Piano di Qualifica

Informazioni sul documento	
Nome file:	piano_di_qualifica.2.0.pdf
Versione:	2.0
Data creazione:	2012-12-10
Data ultima modifica:	2013-01-19
Stato:	Approvato
Uso:	Esterno
Lista di distribuzione:	Prof. Tullio Vardanega Prof. Riccardo Cardin Dott. Gregorio Piccoli
Redattori:	Stefano Farronato Riccardo Tresoldi
Approvato da:	Andrea Meneghinello
Verificatori:	Andrea Rizzi

Storia delle modifiche

Versione	Descrizione intervento	Redattore	Ruolo	Data
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**
1.1 Scopo del prodotto . . . 1
1.2 Scopo del documento . . . 1
1.3 Glossario . . . 1

2 Riferimenti **2**
2.1 Normativi . . . 2
2.2 Informativi . . . 2

3 Obiettivi di qualità **3**
3.1 Qualità di processo . . . 3
3.1.1 SPY . . . 3
3.1.2 CMMI . . . 3
3.2 Qualità di prodotto software . . . 4
3.2.1 Ciclo di Deming . . . 5

4 Visione generale della strategia di verifica **7**
4.1 Organizzazione, pianificazione strategica, pianificazione temporale e responsabilità 7
4.1.1 Analisi dei Requisiti . . . 7
4.1.2 Progettazione . . . 7
4.1.3 Codifica . . . 9
4.1.4 Validazione . . . 9
4.2 Risorse necessarie e disponibili . . . 10
4.2.1 Umane . . . 10
4.2.2 Software . . . 10
4.2.3 Hardware . . . 11

5 Misure, metriche e metodi **12**
5.1 Metriche . . . 12
5.2 Misure . . . 13
5.3 Metodi . . . 14

6 Gestione amministrativa della revisione **15**
6.1 Gestione anomalie e incongruenze . . . 15
6.2 Procedure di controllo di qualità di processo . . . 16

7 Dettagli dell'esito delle revisioni **17**
7.1 Revisione dei requisiti . . . 17

8 Appendice **18**
8.1 Resoconto delle attività di verifica . . . 18
8.1.1 Analisi dei requisiti . . . 18
8.1.2 Progettazione Architetturale . . . 18

9 Prossima versione Piano di Qualifica **19**

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.2.0.pdf* dove vengono definiti e descritti tutti i termini marcati con una sottolineatura.

2 Riferimenti

2.1 Normativi

norme_di_progetto.1.0.pdf allegato;

Verbale *verbale_incontro_2012-12-11.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.1.0.pdf;

piano_di_progetto.1.0.pdf;

glosssario1.0.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 due importanti modelli: SPY e CMMI. La qualità di processo non può essere trascurata in quanto la qualità del prodotto finale risulta essere una sua conseguenza.

3.1.1 SPY

Il modello SPY, 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.

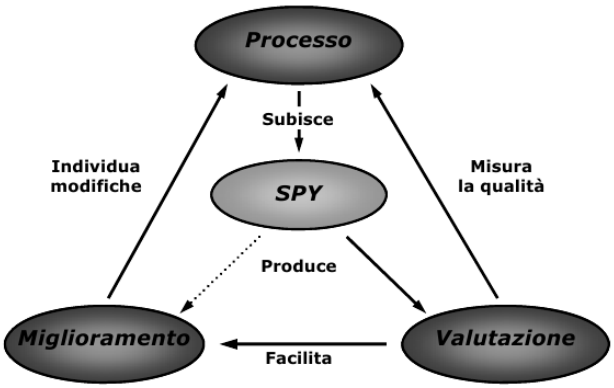

Figura 1: Il modello SPY per la valutazione della qualità di processo.

3.1.2 CMMI

Questo modello è focalizzato sulla qualità interna e permette ai fornitori di autovalutarsi.

Esso infatti individua 24 aree di processo e per ognuna di queste stabilisce gli obiettivi desiderabili e suggerisce delle strategie per raggiungerli. In questo modo l'azienda è in grado di giudicare il proprio livello di maturità per correggersi e quindi migliorarsi laddove non raggiunga il livello desiderato.

In generale, gli obiettivi da misurare sono:

- *capability*: adeguatezza di un processo ai suoi scopi (efficacia);
- *maturity*: livello di governabilità dei processi;
- *model*: requisiti di valutazione del miglioramento;
- *integration*: livello di integrazione della struttura del fornitore.

Le 24 aree possono essere riassunte in quattro gruppi:

process management	comprende tutte le aree relative ai processi organizzativi interni all'azienda;
project management	comprende tutte le aree relative alla gestione dei progetti e dei rischi correlati;
engineering	comprende le aree relative alla gestione dei requisiti e alla verifica e validazione;
support	comprende le aree relative ai processi di supporto.

Il modello CMMI fornisce inoltre una scala di valutazione che è composta da cinque livelli 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.

