stilistiche" segnalate. Revisione a livello generale della filosofia progettuale partendo dalle segnalazioni sollevate in fase di revisione con una più scrupolosa analisi delle implementazioni a livello strutturale, logico e di utilizzo dei design pattern. Descritti infine con maggior dettaglio alcuni diagrammi di classe e diagramma di attività non sufficientemente esposti nel documento.

8 Considerazioni per il miglioramento

Seguendo la logica del PDCA, Il team ha ritenuto necessario stendere una sezione che aiuti i membri del gruppo stesso a migliorare il loro operato sotto i 3 aspetti essenziali dello sviluppo di progetto:

- qualità dei processi;
- ruoli ricoperti dai membri;
- strumenti utilizzati.

Tale considerazioni vengono quindi riportate in corrispondenza della fase di "check" relativa al PDCA. Mancando un ente terzo in grado di giudicare i temi trattati in modo oggettivo, il team ha deciso di affrontare tale attività in termini di autovalutazione. Consci del fatto che tale procedura rischia di non essere ottimale quanto un giudizio esterno, può risultare comunque interessante al fine di:

- incentivare i membri del team a ricercare gli errori al termine di un attività svolta, cosi da migliorare la ripetizione della medesima;
- incentivare i membri del team ad autogiudicare il proprio operato integrandolo con il giudizio altrui;
- incentivare i membri del team a ricercare tecniche d'utilizzo del software sempre più pratiche ed economiche (in termini sia di tempistiche che di effettivo costo strumentale).

L'obbiettivo finale è quello di attuare, sulla base di tali considerazioni, un miglioramento continuo che si ripeterà durante tutti il progetto, aiutando il team a "crescere" non solo in termini di professionalità, ma anche a migliorare l'integrazione, la coesione e la comunicazione interna al gruppo di lavoro stesso.

Ciò potrà ridurre il livello di rischi interpersonali e nel contempo migliorare il livello di qualità dei processi e infine del prodotto stesso.

8.1 Modalità di autovalutazione

Al fine della praticità, si vuole che l'autovalutazione sia composta da:

- errore riscontrato con connessa motivazione;
- giudizio costruttivo

Tali commenti saranno presi in considerazione da ogni membro del team e per ogni "tema" di autovalutazione che compare nel paragrafo precedente.

Nello specifico al termine di ogni fase di sviluppo ogni membro del team sarà richiamato a fare autovalutazione. Tale attività si svolgerà con la compilazione di un apposito documento, fornito di volta in volta dall'amministratore. Il documento dovrà riportare i seguenti punti:

- In merito ai ruoli svolti in questa attività del progetto, che errori pensa di aver commesso?
- Come pensa si possano evitare gli errori evidenziati al punto precedenti?
- In merito agli strumenti utilizzati, crede sia possibile migliorare il loro utilizzo? Se si come?

L'amministratore avrà poi il compito di prendere in considerazione tali documenti, analizzarli e riportarne il contenuto nelle sezioni successive utilizzando una forma appropriata.

8.2 Autovalutazione dei ruoli fino RQ

8.2.1 Responsabile

Errori riscontrati:

- difficoltà nell'assegnazione dei ruoli e dei ticket dovuta, inizialmente, ad una scarsa conoscenza delle competenze dei singoli membri del gruppo;
- non aver applicato in maniera corretta e approfondita nella realizzazione del progetto alcuni concetti teorici solo descritti nella documentazione;
- difficoltà scarsa esperienza nel stimare le risorse in maniera corretta;
- difficoltà nel cogliere la situazione generale dello stato dell'avanzamento progettuale;
- difficoltà nell'esercitare l'autorità a causa di problemi caratteriali e una non sempre elevata attitudine al comando;
- errori nella creazione dei ticket: non sempre i ticket creati presentano una buona "granularità". È stato osservato che alcuni ticket risultavano essere eccessivamente onerosi per essere assegnati ad un unico membro del team.

