MyTalk

Sofware di comunicazione tra utenti senza requisiti di installazione

clockworkTeam7@gmail.com

Norme di Progetto

v 4.0

Informazioni sul documento

Nome documento	Norme di Progetto
Versione documento	v 4.0
Data creazione	2012/11/28
Data ultima modifica	2013/06/29
Uso documento	Interno
Redazione	Gavagnin Jessica
Verifica	Palmisano Maria Antonietta
Approvazione	Ceseracciu Marco
Lista distribuzione	gruppo *Clockwork* Prof. Tullio Vardanega

Sommario

Questo documento vuol definire le norme volte alla regolamentazione del gruppo *Clockwork* necessarie al processo di sviluppo del progetto **MyTalk**.

Diario delle modifiche

Autore	Modifica	Data	Versione
Ceseracciu Marco	Approvazione del documento	2013/06/29	v 4.0
Palmisano Maria Antonietta	Verifica del documento	2013/06/28	v 3.7
Gavagnin Jessica	Resa la sezione 17 in forma normativa	2013/06/27	v 3.6
Gavagnin Jessica	Resa la sezione 16 in forma normativa	2013/06/27	v 3.5
Gavagnin Jessica	Resa la sezione 15 in forma normativa	2013/06/27	v 3.4
Gavagnin Jessica	Resa la sezione 14 in forma normativa	2013/06/26	v 3.3
Gavagnin Jessica	Inserite le norme riguardanti la numerazione romana ed araba delle pagine nella sezione 7	2013/06/25	v 3.2
Gavagnin Jessica	Accorpate le norme relative alle attività con quelle relative ai documenti da esse prodotte ed eliminato l'avverbio fortemente nelle note 11 e 12	2013/06/24	v 3.1
Furlan Valentino	Approvazione del documento	2013/02/24	v 3.0
La Bruna Agostino	Verifica del documento	2013/02/23	v 2.7
Gavagnin Jessica	Suddiviso il documento in Parti	2013/02/23	v 2.6
Gavagnin Jessica	Aggiunto capitolo Attività di Analisi	2013/02/22	v 2.5
Gavagnin Jessica	Aggiunto capitolo Procedure di Verifica e Attività di Test	2013/02/21	v 2.4
Zohouri Haghian Pardis	Aggiunto capitolo Attività di Progettazione Architetturale e Attività di Progettazione in dettaglio	2013/02/20	v 2.3
Gavagnin Jessica	Aggiunto capitolo Attività di Codifica	2013/02/19	v 2.2
Zohouri Haghian Pardis	Cambiato l'ordine dei capitoli dal capitolo 8 al 13	2013/02/13	v 2.1
Ceseracciu Marco	Approvazione finale documento	2013/01/14	v 2.0
Ceseracciu Marco	Effettuato controllo contenutistico, applicando piccole modifiche chiarificatrici	2013/01/14	v 1.12
Ceseracciu Marco	Effettuate correzioni grammaticali, sintattiche e strutturali	2013/01/14	v 1.11

Ceseracciu Marco	Cambiato ordine dei capitoli, anticipati Gestione delle attività e Glossario	2013/01/14	v 1.10
Furlan Valentino	Aggiunta sezione Procedura di Verifica e Validazione	2013/01/12	v 1.9
Furlan Valentino	Inserita sottosezione Avanzamento di versione	2013/01/12	v 1.8
Furlan Valentino	Sistemata sezione Formati ricorrenti, sistemato anche il formato delle date	2013/01/12	v 1.7
Furlan Valentino	Inserita sez. Rotazione dei ruoli e spostata incompatibilità come sottosezione	2013/01/12	v 1.6
Furlan Valentino	Sistemati capitoli Progettazione e Codifica	2013/01/11	v 1.5
Furlan Valentino	Sistemate sezioni Componenti visive e Formattazione documenti	2013/01/10	v 1.4
Furlan Valentino	Inserimento nuova immagine repository, stesura sezione Cartella Codice	2013/01/10	v 1.3
Furlan Valentino	Stesura capitoli Ruoli	2013/01/09	v 1.2
Furlan Valentino	Inserita sezione Riferimenti	2013/01/09	v 1.1
Bain Giacomo	Approvazione finale documento	2012/12/03	v 1.0
Gavagnin Jessica	Sistemazione e stesura capitoli 9 e 11	2012/12/03	v 0.9
Bain Giacomo	Bozza capitoli 9 e 11	2012/11/30	v 0.8
Gavagnin Jessica	Stesura capitoli 8 e 10	2012/11/30	v 0.7
Gavagnin Jessica	Sistemazione e stesura completa capitoli 4, 5, 6 e 7	2012/11/30	v 0.6
Bain Giacomo	Bozza capitoli 6 e 7	2012/11/29	v 0.5
Gavagnin Jessica	Creazione e stesura dei capitoli 1, 2 e 3	2012/11/29	v 0.4
Zohouri Haghian Pardis	Correzione bozza capitolo 5	2012/11/29	v 0.3
Ceseracciu Marco	Bozza del capitolo 4, repository	2012/11/28	v 0.2
Palmisano Maria Antonietta	Bozza del capitolo 5, sulla struttura dei documenti	2012/11/28	v 0.1

