



MyTalk

Norme di Progetto

Informazioni sul documento

Nome file:	norme_di_progetto.2.0.pdf
Versione:	2.0
Data creazione:	2012-12-01
Data ultima modifica:	2013-01-18
Stato:	Approvato
Uso:	Interno
Lista di distribuzione:	Membri del Team
Redattori:	Andrea Meneghinello Diego Beraldin
Approvato da:	Riccardo Tresoldi
Verificatori:	Marco Schivo Stefano Farronato

Storia delle modifiche

Versione	Descrizione intervento	Membro	Ruolo	Data
2.0	Approvazione documento	Riccardo Tresoldi	Responsabile	2013-01-23
1.6	Correzione errori rilevati dal verificatore	Diego Beraldin	Amministratore	2013-01-22
1.5	Validazione del documento	Stefano Farronato	Verificatore	2013-01-22
1.4	Aggiornata sezione software di tracciamento e verifica	Diego Beraldin	Amministratore	2013-01-17
1.3	Conclusa sezione progettazione	Diego Beraldin	Amministratore	2013-01-16
1.2	Aggiunta sezione progettazione	Diego Beraldin	Amministratore	2013-01-15
1.1	Correzione errori rilevati in RR	Diego Beraldin	Amministratore	2013-01-12
1.0	Approvazione documento	Stefano Farronato	Responsabile	2012-12-04
0.6	Validazione documento	Marco Schivo	Verificatore	2012-12-04
0.5	Fine stesura documento	Andrea Meneghinello	Amministratore	2012-12-03
0.4	Stesura sezione Milestones e Ticketing, Analisi dei requisiti	Andrea Meneghinello	Amministratore	2012-12-03
0.3	Stesura sezione ambiente documentale	Andrea Meneghinello	Amministratore	2012-12-03
0.2	Stesura sezione ambiente di lavoro	Andrea Meneghinello	Amministratore	2012-12-01
0.1	Stesura scheletro documento, sezione comunicazione	Andrea Meneghinello	Amministratore	2012-12-01



Indice

1	Introduzione	1
1.1	Scopo del prodotto	1
1.2	Scopo del documento	1
1.3	Ambiguità	1
2	Relazioni interpersonali e comunicazione	2
2.1	Comunicazione interna	2
2.2	Comunicazione esterna	2
2.3	Incontri Interni	2
2.4	Incontri Esterni	2
3	Ambiente di lavoro	3
3.1	Repository	3
3.1.1	Registrazione	3
3.1.2	Installazione	3
3.1.3	Struttura	3
3.1.4	Sistema di versionamento	4
3.2	Sistema di tracciamento dei requisiti	4
3.3	Sistema operativo	5
3.4	Diagrammi UML	5
3.5	Diagrammi di Attività	5
4	Ambiente documentale	6
4.1	Software utilizzati	6
4.2	Nomi dei documenti	6
4.3	Impostazioni di base di un documento	6
4.4	Tipi di documento	7
4.5	Ciclo di vita di un documento	8
4.6	Template	9
4.7	Glossario	9
4.8	Norme tipografiche	10
4.8.1	Stili di testo	10
4.8.2	Punteggiatura	10
4.8.3	Composizione del testo	10
4.8.4	Formati ricorrenti	10
4.9	Immagini	11
4.9.1	Inserimento Immagini	11
4.10	Tabelle	11
5	Milestones e Ticketing	12
5.1	Milestones	12
5.1.1	Creazione nuova Milestone	12
5.1.2	Modifica Milestone	13
5.2	Ticketing	13
5.2.1	Creazione nuovo Ticket	13
5.2.2	Modifica Ticket	14
5.2.3	Ciclo di vita di un Ticket	14



6	Analisi dei Requisiti	16
6.1	Norme per i requisiti	16
6.2	Norme per i casi d'uso	17
6.3	Tracciamento	18
6.3.1	Casi d'Uso-Requisiti e viceversa	18
6.3.2	Requisiti-Test	18
7	Progettazione	19
7.1	Stile di progettazione	19
7.2	Design Pattern	19
7.3	Convenzioni sui diagrammi	20
7.4	Classi di verifica	20
7.5	Tracciamento	20
7.6	Struttura della Specifica Tecnica	20
8	Codifica	21
8.1	Convenzioni di codifica	21
8.1.1	Convenzioni sui nomi	21
8.1.2	Struttura delle classi	21
8.1.3	Struttura del codice	21
9	Norme di verifica	22
9.1	Strumenti	22
9.2	Tecniche	23
9.2.1	Analisi statica	23
9.2.2	Analisi dinamica	24
9.3	Verifica dei documenti	24
9.4	Verifica del codice	25

Elenco delle tabelle

1	Revisioni e Milestones	6
---	----------------------------------	---



Elenco delle figure

1	Ciclo di vita di un documento	9
2	Creazione di una nuova Milestone	12
3	Modifica di una Milestone	13
4	Creazione di un nuovo Ticket	14
5	Modifica di un ticket	15
6	Ciclo di vita di un ticket	16

1 Introduzione

1.1 Scopo del prodotto

Con progetto “MyTalk” intendiamo un sistema software di comunicazione tra utenti mediante browser, utilizzando solo componenti standard, senza dover installare plugin o programmi esterni. L'utilizzatore dovrà poter chiamare un altro utente, iniziare la comunicazione sia audio che video, svolgere la chiamata e terminare la chiamata ottenendo delle statistiche sull'attività.

1.2 Scopo del documento

Questo documento viene redatto per definire le norme da adottare da parte di tutti i componenti del gruppo Software Synthesis durante il periodo di svolgimento del capitolato “MyTalk”, commissionato dall'azienda Zucchetti SPA. In particolare si andranno a definire le regole per:

- Relazioni interpersonali e comunicazione
- Definizione dell'ambiente di lavoro
- Redazione documenti
- Convenzioni e norme per l'analisi e la progettazione
- Convenzioni e norme per la verifica di file e documenti

1.3 Ambiguità

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

2 Relazioni interpersonali e comunicazione

2.1 Comunicazione interna

La comunicazione tra i vari componenti del team avverrà principalmente durante gli incontri che si terranno in un luogo fisico comune. Per evitare problemi dovuti alla mancata presenza da parte di un membro, le notifiche dell'attività svolta durante l'incontro verranno scritte in un calendario comune messo a disposizione. Nei casi in cui il lavoro si svolgerà presso la propria abitazione, si potrà utilizzare *Skype* per avviare chat, chiamate e videoconferenze di gruppo. Qualora si evinca la necessità di un ritrovo fisico comune ma vi è l'impossibilità da parte di alcuni membri di recarsi nel luogo prefissato, allora si procederà suddividendo il gruppo in due sottogruppi, ciascuno con un luogo di ritrovo prefissato. I due team di sviluppo quindi potranno comunicare con l'altro team mediante software di comunicazione prestabilito.

2.2 Comunicazione esterna

I contatti verso l'esterno saranno a cura del responsabile di progetto. A tale scopo è stato creato un indirizzo di posta elettronica tramite il quale la persona incaricata tratterà a nome dell'intero gruppo Software Synthesis. L'email sopra descritta è *info@softwaresynthesis.org* e l'inoltro di risposte a tutti gli altri membri sarà sempre a carico del responsabile di progetto che dovrà utilizzare i metodi descritti nella sezione 2.1 "Comunicazione Interna".

2.3 Incontri Interni

Saranno fissate delle riunioni formali decise dal responsabile di progetto che provvederà rendere noto a tutti luogo, data e ora della seduta mediante email personale con almeno tre giorni di anticipo. Nell'email sarà contenuta anche la motivazione per cui si è reso necessario l'incontro. Ai membri è richiesta la conferma di partecipazione tramite risposta sempre via email. Nel caso un componente del team non possa essere presente alla riunione decisa, dovrà specificarne il motivo nell'email di risposta al responsabile. Ogni membro del gruppo ha la possibilità di richiedere un incontro interno: tale domanda dovrà venir indirizzata sempre al responsabile di progetto, il quale la visionerà e pianificherà un eventuale incontro. Ad ogni riunione verrà redatto un verbale dove verranno annotate tutte le decisioni prese e i punti salienti della discussione.