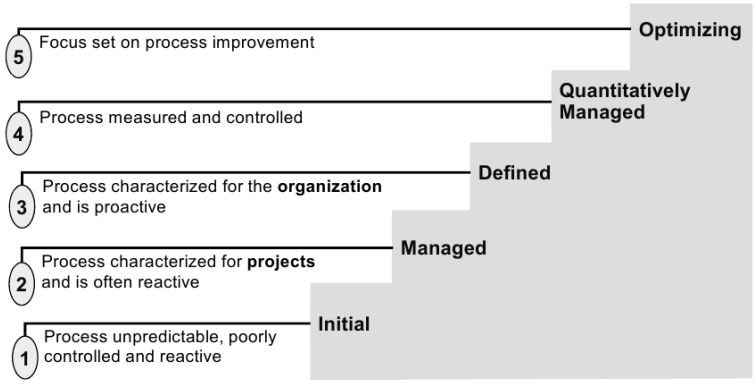

Figura 2: I livelli della classificazione CMMI.

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 generica.

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).

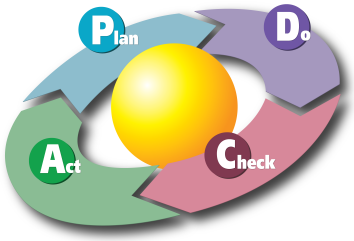

Figura 3: Il ciclo di Deming.

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.

È chiaro che gli obiettivi di qualità prefissati possono essere raggiunti tramite una adeguata pianificazione delle attività, che deve essere rispettata attentamente per poter permettere al

team un'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.

4 Visione generale della strategia di verifica

4.1 Organizzazione, pianificazione strategica, pianificazione temporale e responsabilità

Il processo di verifica inizierà quando il prodotto di un processo raggiungerà uno stadio che si potrà definire diverso da quello precedente. La verifica di tali cambiamenti sarà operata in modo mirato e circoscritta, grazie al registro delle modifiche che verrà compilato durante lo stilamento 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 Piano di Progetto).

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 nell'Analisi dei Requisiti. 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.2.0.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 di 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: non comporta l'esecuzione del codice in quanto la verifica avviene tramite lettura da parte di un verificatore (tramite inspection e walkthrough) 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.

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.

Ai fini pratici i test hanno il fine di scoprire malfunzionamenti o la presenza di eventuali difetti nel software, mentre se effettuato per uno scopo prevalentemente di controllo ha il semplice obbiettivo di verificare la correttezza funzionale (più o meno mirata) del programma. Controlli più mirati con tale tipo d'analisi possono comprendere fattori d'affidabilità, usabilità o efficienza del software.

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).
- 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.2.0.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.

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 Misure, metriche e metodi

5.1 Metriche

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

$$IS := \frac{DIT \times NORM}{NOM}$$

- complessità ciclomatica (o complessità condizionale), un indice che misura all'interno di un metodo il numero di cammini distinti che il flusso di controllo può intraprendere nel codice sorgente incrementando un indice di 1 unità per ogni istruzione di branch (if, for, while, do case, catch, operatore condizionale ternario, operatori logici cortocircuitati);
- 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

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

- 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

$$I := \frac{Ce}{Ca + Ce}$$

- 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 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.

5.2 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 ed quanto sarà facile eseguire eventuali interventi di manutenzione, valutandone l'impatto che avranno nel contesto d'utilizzo.
- **Fase di collaudo:** confrontare che quanto sarà 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 Use Case 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.

Mentre 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.

5.3 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 PDCA. 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' del ciclo di Deming): 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.

In via preliminare, si prevede di utilizzare i seguenti metodi di analisi:

analisi del flusso di controllo volta ad assicurare che il codice sia ben strutturato e non contenga punti non raggiungibili;

analisi del flusso dei dati per scongiurare l'utilizzo inconsapevole di variabili non inizializzate e rilevare variabili inutilizzate o 'write-only';

analisi del flusso di informazione al fine di assicurare l'assenza di dipendenze fra valori di variabili incoerenti con quanto previsto in fase di design architetturale (a livello di modulo, di componente o di sistema nella sua totalità).

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.

8 Appendice

8.1 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.

8.1.1 Analisi dei requisiti

Le verifiche sono state prevalentemente mirate al tracciamento dei requisiti riportati nel documento *analisi_dei_requisiti.2.0.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 20 funzionali obbligatori, 5 funzionali desiderabili e 46 funzionali facoltativi. Si rimanda al documento *analisi_dei_requisiti.2.0.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.2.0.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.

8.1.2 Progettazione Architetturale

Nella fase di progettazione si è posta particolare attenzione 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.1.0.pdf*.

9 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 2.0) saranno aggiornate le sezioni riguardanti:

- procedure di verifica e validazione del codice prodotto;
- resoconto finale delle attività di verifica della fase di progettazione;