Indice

Elenco delle figure

Elenco delle tabelle

Parte I
Introduzione

1 Scopo del documento

Il documento Norme di Progetto viene redatto per definire tutte le norme che disciplineranno lo svolgimento del progetto. Gli argomenti che le norme trattano riguardano:

- *Relazioni interpersonali*
- *Procedure delle varie attività*
- *Redazione documenti*
- *Codifica*
- *Definizione delle particolarità dei vari documenti*
- *Definizione dell'ambiente di lavoro*

Tutti i membri sono obbligati a visionare questo documento ed a sottostare alle norme descritte, per migliorare l'efficienza, le operazioni di verifica, la coerenza e l'organicità tra i vari file prodotti.
Se si presenterà la necessità, ogni membro del gruppo *Clockwork* potrà suggerire cambiamenti o aggiungere eccezioni all'Amministratore di Progetto.

2 Ambiguità

Per evitare ogni ambiguità riguardante il linguaggio e termini utilizzati nei documenti formali, viene allegato il glossario nel file `Glossario_v2.0.pdf`, dove sono definiti e descritti tutti i termini che sono marcati da una sottolineatura.

3 Riferimenti

3.1 Normativi

- Regole di partecipazione alle revisioni di progetto: `http://www.math.unipd.it/~tullio/IS-1/2012/`

3.2 Informativi

- Guida introduttiva a LaTeX: `http://www.mat.uniroma1.it/centro-calcolo/manuali/impara_latex.pdf`
- Git community book: `http://git-scm.com/book`

Parte II
Norme generali

4 Ruoli

I ruoli necessari per la produzione di **MyTalk** sono i seguenti:

- **Responsabile di Progetto:** ha il dovere di coordinare e gestire l'attuazione del processo di validazione e verifica. È il solo a poter approvare o no la correttezza di un documento perciò è responsabile della corretta realizzazione del prodotto nei confronti del committente. Ha l'obbligo di accertarsi che i lavori assegnati seguino i criteri dettati nel Piano di Progetto e che non ci siano conflitti fra i Verificatori e l'oggetto di verifica. Deve tenere traccia di eventuali anomalie intervenendo tempestivamente. Assieme all'Amministratore è responsabile dell'emanazione dei ticket e della loro assegnazione. È il riferimento del gruppo, deve accogliere e risolvere eventuali problematiche tra i membri, tenendo traccia di tali problemi e delle soluzioni adottate

- **Responsabile Standard:** controlla, tramite i siti `www.webrtc.org` per WebRTC e `www.w3schools.com` per HTML5, se ci sono state delle modifiche.riferendole al Responsabile di Progetto per applicare le strategie di mitigazione (`Piano_di_Progetto_v4.0.pdf`, sez. Analisi dei rischi)

- **Amministratore:** si occupa dell'efficienza e dell'operatività dell'ambiente di sviluppo. Definisce le metodologie e le norme delle attività di verifica, compresa la distribuzione dei resoconti relativi ai test eseguiti, e la gestione e risoluzione di anomalie e discrepanze (`Piano_di_Qualifica_v4.0.pdf`, sez. Comunicazione e risoluzione di anomalie)

- **Analista:** si occupa delle attività di analisi, traducendo il bisogno del cliente in specifica utile al progettista per trovare una soluzione. Deve comprendere cosa il cliente vuole realmente