2.4 Incontri Esterni

Sarà il responsabile di progetto a prendere accordi per incontri con il committente o con i proponenti. Ogni membro del gruppo può richiedere un incontro esterno al responsabile, presentando una motivazione. Questa necessità verrà presentata all'intero team e solo nel momento in cui ci saranno tre conferme, con relativa presenza all'incontro, il responsabile provvederà a prendere appuntamento con la parte esterna. In caso contrario la proposta sarà bocciata.

3 Ambiente di lavoro

3.1 Repository

Per un corretto svolgimento del progetto si è reso necessario adottare un luogo dove poter cercare e salvare tutti i documenti da redigere, nonché i file di codifica. Per questo si è deciso di adottare un repository ospitato da *github*. Il motivo principale di questa scelta sta nel fatto che Git, rispetto ad altri sistemi quali Sourceforge ad esempio, è un sistema di controllo di versione distribuito. Rientra infatti nei repository di tipo Distributed Control Version System (DVCS) mentre sourceforge rientra nei repository di tipo Centralized Control Version System. La differenza sta nel fatto che in Git ognuno di noi avrà in locale l'intera copia del repository, quindi tutti i vari commit, rami ecc. mentre con sourceforge in locale si ha solo la situazione dell'ultimo commit e se si deve eseguire operazioni di rollback si ha bisogno della connessione di rete per potersi connettere al server centrale.

3.1.1 Registrazione

Per utilizzare github occorre registrarsi presso:

<https://github.com>.

per far ciò basta inserire un username, un indirizzo email e una password nell'apposito form proposto nella pagina sopra indicata.

3.1.2 Installazione

Per aiutare i componenti del gruppo nella fase di installazione di github, è stata creata appositamente una guida che passo passo spiega come installare e configurare nel modo giusto il software. La guida è reperibile all'indirizzo

<https://github.com/SoftwareSynthesis/SoftwareEngineeringProject/blob/master/Manuals/GuidaGit/MasterDocument.pdf>.

3.1.3 Struttura

- **Progetto:** l'indirizzo web a cui fa riferimento l'intero progetto è:

<https://github.com/SoftwareSynthesis/SoftwareEngineeringProject>

- **Documentazione:** tutti i documenti saranno reperibili all'indirizzo

<https://github.com/SoftwareSynthesis/SoftwareEngineeringProject/Documents>

Ogni documento redatto dovrà essere contenuto in una sotto-cartella della directory "Documents" nominata come il nome del documento stesso: questa conterrà i file \LaTeX . E' inoltre presente una cartella "Revisioni", composta a sua volta da quattro sotto-cartelle ("RR", "RP", "RQ", "RA") che conterranno i documenti formali consegnati alle quattro revisioni del progetto ed una cartella "Pics" che conterrà tutte le immagine utilizzate nei documenti.

- **Codice:** tutti i file sorgente saranno reperibili all'indirizzo

<https://github.com/SoftwareSynthesis/SoftwareEngineeringProject/Code>

Le regole da adottare per utilizzare questa cartella saranno redatte non appena si verificherà la necessità.

- **Others:** per qualsiasi altro file di utilità per il progetto, è stata predisposta una cartella "Others" utilizzabile da tutti i membri.

3.1.4 Sistema di versionamento

Come sistema di controllo di versione è stato adottato *git*. Questa scelta è stata decisa dopo aver individuato diversi pregi, tra i quali:

- velocità;
- design semplice;
- forte supporto allo sviluppo non-lineare (possibilità di migliaia di rami paralleli);
- completamente distribuito;
- capacità di gestire, in modo efficiente (velocità e dimensione dei dati), grandi progetti come il kernel Linux.

3.2 Sistema di tracciamento dei requisiti

Al fine di rendere automatico e sistematico la gestione del tracciamento dei requisiti, è stato creato un sistema che si appoggia al sito aziendale. Il gestionale è stato nominato “SynthesisRequirementManager”. Ogni volta che un membro del gruppo necessita di interagire con tale sistema, si dovrà autenticare alla pagina

<http://www.softwaresynthesis.org/Login.php>

Il sistema di tracciamento offre le seguenti 5 opzioni:

- **Gestione requisiti:** permette l’inserimento, modifica e la cancellazione di un requisito;
- **Gestione fonti:** permette l’inserimento, modifica e la cancellazione di una fonte da intendersi come ad esempio il capitolato d’appalto;
- **Gestione casi d’uso:** permette l’inserimento di un caso d’uso con relativi scenari e requisiti associati;
- **Gestione attori:** permette l’inserimento, modifica e la cancellazione di un attore utilizzabile in seguito per lo sviluppo di un caso d’uso;
- **Gestione classi:** permette l’inserimento e la cancellazione di una classe rilevata nell’attività di progettazione;
- **Gestione componenti:** permette l’inserimento e la cancellazione di una componente rilevata nell’attività di progettazione;
- **Gestione design pattern:** permette l’inserimento e la cancellazione di un design pattern utilizzato nel progetto;
- **Stampa:** è suddivisa in due sezioni, la prima per la parte di analisi, la seconda per la parte di progettazione. Il comportamento delle due stampe è analogo, crea in modo automatico un file di estensione .tex contenente nel primo caso la lista dei requisiti analizzati, i casi d’uso studiati e il tracciamento tra i requisiti-casi d’uso e viceversa, nel secondo caso viene stampato la lista di tracciamenti tra classi-componenti-design pattern (vedi sezione 7.5). Successivamente alla stampa tale file sarà da includere all’intero del documento “*analisi_dei_requisiti.tex*” o “*specifica_tecnica.tex*” a seconda.

Il sistema è basato su tecnologie MySQL con creazione del database sotto engine inno_db. Per sopperire ad alcune mancanze del sistema di tracciamento, lo sviluppatore si potrà appoggiare alla piattaforma phpMyAdmin fornita dal web host, al fine di modificare e gestire alcuni dati del database. In particolare, se ne obbliga l’utilizzo nei seguenti casi:

- modifica di un caso d'uso;
- cancellazione di un caso d'uso.

Quest'obbligo di gestione è dovuto a mancanze temporali nella fase di sviluppo del sistema, nel periodo antecedente alla consegna dei capitolati, e anche in parte per la complessità di implementazione.

Ogni altra modifica attuata tramite phpmyadmin è assolutamente vietata per mansioni che non siano riportate al punto precedente.

3.3 Sistema operativo

L'intero progetto verrà portato avanti attraverso sistemi Unix e Windows, più in particolare Mac OSX 10.8.2, Fedora 17, Windows 7, Windows 8. Questa scelta deriva dal fatto che, essendo il progetto MyTalk un applicativo web, non si sono riscontrate dipendenze alcune da librerie di sistema.

3.4 Diagrammi UML

Per la produzione di diagrammi UML si adotterà il software ConceptDraw. Questo scelta perchè il tool si presenta molto semplice da usare, è molto potente e permette la creazione di moltissimi diagrammi UML adottando lo standard 2.0. La qualità dei diagrammi prodotti inoltre è molto buona.

3.5 Diagrammi di Attività

Come supporto alla pianificazione del progetto si è scelto di utilizzare Project Libre nella versione 1.5.2, software molto leggero e facile da installare. E' un tool opensource e multi piattaforma. Permette la creazione di diagrammi di Gantt in modo molto semplice con possibilità di esportarli in formato jpg.

Per il download e l'installazione si rimanda al seguente link:

<http://sourceforge.net/projects/projectlibre/>.

4 Ambiente documentale

In questa sezione verranno illustrati i vari standard utilizzati dal team per la produzione di documenti durante l'intero progetto.

4.1 Software utilizzati

Tutti i documenti dovranno essere scritti in italiano con \LaTeX , un software free di composizione testuale che comprende una serie di caratteristiche atte alla produzione di documentazione tecnica e scientifica.

Per attuare questa scelta, l'azienda Software Synthesis ha deciso di utilizzare come editor \LaTeX *TexMaker*, software disponibile sia per ambienti Unix che Windows. Si presenta come un software molto flessibile che include al suo interno la correzione ortografica automatica, l'autocompletamento e un visualizzatore di PDF integrato.