Suggerimenti per l'automiglioramento:

- aumentare la conoscenza delle competenze dei vari membri del team, al fine di poter assegnare correttamente i ruoli e le responsabilità;
- il rispetto delle pratiche e norme inserite nei documenti deve essere sistematico e disciplinato. Il responsabile non dovrà lasciare che le tempistiche lo portino ad abbandonare una buona norma di lavoro;
- come osservato l'inesperienza può portare a diversi errori (soprattutto di valutazione). Per evitarli si consiglia il confronto con chi ha svolto l'attività di responsabile in precedenza, soprattutto in caso di dubbio;
- controllare in maniera rigorosa i commit dei membri del gruppo, rilasciando eventualmente commenti appropriati e costruttivi.

8.2.2 Amministratore

Errori riscontrati:

- nello stendere i documenti si è utilizzata una forma eccessivamente verbosa a discapito di una sintassi procedurale;
- a causa di una mancanza di competenza e di conoscenza dell'attività di amministrazione, e degli strumenti di automatizzazione, gli strumenti non sono stati resi immediatamente disponibili. In particolare in merito all'utilizzo di LaTeX, il template utilizzato dal team non è stato sufficientemente documentato. Ciò ha reso difficile la modifica del medesimo da parte degli amministratori successivi che hanno dovuto interpellare il creatore del template rallentando di conseguenza il lavoro.

Suggerimenti per l'automiglioramento:

- fare più uso di diagrammi delle attività, che risultano essere veloci, pratici e chiari;
- migliorare la documentazione sull'utilizzo di determinati strumenti. In merito alla creazione di template o di script di automatizzazione, tali dovranno essere adeguatamente commentati. In caso di dubbi si invita a consultare chi ha svolto l'attività di amministratore in precedenza. Tale pratica (ovviamente costosa) va però svolta con parsimonia e non per banalità. Il giudizio su quando ciò sia o meno necessario è lasciato agli amministratori stessi con la consapevolezza che ciò crescerà con l'esperienza.

8.2.3 Analista

Errori riscontrati:

- errori nella decomposizione dei requisiti. Alcuni di questi non erano stati portati ad un buon livello d'atomicità;
- errori dovuti ad alcuni dubbi sulla rappresentazione degli use case;

Suggerimenti per l'automiglioramento:

- ogni volta che si inserisce un requisito è giusto chiedersi: tale requisito può essere inteso come aggregato di più sotto-requisiti? Se si va suddiviso ulteriormente.
- ogni analista, prima di prendere parte ad un'attività d'analisi è invitato a ripassare il linguaggio UML usato negli use case. In casi di dubbi è anche invitato a consigliarsi con gli altri analisti.

8.2.4 Progettista

Errori riscontrati:

- come per l'analista, anche in questo caso molti errori riscontrati sulla specifica tecnica sono da ricondursi ad uno studio superficiale della sintassi UML;
- lo scarso studio del dominio tecnologico (soprattuto relativa alle conoscenze sul webRTC) hanno portato ad alcuni errori di progettazione dell'architettura di sistema;

Suggerimenti per l'automiglioramento:

- si invitano i progettisti a migliorare le proprie conoscenze in merito al linguaggio UML.
- il dominio tecnologico è importante. Nel progettare il sistema è importante non solo tener conto di quelli che sono i dogmi della programmazione ad oggetti, ma è anche richiesto ai progettisti di analizzare in modo rigoroso le tecnologie da implementare.

8.2.5 Verificatore

Errori riscontrati:

- difficoltà nel rilevare gli errori, dovuti ad una scarsa conoscenza del dominio trattato nei documenti;
- errori dovuti ad eccessiva fretta;

Suggerimenti per l'automiglioramento:

- prendere in considerazione la possibilità di assegnare più tempo all'attività di verifica;
- dare maggior importanza all'attività di verifica, evitando di svolgere frettolosamente il ruolo a causa di tempistiche stringenti.