- **Progettista:** si occupa delle attività di progettazione, indicando la tecnologia più idonea e il modo in cui utilizzare gli strumenti per risolvere il problema indicato dall'Analista

- **Verificatore:** si occupa delle attività di verifica, applicando processi di verifica e validazione, e tiene traccia dei risultati ottenuti[1]. Tali attività riassumeranno gli esiti delle analisi delle misurazioni, individuando eventuali problematiche che verranno presentate al Responsabile di Progetto per essere risolte (`Piano_di_Qualifica_v4.0.pdf`, sez. Gestione amministrativa della revisione)

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

- **Programmatore:** si occupa delle attività di codifica per la realizzazione del prodotto e delle componenti di ausilio. A sua discrezione può applicare procedure di debugging per verificare il codice che ha prodotto; risolvendo anomalie evidenziate dai verificatori

5 Comunicazioni

5.1 Comunicazioni interne

Le comunicazioni interne avverranno principalmente tramite Facebook e il servizio di ticket offerto da GitHub, e utilizzate nel seguente modo:

- **Nota su Facebook:** per comunicazioni rivolte a tutto il gruppo si lasciano delle note nel social network. Il messaggio dovrà iniziare con l'oggetto nel seguente formato [OGGETTO][2]
- **Ticket:** il servizio di ticket offerto da GitHub sarà usato per tutte le comunicazioni da persona a persona, ossia comunicazioni "point-to-point". La natura del ticket si evidenzierà tramite l'apposizione all'oggetto del tag [MESSAGGIO], senza indicare alcuna milestone

5.2 Comunicazioni esterne

Sarà il Responsabile di Progetto, che si occuperà di comunicare con persone esterne al gruppo, come il committente e il proponente.
A tale scopo è stata creata la casella mail *clockworkTeam7@gmail.com* tramite la quale il Responsabile di Progetto parlerà per conto e in nome dell'intero gruppo di progetto.
Sarà a cura del Responsabile di Progetto informare gli altri componenti del gruppo di eventuali corrispondenze pervenute da persone esterne al gruppo: in questo caso si applicheranno le norme relative alle comunicazioni interne definite nella sezione 5.1.

[2] Per rendere nota la visione dei messaggi, bisognerà commentare la nota.

6 Incontri

6.1 Incontri interni

Gli incontri interni saranno fissati dal Responsabile di Progetto.
Tali incontri potranno essere indetti:

- Su richiesta di un membro del gruppo (tramite ticket indirizzato al Responsabile di Progetto) e con approvazione di almeno altri due membri del gruppo (ad esclusione del proponente stesso)
- Su proposta dell'Amministratore di Progetto e con l'assenso del Responsabile di Progetto o viceversa

Il Responsabile di Progetto dovrà occuparsi di comunicare tali incontri tramite un messaggio nel social network e dovrà contenere la **motivazione**, il **nome** del proponente, la **data**, l'**ora** e il **luogo**, esattamente in questo ordine. Tale comunicazione dovrà avvenire con almeno tre giorni di anticipo.
Ogni membro del gruppo ha la facoltà di fare richiesta di un incontro interno. Tale richiesta sarà indirizzata unicamente al Responsabile di Progetto, che la visionerà e pianificherà l'eventuale incontro.
Ogni membro dovrà confermare la sua presenza a tale incontro o spiegare i motivi della sua assenza.
La riunione si considererà valida solo nel caso in cui siano presenti il Responsabile di Progetto e almeno il membro proponente e i sottoscritti alla riunione[3].

6.1.1 Casi particolari

- *Vicinanza*[4] *ad una milestone*

  Le richieste di incontri che ricadono in questo periodo, possono non soddisfare i requisiti sopra scritti. In particolare basterà l'approvazione del Responsabile di progetto, e la riunione potrà svolgersi anche il giorno successivo alla richiesta. Le comunicazioni di tali richieste dovranno seguire le modalità descritte nella sezione 5.1

- *Impossibilità di riunione per mancanza del numero legale*

  In questo caso la riunione si considera annullata d'ufficio. Non ci sono modifiche alle regole sopra scritte o ulteriori provvedimenti

6.2 Incontri esterni

Come incontri esterni si intende un qualsiasi incontro fra un gruppo di rappresentanza del gruppo di progetto e i proponenti o il committente.
Sarà il Responsabile di Progetto a prendere accordi con il committente o i proponenti, utilizzando la procedura descritta nella sezione 5.2.