La versione da utilizzare è l'ultima disponibile in tale momento ed è la 3.5.2. Per il download seguire il seguente link:

<http://www.xmlmath.net/texmaker/download.html>.

Per l'installazione, una volta scaricato, basta seguire le indicazioni presenti nel file scaricato precedentemente.

4.2 Nomi dei documenti

Il nome dei documenti dovrà rappresentare in modo univoco la natura del file e sarà composto nel seguente modo:

nome_del_file.X.Y.estensione

dove:

- **nome_del_file**: il nome del file non dovrà contenere lettere maiuscole né caratteri speciali (compresi accenti). Nel caso sia composto da più parole, queste vanno separate con il carattere "underscore".
- **Variabile X**: Indica il raggiungimento di uno stato formale rispetto ad una Milestone. Assumerà quindi principalmente 4 valori, dall'1 al 4, attribuiti come segue:

Milestone	X
Revisione dei Requisiti (RR)	1
Revisione dei Requisiti (RP)	2
Revisione dei Requisiti (RQ)	3
Revisione dei Requisiti (RA)	4

Tabella 1: Revisioni e Milestones

- **Variabile Y**: parte dal valore 0 e viene incrementata ogni qualvolta, senza limite superiore, si apportano modifiche sostanziali al documento, come ad esempio la stesura di una nuova sezione o correzione di errori.

4.3 Impostazioni di base di un documento

Ogni documento dovrà attenersi alle seguenti regole per la sua redazione:

- **Intestazione:** composta dalla sezione del documento nella parte sinistra e dal logo nella parte destra.
- **Piè di pagina:** composto dal nome del documento con relativa versione sulla sinistra e dal numero di pagina sulla destra.

La prima pagina di ogni documento dovrà contenere come prima cosa il logo dell'azienda Software Synthesis, il nome del documento e una tabella che riporta le seguenti informazioni sul file:

- nome del file;
- versione;
- data creazione;
- data ultima modifica;
- stato;
- uso;
- redattori;
- approvato da;
- verificatori.

Nella seconda pagina invece dovrà essere presente la tabella delle modifiche apportate. Queste dovranno essere inserite in ordine cronologico dalla più recente alla creazione, specificando:

- versione del documento;
- descrizione dell'intervento;
- redattore;
- data modifica.

Seguirà su una nuova pagina l'indice dell'intero documento e in caso di presenza di tabelle e immagini sarà presente anche l'indice delle tabelle e l'indice delle immagini a seguire. La presenza di questi due indici va specificata settando a "true" due flag presenti nel documento "Template" e ognuno di essi dovrà comparire in una nuova pagina.

Il documento dovrà essere suddiviso in sezioni con relative sottosezioni se presenti. Ogni sezione deve cominciare in una nuova pagina.

4.4 Tipi di documento

- **Informali:** verranno "etichettati" come informali tutti i documenti in fase di redazione, oppure completati ma non ancora approvati dal responsabile di progetto. Vanno utilizzati solo internamente e quindi non verranno distribuiti all'esterno. Sono reperibili nella cartella "Documents" del repository e sono organizzati secondo quanto previsto nella sezione 3.1.3 "Struttura repository".
- **Formali:** tutti i documenti approvati dal responsabile di progetto sono da considerarsi formali e quindi pronti alla distribuzione verso l'esterno. Sono reperibili nella cartella "Revisioni" del repository e sono organizzati secondo quanto previsto nella sezione 3.1.3 "Struttura repository".

- **Verbali:** documenti redatti dal responsabile di progetto dopo un avvenuto incontro esterno. Servono come promemoria dell'incontro avvenuto e delle tematiche trattate. Dovranno essere denominati

verbale_incontro_{dataincontro}

dove con {dataincontro} intendiamo la data dell'incontro espressa nel formalismo descritto nella sezione 4.8.4 "Formati Ricorrenti".

Ogni verbale dovrà contenere le seguenti informazioni riportate come seguono:

- **Data:** indica la data dell'incontro
- **Ora:** indica l'ora dell'incontro espressa in "HH:MM" dove HH sarà compresa tra 00 e 23 ed indicherà le ore mentre MM sarà compreso tra 00 e 59 ed indicherà i minuti.
- **Luogo:** indica il luogo dell'incontro e dovrà essere espresso nel seguente formalismo:
provincia, città, via, numero civico
- **Durata:** indica la durata dell'incontro espressa in minuti
- **Partecipanti:** indica tutti i partecipanti presenti alla riunione. Deve essere un elenco di nomi e cognomi suddivisi per membri interni ed esterni.
- **Oggetto:** indica l'argomento trattato all'incontro. Seguono i punti salienti e le decisioni prese.

Il verbale dovrà essere approvato da tutti i componenti firmandolo entro e non oltre il successivo incontro. L'approvazione è ufficializzata automaticamente nel momento in cui sono poste tutte le firme dei presenti. In via del tutto eccezionale, nel qual caso un componente non possa approvare il documento nei termini prestabiliti, è compito del responsabile assicurarsi che ciò avvenga il prima possibile. Nel caso in cui un membro del gruppo non abbia potuto partecipare alla riunione, esso dovrà comunque prendere visione del verbale e dovrà firmarlo per accertare la presa visione dello stesso.

4.5 Ciclo di vita di un documento

Come mostrato in figura 1, un documento seguirà i seguenti passi:

1. Il redattore di un nuovo documento dovrà redigere in locale la prima impostazione del documento e solo allora potrà caricarlo nel repository secondo le regole indicate nella sezione 3.1 "Repository".
2. Ad ogni sostanziale modifica verrà incrementato il contatore Y (vedi sezione 4.2 "Nomi di documenti") e seguirà un commit nel repository.
3. Quando il redattore finisce la stesura o revisione del documento, questo verrà proposto al verificatore il quale avrà il compito di controllare che siano state seguite le norme riguardanti la stesura di documenti (vedi sezione 9.3 "Verifica dei documenti"). Il verificatore potrà dare due esiti, positivo o negativo.
4. In caso di esito negativo, il documento viene riproposto al redattore evidenziandone gli errori riscontrati. Se invece l'esito è positivo, il documento è stato redatto secondo le norme e non presenta errori, passerà quindi al responsabile di progetto che provvederà all'approvazione formale.
5. Anche il responsabile di progetto può dare due giudizi, idoneo o non idoneo. Nel caso di idoneità, il documento è da considerarsi formale e può essere inserito nella cartella corrispondente alla Milestone presente nella directory "Revisioni".
6. In caso di non idoneità, il documento verrà rimandato al redattore notificando il motivo di tale decisione e si ripartirà dal punto 3.