8.3 Autovalutazione degli strumenti fino RQ

GitHub e sistema di repository: si è riscontrata una certa difficoltà nel gestire i *merge*, pertanto si consiglia di dedicare un po' di tempo per eseguire delle mini esercitazioni nell'uso di tale pratica. Successivamente si dovrà imporre la seguente norma: nessun membro del gruppo dovrebbe lavorare direttamente sul ramo *master*, ma aprirsi un ramo locale , eseguire il proprio lavoro e fare *merge*. Ciò aumenterà il livello di parallelismo nel lavoro del gruppo.

Concept Draw: per quanto tale software si sia rivelato pratico ed intuitivo, esso si è anche rivelato più pesante della norma (in particolare in fase di esportazione dei diagrammi). Si suggerisce quindi (nei prossimi progetti del team) di prendere in considerazione altri software alternativi.

LaTeX: difficoltà nel suo utilizzo sono dovute ad una scarsa documentazione sugli appunti riportati dall'amministratore in merito al suo utilizzo. Si consiglia ai membri del gruppo di prendere visione di tali appunti con maggior serietà. Evitare in assoluto modalità "copia-incolla" di parti di documento. Utilizzare norme appropriate per la stesura dei documenti, evitando *worst practices*.

9 Appendice

9.1 Resoconto delle attività di verifica

In questa sezione verrà descritto in che modo sono state effettuate le attività di verifica e i loro relativi esiti.

9.1.1 Analisi dei requisiti

Oltre alla doverosa analisi statica sui documenti per procedere alla correzione degli errori riscontrati, le verifiche sono state prevalentemente mirate al tracciamento dei requisiti riportati nel documento *analisi_dei_requisiti.3.0.pdf*.

Da questa verifica si possono trarre le seguenti conclusioni

- tutte le funzioni e caratteristiche emerse nel capitolato e nell'incontro con il committente sono coperte dai requisiti; in totale sono stati rilevati 99 requisiti, di cui 22 funzionali obbligatori, 7 funzionali desiderabili e 46 funzionali facoltativi. Si rimanda al documento *analisi_dei_requisiti.3.0.pdf* per maggiori dettagli, relativi anche ai requisiti di qualità, dichiarativi e prestazionali;
- i requisiti sono tutti coperti dai relativi use case. Si rimanda al documento *analisi_dei_requisiti.3.0.pdf* per maggiori dettagli.

Per quanto riguarda la restante documentazione la priorità del team è stata incentrata sulla qualità generale, la pertinenza dei contenuti rispetto allo scopo stesso del documento e la doverosa correttezza sintattica e ortografica.

9.1.2 Progettazione Architetturale

Nell'attività di progettazione si è posta particolare attenzione, oltre alla già citata analisi statica della documentazione, al particolare che ad ogni requisito funzionale espresso durante l'attività di analisi dei requisiti corrisponda almeno una componente architetturale (e viceversa), garantendo pertanto il soddisfacimento dei requisiti stessi e l'assenza di componenti non necessarie.

L'esito di tale tracciamento è consultabile nel documento allegato *specifica_tecnica.2.0.pdf*.

9.1.3 Progettazione di Dettaglio e Codifica

Anche in queste attività si è provveduto all'analisi statica dei documenti, analogamente a quella delle precedenti revisioni. Sono stati inoltre effettuati i test di unità e d'integrazione sul codice prodotto, contenuti nel documento allegato *test_e_misurazioni1.0.pdf*.

10 Prossima versione Piano di Qualifica

È stato preventivato che nella prossima versione del Piano di Qualifica (coerentemente alle norme di progetto redatte si tratterà della versione 4.0) saranno aggiornate le sezioni riguardanti:

- procedure di verifica e validazione del codice prodotto;
- resoconto finale delle attività di verifica nel documento inerente ai *test*.