[3] D'ora in poi questa condizione sarà definita "numero legale".
[4] Vicinanza: $\Delta t \leq \mathbf{1}settimana$.

Ogni membro del gruppo può presentare richieste di incontro, motivando la necessità, al Responsabile di Progetto, il quale dovrà raccogliere almeno due conferme, e successivamente presenze, da parte dei membri del gruppo (escludendo il proponente dell'incontro) riguardo l'effettiva necessità dell'incontro. Nel caso in cui non si raggiungano le conferme necessarie la proposta sarà bocciata, altrimenti il Responsabile di Progetto contatterà la parte esterna di interesse per prendere accordi con la stessa. Le informazioni sull'incontro dovranno essere rese disponibili il prima possibile.
Sarà compito del Responsabile di Progetto redigere in seguito il verbale dell'incontro avvenuto.

7 Repository

Per lo svolgimento di qualsiasi progetto complesso è necessario l'utilizzo di un repository in cui conservare la documentazione e il codice tenendo traccia dello sviluppo.

1. **Ubicazione**: l'indirizzo web in cui trovare il repository del progetto è: `https://github.com/ClockworkTeam/ClockWork`
2. **Struttura**:

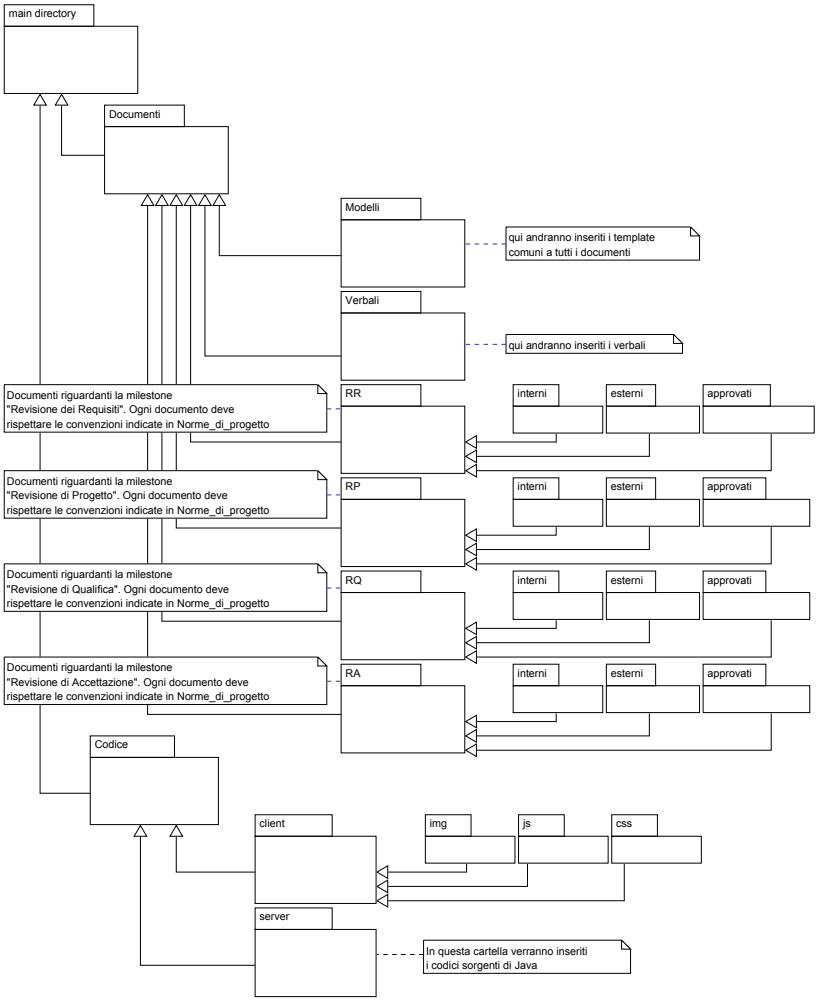

Figura 1: Struttura del Repository

Dalla cartella radice sarà possibile accedere alle cartelle Documenti e Codice.