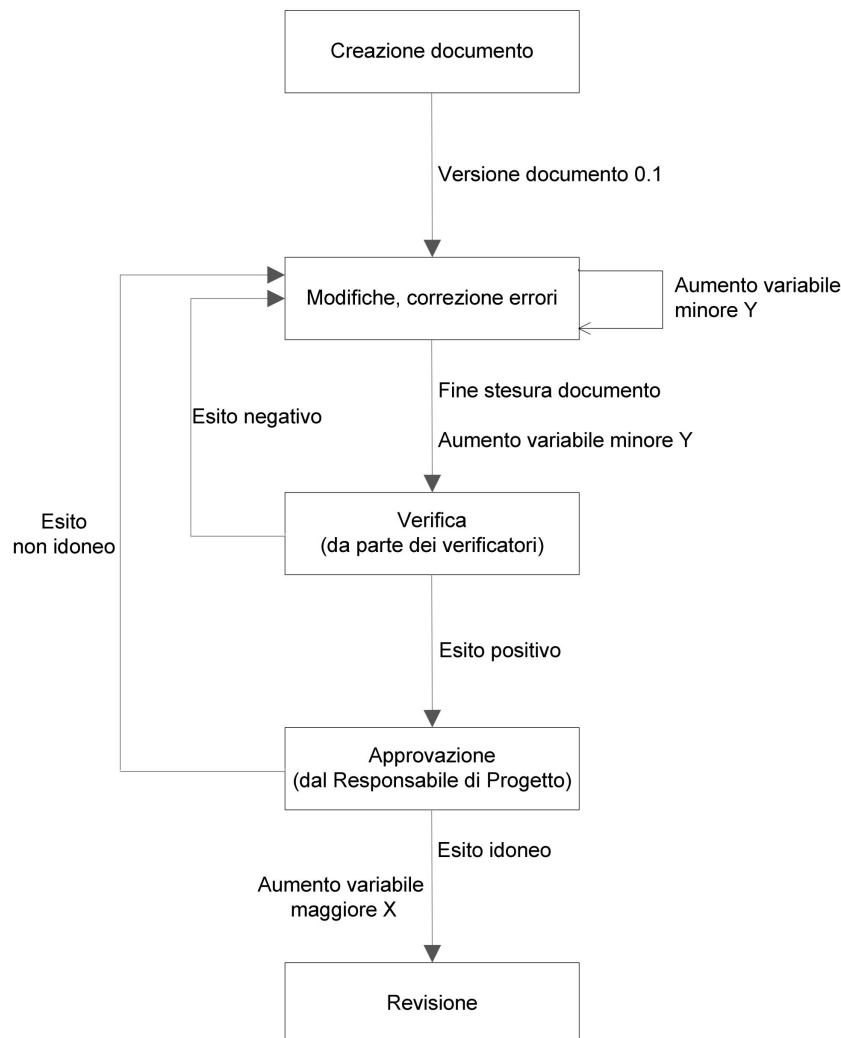


Figura 1: Ciclo di vita di un documento

4.6 Template

Ogni documento dovrà essere generato includendo il template $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ presente nella cartella “Documents”. Questo file è stato redatto prima dell’inizio di stesura di ogni altro documento sotto comune accordo. La modifica perciò di tale template deve essere richiesta all’amministratore di progetto che analizzerà l’esigenza e darà una risposta positiva o negativa. In caso positivo, si procederà all’attuazione delle modifiche e verrà informato l’intero team al fine che ogni membro modifichi il documento su cui sta lavorando.

4.7 Glossario

Il glossario è un documento nel quale vengono riportate tutte le parole difficilmente comprensibili o dal significato ambiguo presenti nei documenti. E’ un documento un po’ particolare in quanto non rispetta le regole stilistiche degli altri documenti. Non sarà presente l’indice dei contenuti, ma solo una pagina iniziale contenente le informazioni del documento, la tabella delle modifiche, una breve introduzione e a seguire le voci ordinate alfabeticamente.

4.8 Norme tipografiche

Questa sezione descrive le norme da utilizzare per l'utilizzo dell'ortografia, tipografia e l'assunzione di uno stile uniforme nel redigere i documenti.

4.8.1 Stili di testo

- **Grassetto:** usato per evidenziare passaggi importanti. Si utilizza anche su elementi immediatamente seguenti agli elenchi per evidenziare l'oggetto trattato nel paragrafo.
- **Corsivo:** usato per dare enfasi a parole o frasi su cui prestare attenzione.
- **Sottolineato:** le parole sottolineate sono riportate nel glossario. Ogni occorrenza di tali parole va sottolineata.
- **Maiuscolo:** le parole in maiuscolo vanno usate solo per acronimi.

4.8.2 Punteggiatura

Qualsiasi segno di punteggiatura non deve mai seguire uno spazio ma deve precederne uno. Dopo un punto fermo deve esserci uno spazio seguito da una lettera maiuscola.

Le parentesi vanno usate per raggruppare un periodo, non devono aprirsi con un carattere di spaziatura e non devono chiudersi con un carattere di punteggiatura o di spaziatura.

4.8.3 Composizione del testo

Gli elenchi, sia puntati che numerati, vanno usati per elencare una serie di elementi e solitamente hanno una breve lunghezza con la quale descrivono l'elemento in questione. Per questo, ogni punto elenco deve finire con un punto e virgola (;) tranne l'ultimo elemento che finirà con un punto (.), mentre nel caso di punti elenco più discorsivi la frase finirà con un punto ogni volta. Per quanto concerne le note a piè di pagina invece, ogni nota dovrà cominciare con la prima lettera della prima parola maiuscola, senza spaziatura davanti e finire con un punto.

4.8.4 Formati ricorrenti

- **Path:** per scrivere indirizzi web e url si adotterà il comando prestabilito da L^AT_EX `\url`.
- **Date:** per scrivere una data si dovrà utilizzare il formato AAAA-MM-GG (standard ISO 8601) dove:
 - AAAA: sta ad indicare l'anno;
 - MM: sta ad indicare il mese;
 - GG: sta ad indicare il giorno.
- **Sigle:** di seguito sono elencate una serie di sigle, da usare principalmente all'interno di diagrammi o tabelle per risparmiare spazio.
 - AR: documento "Analisi dei Requisiti"
 - PdP: documento "Piano di Progetto"
 - PdQ: documento "Piano di Qualifica"
 - NdP: documento "Norme di Progetto"

4.9 Immagini

Tutte le immagini utilizzate nei documenti devono avere estensione .png. Nel caso l'immagine abbia una grande dimensione, va compressa con il formato .jpg. Devono risiedere tutte nella cartella “pics” (<https://github.com/SoftwareSynthesis/SoftwareEngineeringProject/Documents/pics>).

Nel caso di immagini riguardanti diagrammi UML, si deve rispettare la seguente regola per il nome file:

$$tipologia\{x1-x2\ldots xn\}$$

dove “tipologia” dovrà essere una di queste sigle:

- **UC**: diagrammi casi d'uso;
- **DC**: diagrammi delle classi;
- **DO**: diagrammi degli oggetti;
- **DS**: diagrammi di sequenza;
- **DA**: diagrammi di attività;
- **DP**: diagrammi di package.

e “x1-x2-...-xn” rappresenteranno la numerazione gerarchica dei diagrammi, dove il numero più a sinistra rappresenta il padre e quelli più a destra i figli.

Nel caso di immagine generiche, il loro nome dovrà rispecchiare il contenuto di tale immagine.

4.9.1 Inserimento Immagini

L'inserimento di immagini all'interno del documento verrà fatto tramite l'utilizzo del comando `\begin{figure}` e `\includegraphics[parametri]{riferimento}` dove:

- **parametri**: indica una serie di parametri da utilizzare per formattare al meglio l'immagine, come una scala di ridimensionamento (`[scale=0.8]` indica un ridimensionamento del 20%), l'altezza e larghezza (`[width=2cm, heigh=4cm]` indica un'immagine larga 2 cm e alta 4 cm) oppure parametri del tipo `[width=\textwidth]` per indicare che la figura corrisponda esattamente alla larghezza del testo;
- **riferimento**: va messo il nome dell'immagine presente nella cartella “pics” (vedi sezione 3.1.3) che si vuole inserire.

Prima della chiusura (`\end{figure}`) vanno inseriti anche i comandi `\caption{didascalia}` e `\label{riferimento}` per permettere l'indicizzazione e l'identificazione delle immagini tramite un codice.

4.10 Tabelle

L'inserimento di tabelle all'interno del documento verrà fatto tramite l'utilizzo del package `tabularx` che permette la creazione di colonne a larghezza dinamica. Vanno inseriti prima della chiusura della tabella (quindi prima del comando `\endtabularx`) i comandi `\caption{didascalia}` e `\label{riferimento}` per permettere l'indicizzazione e l'identificazione delle tabelle tramite un codice.