7.1 Cartella documenti

La cartella Documenti è così suddivisa:

- **Modelli**: in Modelli sono presenti file LaTeX che definiscono la struttura comune a tutti i documenti e la cartella:
  - **img**: in Modelli/img sono presenti tutte le immagini da inserire nei documenti
- **Verbali**: in Verbali saranno presenti tutti i verbali degli incontri effettuati, suddivisi in Formali ed Informali. Questi saranno a loro volta suddivisi per mensilità in apposite cartelle
- **Cartelle Revisioni**: ciascuna sottocartella di revisione (*RR, RP, RQ* ed *RA*) conterrà tutti i documenti utili alla revisione, ognuna avrà, al suo interno le cartelle:
  - **interni**: si troveranno tutti i documenti formali necessari per l'organizzazione interna del gruppo ma non utili al proponente
  - **esterni**: si troveranno tutti i documenti rivolti al proponente
  - **approvati**: si troveranno tutti i documenti in formato PDF dopo essere stati approvati
    - La cartella approvati ha un'ulteriore suddivisione in interni ed esterni

7.2 Cartella Codice

La cartella Codice è così suddivisa:

- **client**
  - css: conterrà i fogli di stile applicati al prodotto
  - img: conterrà le immagini usate nel prodotto[5]
  - js: conterrà i file JavaScript usati nel prodotto
- **server**
  - la gestione di questa cartella è lasciata completamente ad Eclipse che la organizzerà secondo la struttura del progetto Java

[5] Usate nel layout, come loghi e icone.

8 Gestione delle attività

Le attività saranno gestite tramite il servizio di ticketing offerto da GitHub[6] poiché, assieme al servizio di milestone si può tenere facilmente traccia dello stato di avanzamento dei lavori.
Questi strumenti seguiranno tale protocollo:

1. **Creazione milestone**: il Responsabile creerà la milestone per la successiva revisione a cui il gruppo intende di partecipare. L'avanzamento si vedrà dal numero di ticket completati. La milestone avrà le seguenti informazioni:

   - Nome della milestone
   - Nome del Responsabile
   - Data di conclusione della milestone

   (a) **Creazione di un ticket**: il Responsabile creerà un ticket per ogni compito assegnato ad un membro del gruppo. Altri ticket possono essere creati da qualsiasi membro per segnalare *Anomalie* e *Discrepanze* (`Piano_di_Qualifica_v4.0.pdf`, sez. Comunicazione e risoluzione di anomalia). Il ticket avrà obbligatoriamente i seguenti campi definiti:

   - **Title**: nome del compito assegnato
   - **Assegnee**: colui che svolgerà il compito
   - **Milestone**: la milestone per la quale il compito dovrà essere terminato
   - **Label**: si imposterà a seconda dell'argomento
   - **Text**: dovrà contenere la descrizione del compito, la data di inizio e di fine previste ed il collega[7]

2. **Esecuzione compito e risoluzione di un ticket**: ogni membro del gruppo visionerà i ticket assegnatili e li aggiornerà sullo stato. Quando il compito sarà finito, si assegnerà la label "*Completato*" e si menzionerà il ticket nel relativo commit. Il Responsabile confermerà l'effettiva chiusura del ticket.

3. **Ticket di verifica**: ticket di questa tipologia verranno creati secondo quanto scritto precedentemente

4. **Chiusura milestone**: una milestone verrà considerata terminata una volta che tutti i ticket creati siano stati chiusi. Quando una milestone viene conclusa, il Responsabile ripartirà dal punto 1 di tale procedura

[6] Servizio usato anche per le comunicazioni fra membri singoli, vedasi sezione 5.1.
[7] Campo facoltativo.

9 Documenti

In questa sezione si presenteranno i vari standard adottati dal gruppo *Clockwork* per i vari documenti prodotti durante il processo di sviluppo del software.

9.1 Formattazione e copertina

Ogni documento dovrà essere redatto seguendo queste indicazioni[8]:

- **Frontpage:** ogni documento dovrà includere inizialmente il file `frontpage.lyx`[9]. Nel file sono presenti la copertina e la pagina con le informazioni sul documento
- **Diario delle modifiche:** la struttura del diario delle modifiche è trovabile nel file `diario_modifiche.lyx`. Il file sarà copiato ed incollato all'interno della cartella del documento in lavorazione ed incluso nel documento dopo il frontpage. Verrà costantemente aggiornato per mostrare correttamente il diario delle modifiche del documento
- **Impostazioni del "Preambolo di LaTeX":** il file `preface.tex` si dovrà copiare-incollare all'interno della cartella del documento in lavorazione. All'interno di LyX, si dovrà andare in Documento ->Impostazioni, selezionare nell'elenco di sinistra la voce Preambolo di LaTeX, quindi scrivere `\include{preface}`
- **Loghi:** all'interno della cartella img si troveranno i file `logoBW.pdf` e `logo.pdf`. Si dovrà copiare la cartella all'interno della cartella del proprio documento