5 Milestones e Ticketing

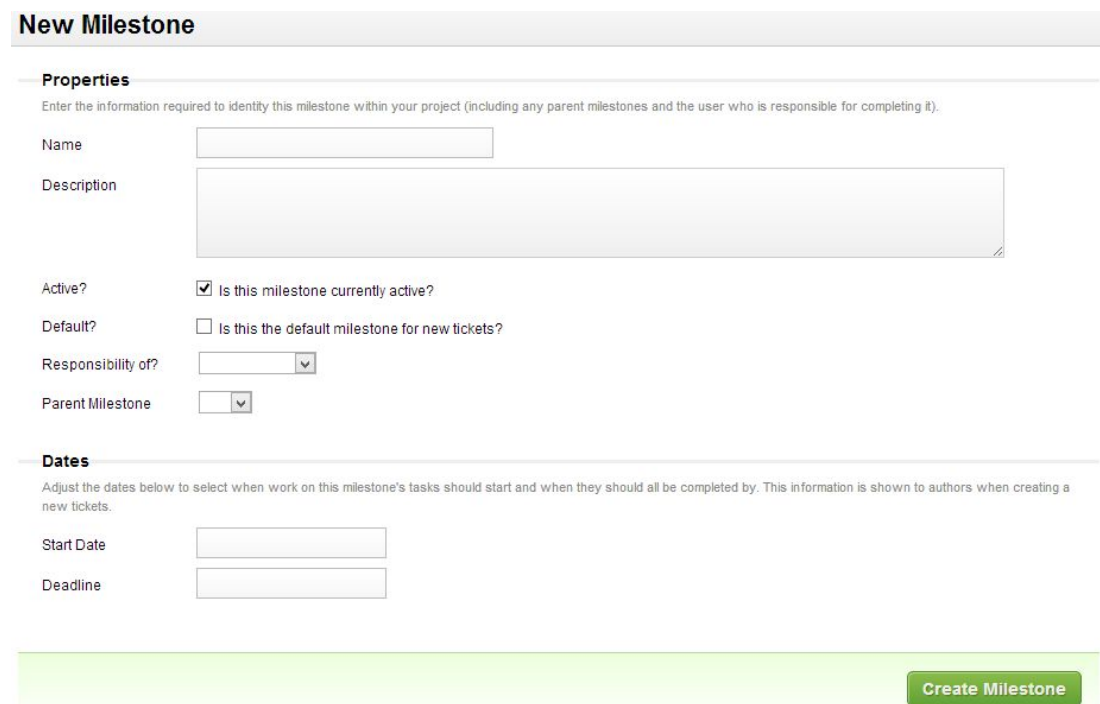
Al fine di aiutare lo svolgimento delle operazioni di assegnazione dei lavori e revisione durante l'intero ciclo di vita del progetto, l'azienda Software Synthesis userà il servizio *Codebase*. Questo permette la creazione di Milestones e tickets, strumenti molto utili per tracciare il lavoro del team e anche per delineare una roadmap degli obiettivi a breve e lungo termine che l'azienda si prefigge di raggiungere.

5.1 Milestones

Come prima cosa, il responsabile di progetto dovrà inserire tutte le Milestones previste dal progetto, ossia le quattro revisioni.

5.1.1 Creazione nuova Milestone

Per inserire una nuova Milestone ci si dovrà posizionare nella pagina web dedicata alle Milestone e cliccare su "New Milestone". Il form va completato nel seguente modo:



New Milestone

Properties

Enter the information required to identify this milestone within your project (including any parent milestones and the user who is responsible for completing it).

Name

Description

Active? ☒ Is this milestone currently active?

Default? ☐ Is this the default milestone for new tickets?

Responsibility of?

Parent Milestone

Dates

Adjust the dates below to select when work on this milestone's tasks should start and when they should all be completed by. This information is shown to authors when creating a new tickets.

Start Date

Deadline

Create Milestone

Figura 2: Creazione di una nuova Milestone

- **Name:** inserire il nome della Milestone;
- **Description:** inserire una breve descrizione della Milestone;
- **Active?:** flag da spuntare se si vuole che la Milestone creata sia quella attiva;
- **Default?:** flag da spuntare se si vuole che i futuri tickets creati siano riferiti a questa Milestone;
- **Responsibility of?:** selezionare la persona responsabile;

- **Parent Milestone**: selezionare se la Milestone che si va a creare è figlia di qualche altra Milestone;
- **Start Date**: inserire la data da cui far iniziare la Milestone;
- **Deadline**: inserire la data di chiusura delle Milestone.

Nel caso non si conosca qualche informazione, si può lasciare il relativo campo bianco e modificarlo successivamente (vedi paragrafo seguente) quando sarà resa pubblica l'informazione.

5.1.2 Modifica Milestone

Per modificare una Milestone, ci si deve posizionare nella scheda "Edit Milestone". Per raggiungerla, basta entrare nella Milestone che si vuole modificare e cliccare sul bottone "Edit Milestone" situato nella parte alta dello schermo a destra.



Figura 3: Modifica di una Milestone

Per la compilazione dei campi, valgono le stesse regole riportate nella sezione 5.1.1 "Creazione nuova Milestone".

Da notare che si può eliminare anche una Milestone da tale scheda, basta cliccare sul bottone "Delete Milestone" presente nella parte destra della pagina.

5.2 Ticketing

Un ticket rappresenta un compito da svolgere.

5.2.1 Creazione nuovo Ticket

Per inserire un nuovo ticket ci si dovrà posizionare nella pagina web dedicata ai Tickets e cliccare su "New Ticket". Il form va completato compilando obbligatoriamente i seguenti campi:

- **Summary**: inserire il nome del compito da assegnare;
- **Status**: da settare su "New";
- **Priority**: selezionare la priorità effettiva del compito;
- **Ticket Type**: da settare su "Task";
- **Assigned User**: selezionare la persona che dovrà svolgere il compito;
- **Milestone**: selezionare la Milestone corrispondente del compito.

Tutte le altre informazioni possono essere completate se si ritiene opportuno dare maggiori dettagli.

New Ticket

Summary — enter a brief description of this ticket (up to 250 characters)

Additional information — you may enter optional information for this ticket in the field below (this is optional)

Rich Text Editor: B I [link icon] [quote icon] [code icon] [table icon] [list icon] [H2 icon] [undo icon] [redo icon] [link icon] [unlink icon] [search icon]

Form Fields:

- Ticket Properties: STATUS (New), ASSIGNED USER (dropdown)
- Milestone & Dates: PRIORITY (Normal), TAGS (text area)
- Attachments: CATEGORY (General)
- Watchers: TICKET TYPE (Feature)

This ticket will be added to the **RR** milestone. To change this select the Milestone and Dates tab above.

Create Ticket

Figura 4: Creazione di un nuovo Ticket

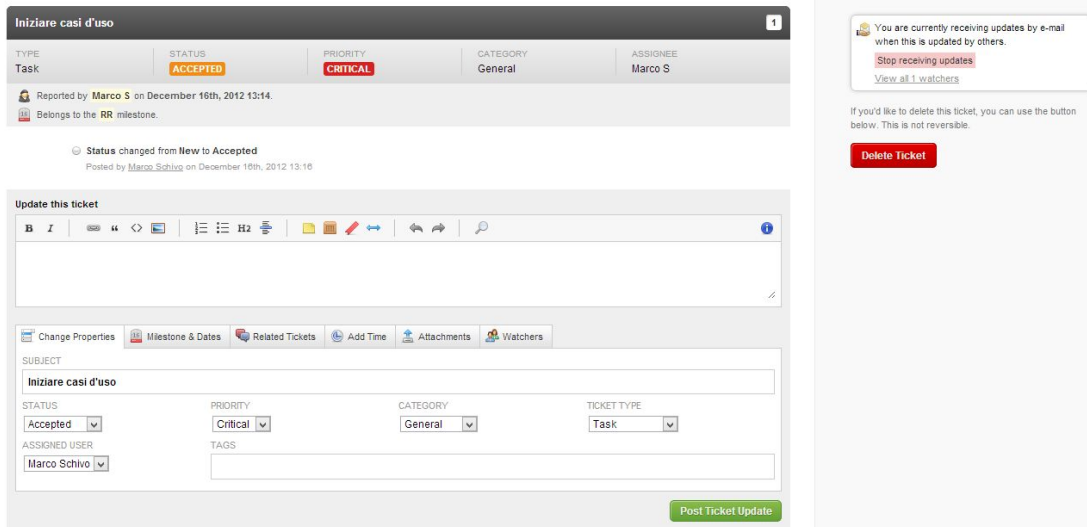
5.2.2 Modifica Ticket

Durante l'intero ciclo di vita di un ticket, questo viene modificato molte volte come avviene ad esempio quando dobbiamo modificare lo stato del ticket. Per far ciò, basta entrare nel ticket che si vuole modificare e appare subito la scheda di modifica.

5.2.3 Ciclo di vita di un Ticket

Come riportato in figura 6, il ciclo di vita di un ticket si articolerà nel seguente modo:

1. **Creazione:** come prima cosa, il ticket va creato come descritto nella sezione 5.2.1 “Creazione nuovo Ticket”.
2. **Esecuzione:** ogni membro dovrà prendere atto dei tickets a lui pervenuti e una volta visionati impostare lo stato ad “Accepted” o ad “Invalid”. Una volta accettati, lo stato va modificato in “In Progress” quando si inizia a lavorarci e “Completed” quando si è portato a termine il compito assegnato.
3. **Verifica:** una volta che il compito è stato completato, si procede nella creazione di un nuovo ticket di tipo “Verification” il quale verrà assegnato ad un verificatore. Il nuovo ticket va compilato nel seguente modo:
 - **Summary:** dovrà contenere il nome del documento o file da verificare;
 - **Status:** da settare su “New”;
 - **Priority:** selezionare la priorità effettiva del compito;
 - **Ticket Type:** da settare su “Verification”;
 - **Assigned User:** selezionare il verificatore che dovrà svolgere il compito;
 - **Milestone:** selezionare la Milestone entro cui il documento dovrà essere verificato.
4. **Esecuzione verifica:** per ogni errore (esclusi quelli grammaticali) dovrà essere creato un nuovo ticket nel seguente modo:



Iniziare casi d'uso 1

TYPE	STATUS	PRIORITY	CATEGORY	ASSIGNEE
Task	ACCEPTED	CRITICAL	General	Marco S

Reported by **Marco S** on December 16th, 2012 13:14.
Belongs to the **RR** milestone.

Status changed from **New** to **Accepted**
Posted by **Marco Schivo** on December 16th, 2012 13:16

Update this ticket

SUBJECT
Iniziare casi d'uso

STATUS: **Accepted** | PRIORITY: **Critical** | CATEGORY: **General** | TICKET TYPE: **Task**

ASSIGNED USER: **Marco Schivo** | TAGS:

Post Ticket Update

You are currently receiving updates by e-mail when this is updated by others.
Stop receiving updates
[View all 1 watchers](#)

If you'd like to delete this ticket, you can use the button below. This is not reversible.
Delete Ticket

Figura 5: Modifica di un ticket

- **Summary:** dovrà contenere il nome del documento o file da modificare;
- **Additional information:** dovrà contenere la descrizione dell'errore presente nel documento;
- **Status:** da settare su "New";
- **Priority:** selezionare la priorità effettiva del compito;
- **Ticket Type:** da settare su "Bug";
- **Assigned User:** selezionare il Responsabile del progetto;
- **Milestone:** selezionare la Milestone entro cui il documento dovrà essere verificato.

Sarà compito del Responsabile approvare o invalidare il ticket. In caso di non approvazione, il Responsabile dovrà decidere a chi far eseguire la modifica. Il membro selezionato dovrà quindi risolvere l'errore. Una volta completata la modifica il Responsabile dovrà riapplicare la procedura a partire dal punto 3.

Quando tutti i tickets di una Milestone avranno stato "Completed", significa che tutte le attività sono state svolte e il Responsabile potrà procedere alla chiusura della Milestone.

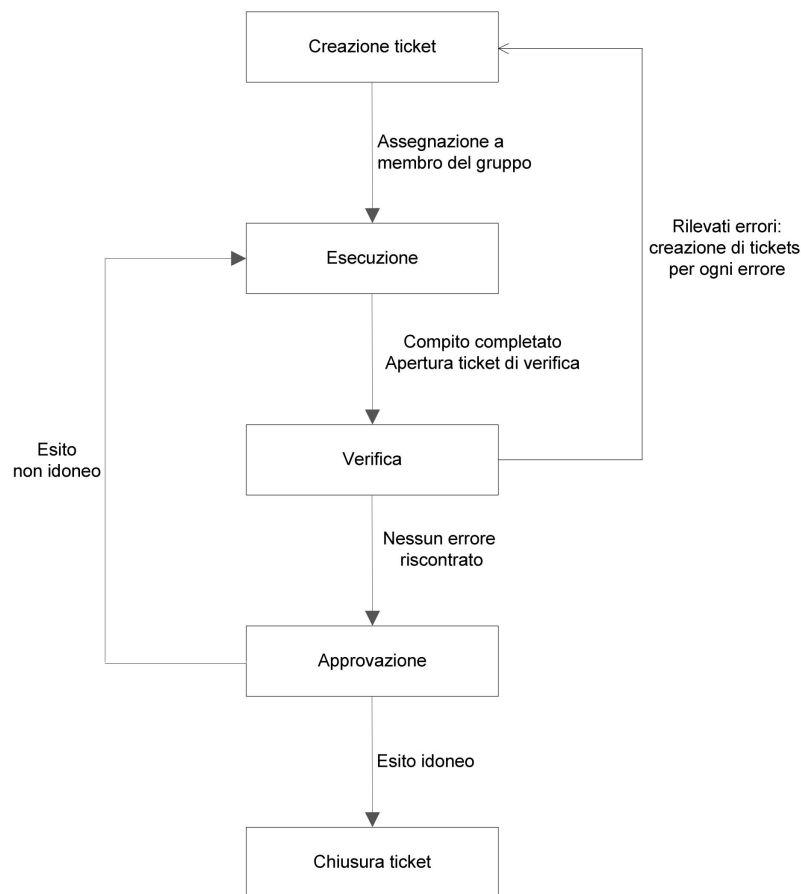


Figura 6: Ciclo di vita di un ticket

6 Analisi dei Requisiti

L'analisi dei requisiti dovrà essere redatta dagli analisti. Il documento si presenterà suddiviso principalmente in tre parti: una prima parte riguardante i requisiti, una seconda parte riguardante i casi d'uso e la parte finale riguardante il tracciamento dei due elementi appena elencati.

6.1 Norme per i requisiti

La parte riguardante i requisiti dovrà elencare ordinatamente tutti i requisiti evinti dal capitolato d'appalto o dagli incontri con il proponente. Per far ciò, *ogni requisito avrà nome univoco* e dovrà essere indicato nel formalismo:

$$R\{\text{Ambito}\}\{\text{Tipo}\}\{\text{Priorità}\}\{\text{Gerarchia}\}$$

dove:

- **Ambito:** indicherà l'ambito del requisito e potrà assumere i seguenti valori
 - U per indicare “Utente”;
 - S per indicare “di Sistema”.

- **Tipo:** indicherà il tipo del requisito e potrà assumere i seguenti valori:
 - F per indicare “Funzionale”, ossia requisiti che definiscono quali funzioni devono essere comprese. Tali requisiti si verificano per mezzo di test, dimostrazioni formali e revisioni.
 - Q per indicare “di Qualità”, ossia definisce i requisiti sui fattori di rilievo della qualità, usabilità, sicurezza, scalabilità. Per questi deve essere adoperata una verifica ad hoc.
 - P per indicare “Prestazionale”, ossia requisiti che stabiliscono vincoli su tempistiche e spazio, in relazione alla funzione da svolgere. Vengono verificati tramite misurazioni.
 - D per indicare “Dichiarativo”, ossia requisiti con vincoli spesso ambientali e culturali. Si verificano per mezzo di revisione.
- **Priorità:** indicherà la priorità del requisito e potrà assumere i seguenti valori
 - O per indicare “Obbligatorio”, ossia necessario per la realizzazione del progetto;
 - D per indicare “Desiderabile”, ossia non obbligatorio ma, se implementato, aumenta il valore del progetto;
 - F per indicare “Facoltativo”, ossia un requisito da implementare se avanzano risorse al termine del progetto.
- **Gerarchia:** indicherà il grado di parentela dei requisiti e andrà specificato nella sintassi

$$x.y.z$$

dove il numero più a sinistra rappresenta il padre e quelli più a destra i figli.

Ogni requisito dovrà essere presente nel documento attraverso una “scheda”, dove oltre i dettagli ricavabili dal nome, dovranno essere specificati anche una breve descrizione, la fonte (ossia la loro origine) e il motivo.

Esempio

“RUFO2.3.1” indica un requisito utente, funzionale obbligatorio con padre RUFO2.3.0 che a sua volta è figlio di RUFO2.0.0.

6.2 Norme per i casi d’uso

Il nome di ogni caso d’uso dovrà essere anch’esso, come per i requisiti, univoco e nella forma:

$$UC\{n1.n2...n3\}$$

dove con “n1.n2...n3” intendiamo la gerarchia, ossia il grado di parentela. Come per i requisiti, il numero più a sinistra rappresenta il padre e quelli più a destra i suoi figli.

Ogni caso d’uso dovrà avere i seguenti dettagli:

- Nome;
- Grafico UML;
- Scopo;
- Pre;
- Post;
- Attori;

- Scenario Principale;
- Scenari Secondari.

Esempio

“UC1.2.3” indica un caso d’uso con padre UC1.2.0 che a sua volta è figlio di UC1.0.0.

6.3 Tracciamento

6.3.1 Casi d’Uso-Requisiti e viceversa

Durante l’intero ciclo di vita del progetto, sono previste misure per il tracciamento dei casi d’uso con i relativi requisiti. Questo avviene grazie all’utilizzo del sistema di tracciamento dei requisiti “SynthesisRequirementManager” descritto nella sezione 3.2 “Sistema di tracciamento dei requisiti”. Tutta la gestione è automatica e in output sarà visualizzata una tabella con tre colonne: Codice UC, Nome UC, Requisito. Questa soluzione facilita notevolmente la gestione di grandi quantità di dati e un notevole risparmio di tempo. Avviene anche la stampa inversa, ossia Requisiti-Casi d’Uso.

6.3.2 Requisiti-Test

E’ prevista anche una tabella che traccia i requisiti e relativi test da effettuare per verificare il soddisfacimento di tutti i requisiti. Tale tabella avrà tre colonne, la prima riporta il codice del requisito, la seconda il codice del test e nella terza colonna una breve descrizione del test da effettuare.

7 Progettazione

In questa sezione verranno descritte le norme che i progettisti dovranno seguire durante la loro attività.

7.1 Stile di progettazione

Al fine di semplificare la futura fase di verifica, nonché la leggibilità degli schemi e diminuire il livello di rischio dovuto ad una progettazione errata, si dovranno rispettare le seguenti regole e linee guida:

- **Information hiding:** si dovrà seguire il principio di *information hiding*, ossia nascondere dettagli organizzativi tramite attributi e metodi privati e/o protetti in modo tale da ridurre la complessità del contratto della classe. Ciò aumenta il grado di protezione e abbassa il rischio di errori logici.
- **Decomposizione:** dovrà essere eseguita una *decomposizione modulare* al fine di identificare fin da subito i componenti terminali che non richiedono ulteriore scomposizione, in modo tale da abbattere tempi e costi già dalle prime fasi.
- **Design patterns:** individuare i *design patterns* da poter usare e applicarli correttamente.
- **Livello di dettaglio:** raggiungere un livello di dettaglio in modo da permettere la parallelizzazione della codifica e non lasciare decisioni ai programmatori.
- **Annidamento di chiamate:** non dovranno essere presenti nell'applicativo chiamate di metodi annidate con profondità maggiore di dieci. Se è necessaria una modifica di tale limite il progettista dovrà proporlo, con relativa giustificazione, al responsabile di progetto che valuterà la richiesta.
- **Costrutti condizionali:** non dovranno essere presenti nell'applicativo flussi condizionali con profondità maggiore di cinque. Se è necessaria una modifica di tale limite il progettista dovrà proporlo, con relativa giustificazione, al responsabile di progetto che valuterà la richiesta.
- **Ricorsione:** la tecnica di ricorsione dovrà essere evitata il più possibile. Nel caso venga utilizzata, il progettista dovrà giustificarne la scelta e fornire inoltre la correttezza per dimostrarne la terminazione.
- **Parametri:** il numero di parametri passati ad una chiamata di metodo non dovrà superare il limite di sette.
- **Concorrenza:** se vengono utilizzati flussi di esecuzione concorrenti, dovrà essere fornito anche il relativo diagramma di flusso.

7.2 Design Pattern

Per ogni design pattern utilizzato, si dovrà dare una breve descrizione indicando:

- la struttura generale;
- il diagramma generale;
- i vantaggi che ne comportano l'uso;
- elenco dei componenti dove è stato utilizzato il design pattern.

7.3 Convenzioni sui diagrammi

Si dovrà utilizzare il linguaggio UML per definire:

- **Diagrammi dei package:** dovranno essere presenti per la parte server implementata con Java. Servono per definire i package, ossia un raggruppamento di classi.
- **Diagrammi delle classi:** dovranno essere presenti per la parte server implementata con Java. Verranno usati per descrivere tipi di entità, con le loro caratteristiche e le eventuali relazioni.
- **Diagrammi di attività:** dovranno essere presenti per mostrare generici flussi di attività che un utente può compiere all'interno dell'applicativo.

7.4 Classi di verifica

Si dovranno sviluppare, quando possibile, delle classi fittizie da utilizzare in fase di verifica e come prototipo.

7.5 Tracciamento

Per il tracciamento generale di componenti, requisiti, classi, si utilizzeranno sei tabelle complessive con due colonne ognuna. Di seguito la lista di tali tabelle che devono comparire nel documento "specifico tecnica":

- requisiti-componenti;
- componenti-requisiti;
- componenti-design pattern
- design pattern-componenti;
- componenti-classi
- classi-componenti.

7.6 Struttura della Specifica Tecnica

Il documento relativo alla Specifica Tecnica sarà strutturato nel seguente modo:

- una breve introduzione che spiega lo scopo del documento, i formalismi adottati e i riferimenti alle norme;
- un breve elenco degli strumenti utilizzati;
- breve trattazione dei design pattern utilizzati seguendo le norme indicate nel paragrafo 7.2;
- descrizione dell'architettura generale evidenziando i componenti rilevati in fase di progettazione indicando il tipo, la funzione e l'obiettivo;
- descrizione dell'architettura di dettaglio (quando disponibile);
- diagrammi UML;
- tracciamenti (come descritto in sezione 7.5).

8 Codifica

Questa sezione andrà ampliata nel momento in cui si avvierà la vera attività di codifica. Vengono intanto fornite delle convenzioni di codifica generale.

8.1 Convenzioni di codifica

Per rendere il codice più leggibile ed ordinato, per quanto riguarda l'uso del linguaggio Java, si dovranno rispettare le *Java Code Convention* reperibili all'indirizzo <http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>. Tali convenzioni possono essere variate, ma tale modifica deve essere richiesta al responsabile di progetto che valuterà la vera necessità di tale modifica.

8.1.1 Convenzioni sui nomi

- **Classi e Interfacce:** il loro nome è un sostantivo e deve avere la prima lettera maiuscola.
- **Metodi:** devono avere il nome di un verbo e la prima lettera minuscola. Nel caso il nome sia composto da più parole, la prima lettera di ogni parola sarà maiuscola.
- **Variabili:** devono avere un nome che fa capire subito il significato. Nel caso il nome sia composto da più parole, la prima lettera di ogni parola sarà maiuscola.
- **Costanti:** il nome va scritto tutto in maiuscolo. Nel caso il nome sia composto da più parole, vanno separate da un underscore.

8.1.2 Struttura delle classi

Le classi dovranno essere composte riportando nel seguente ordine tali elementi:

1. variabili statiche e di istanza;
2. costruttori;
3. metodi.

8.1.3 Struttura del codice

Per la stesura di codice, si seguiranno le seguenti regole:

1. le dichiarazioni di variabili andranno riportate tutte all'inizio e si dovrà usare una riga per ogni variabile;
2. ogni blocco di codice (tipo definizione di un metodo) va separato con una linea bianca vuota;
3. le parentesi graffe che delineano un blocco di codice devono iniziare nella stessa riga della parola chiave a cui appartengono e finire in una nuova riga allineata con la parola d'inizio;
4. parole chiave e parentesi (ad esempio parentesi dovute a costrutti condizionali o ciclici) non devono essere separate da spazi.

9 Norme di verifica

La verifica è un'attività molto importante svolta dal verificatore. Lo scopo è di accertare che i prodotti ottenuti, qualunque sia il loro tipo, siano conformi le norme e convenzioni riportate su questo documento. Oltre a questo, il verificatore dovrà anche assicurare sotto la propria responsabilità che tali prodotti rispettino i requisiti di qualità descritti nel documento Piano di Qualifica ([piano_di_qualifica.2.0.pdf](#)).

9.1 Strumenti

Si riporta a continuazione un elenco dei principali strumenti per la verifica di cui il team si dovrà avvalere nell'arco dello sviluppo del progetto.

- **SynthesisRequirementManager**, il sistema di gestione dei requisiti (citato nel paragrafo 3.2 realizzato da Software Synthesis, avente lo scopo di rendere quanto più agevole gestire il tracciamento dei requisiti a tutti i livelli (requisiti-UC, requisiti-CI, ecc.) e, da un punto di vista prettamente qualitativo, assicurare la necessità e la sufficienza dei casi d'uso e delle componenti software;
- **lacheck** (≥ 1.26) per assicurare la correttezza sintattica e l'adozione delle *best practices* per i sorgenti \LaTeX nonché rilevare in modo semi-automatico le sviste non segnalate dal compilatore in quanto pur corrispondendo a codice ben formato nascondono errori tipografici sottostanti per es. bilanciamento delle virgolette, spaziature scorrette per frasi terminate da un acronimo prima di un punto, mancato utilizzo di spazi inescabibili, ecc. (www.ctan.org/pkg/lacheck);
- **hunspell** (≥ 1.3) come correttore ortografico e analizzatore morfologico in fase di redazione della documentazione, scelto per la sua portabilità ma anche perché alle sue librerie si appoggia l'applicativo **TexMaker** utilizzato per la stesura dei documenti \LaTeX (<http://hunspell.sourceforge.net>);
- le utilità che costituiscono la suite di **QA** del W3C al fine di verificare l'aderenza agli standard delle pagine web generate, in particolar modo gli strumenti online:
 - **Unicorn** in qualità di strumento di validazione unificato (<http://validator.w3.org/unicorn>);
 - **CSS Valitatom Service** come utilità di validazione per i fogli di stile a cascata (<http://jigsaw.w3.org/css-validator>);
- gli strumenti per sviluppatori integrati in **Google Chrome** (<https://developers.google.com/chrome-developer-tools>) e, in particolare:
 - la sezione Sources che rappresenta un'interfaccia al debugger per il motore JavaScript V8 e consente di impostare breakpoint (assoluti o condizionali) per seguire l'esecuzione del codice passo passo (con le consuete funzioni di "step over", "step into" e "step out") nonché arrestare temporaneamente l'esecuzione al sollevamento di un'eccezione (o di un'eccezione non controllata);
 - la sezione Timeline che permette di quantificare i tempi necessari al caricamento e all'esecuzione degli script, nonché di tracciare l'utilizzo della memoria e forzare l'invocazione del *garbage collector*;
 - gli strumenti di benchmark accessibili dalla sezione Profiles, vale a dire il profiler della CPU, che permette di ricostruire l'albero delle chiamate di funzione e la percentuale di utilizzo della CPU per ciascuna funzione, e il profiler dello heap, mediante il quale è possibile ispezionare il contenuto dello heap e salvarne delle rappresentazioni istantanee;

- **FirebugLite**, un'estensione per Chrome che consente di ispezionare gli elementi HTML e la struttura del DOM nonché di modificare in tempo reale i valori delle proprietà dei CSS (<https://getfirebug.com/firebuglite>);
- **SpeedTracer** uno strumento che consente di identificare i problemi di prestazioni nelle applicazioni web visualizzando una serie di metriche in tempo reale grazie all'analisi dei dati resi disponibili a livello di motore di rendering del browser (<https://developers.google.com/web-toolkit/speedtracer>);
- **JSLint** analizzatore statico di codice JavaScript volto a rilevare e impedire l'adozione inconsapevole di "worst practices" in fase di codifica (<http://www.jshint.com>);
- **ApacheBench** per testare l'efficienza prestazionale dell'applicazione lato server mediante la simulazione di un numero arbitrario di richieste da parte dei client (<http://httpd.apache.org/docs/2.2/programs/ab.html>);
- **Eclipse IDE** multiplatforma e cross-language, scelto in particolare come ambiente di sviluppo per la parte server da realizzarsi in Java, che include al suo interno funzionalità di debugging (<http://www.eclipse.org>) utili ai fini della QA;
- plugin **Metrics** per Eclipse, estensione che permette di associare un valore su una scala di riferimento al soddisfacimento di una serie di parametri di qualità del codice sorgente o metriche (<http://metrics.sourceforge.net>);
- il plugin **FindBugs** per Eclipse, al fine di effettuare analisi statica del codice a livello di bytecode alla ricerca di potenziali cause di malfunzionamento (*bug patterns*) o adozione inconsapevole di "worst practices" (<http://findbugs.sourceforge.net>);
- **JUnit** come framework per i test di unità da effettuarsi relativamente alla parte server dell'applicazione (<http://www.junit.org>).

9.2 Tecniche

Responsabili delle attività di controllo interne al gruppo sono i verificatori, che si presuppone essere in ogni caso e senza alcuna deroga distinti dai realizzatori del prodotto soggetto a verifica (programmatore o redattori di documentazione). Al fine di garantire la qualità, i verificatori sono tenuti all'utilizzo di due tecniche di analisi: statica e dinamica.

9.2.1 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 essere 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.

La seconda forma di analisi statica prevede invece l'utilizzo di strumenti appositi denominati **analizzatori statici** e può essere svolta in modo semiautomatico senza richiedere necessariamente l'intervento di un umano. In particolare, come risulta dalla sezione 9.1 si è stabilito di utilizzare degli analizzatori statici tanto per la parte documentale del progetto (come il comando `lacheck`) quanto per la parte propriamente eseguibile (JSLint per la parte JavaScript e FindBugs per la parte Java).

9.2.2 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.

9.3 Verifica dei documenti

La verifica di un documento avviene nel momento in cui il redattore ha finito di redigere completamente il documento. Tale processo avverrà principalmente in tre fasi:

1. Si procederà con un'analisi ortografica, compreso il corretto uso della punteggiatura, degli accenti e degli apostrofi. Per far ciò il verificatore potrà aiutarsi con lo strumento "Verifica ortografia" presente in TexMaker raggiungibile dalla scheda "Modifica->Verifica Ortografia". Successivamente controllerà l'assenza di errori grammaticali, logici e sintattici nei periodi ed infine la coerenza del documento con le norme previste nella sezione 4 "Ambiente documentale".
2. Se è la prima volta che il documento viene verificato, il verificatore procederà con un metodo di tipo walkthrough, come descritto in sezione 9.2.1. Se invece il documento era già stato verificato e rimandato per via di errori, si adopererà un metodo di tipo inspection, come descritto in sezione 9.2.1.
3. Il verificatore prenderà nota in modo ordinato di tutti gli errori riscontrati e procederà con la creazione dei tickets relativi che saranno rivolti al redattore del documento come descritto nella sezione 5.2.3 "Ciclo di vita di un Ticket".

9.4 Verifica del codice

La verifica del codice verrà fatta, come prima cosa, durante lo sviluppo di ogni classe attraverso il proprio IDE che segnalerà gli errori più banali (errori di battitura o piccole mancanze). Lo sviluppatore, prima di caricarlo nel repository, deve assicurarsi che il codice compili e non dia errori o warning. Il verificatore successivamente dovrà assicurarsi che siano state rispettate le convenzioni descritte nella sezione "Progettazione" 7 e in caso di errori dovrà aprire un ticket (rispettando sempre le norme descritte in sezione 5.2.1) rivolto allo sviluppatore che provvederà a risolvere gli errori.