9.2 Norme tipografiche

Al fine di evitare incongruenze tra i vari file, si rimanda a questa sezione per le informazioni riguardanti l'ortografia, la tipografia e l'assunzione di uno stile uniforme in tutti i documenti.

9.2.1 Stile del testo

- **Grassetto:** sarà utilizzato per evidenziare passaggi estremamente importanti o per evidenziare l'oggetto trattato nel paragrafo negli elenchi
- **Corsivo:** sarà utilizzato per evidenziare passaggi o parole estremamente significativi a cui si vuole dare particolare enfasi e per riportare passaggi provenienti da fonti esterne

[8] Ogni file citato in questa sezione è trovabile nel repository all'interno della cartella /root/Documenti/Modelli.

[9] Il percorso del file fornito a LyX non dovrà essere assoluto, ma relativo.

- **Sottolineato:** termini contenuti nel glossario, se il termine figura più volte nel documento sarà sottolineata solo la prima occorrenza[10]
- **Monospace:** indicherà un passaggio utile alla definizione di un formato, di uno standard adottato o per inserire del codice all'interno del documento
- **Maiuscolo:** una parola scritta maiuscola indicherà un acronimo. Non sono previsti ulteriori casi di parole scritte solo in maiuscolo[11]

9.2.2 Punteggiatura

- **Punteggiatura:** non dovrà mai seguire un carattere di spaziatura, ma sarà seguito da un carattere di spazio. Si ricorda che la lettera maiuscola segue esclusivamente il punto, il punto esclamativo ed il punto di domanda
- **Parentesi:** qualsiasi frase racchiusa fra parentesi non dovrà finire con un carattere di spaziatura e di punteggiatura

9.2.3 Composizione del testo

- **Elenchi:** la prima parola che segue l'oggetto di indentazione deve avere la lettera maiuscola[12]. La fine di ogni paragrafo sarà priva di punteggiatura
- **Note a piè pagina:** dovrà cominciare con l'iniziale della prima lettera maiuscola, terminare con un punto e non deve essere preceduta da alcun carattere di spaziatura

9.2.4 Formati ricorrenti

- **Path:** per gli indirizzi web completi andrà utilizzato il comando appositamente fornito da LyX , Inserisci -> URL, mentre per eventuali indirizzi web relativi verrà utilizzata la formattazione monospace
- **Riferimenti a documenti e/o sezioni:** qualora si dovesse utilizzare il nome di un documento senza riferirlo con l'indirizzo completo, si impone che si segua il formato

  (`Nome_del_Documento_vX.Y.estensione`, sez. Sezione di Riferimento)

  dove:

  - Nome_del_Documento_vX.Y.estensione: nome del documento di riferimento, con carattere monospace
  - Sezione di Riferimento: titolo della sezione a cui ci si riferisce

[10] Non si considerano valide come occorrenze le parole all'interno dei titoli delle sezioni o all'interno di percorsi.
[11] Fanno eccezione tutti i casi in cui la grammatica italiana preveda l'uso della maiuscola.
[12] Eccezione fatta solo nel caso in cui sia il nome di un file e/o cartella.

- **Date:** ove non specificato diversamente, dovrà essere espressa seguendo la notazione definita dalla ISO (ISO 8601), ovvero:

  AAAA/MM/GG

  dove:

  - AAAA rappresenta l'anno trascritto utilizzando esattamente quattro cifre
  - MM rappresenta il mese trascritto utilizzando esattamente due cifre[13]
  - GG rappresenta il giorno trascritto utilizzando esattamente due cifre[14]

- **Nomi ricorrenti:**

  - *Ruoli di progetto*: avranno la prima lettera maiuscola per ogni parola che non sia una preposizione (es: Responsabile di Progetto)
  - *Nomi dei documenti*: avranno la prima lettera maiuscola per ogni parola che non sia una preposizione (es: Norme di Progetto)
  - *Nomi dei file e delle cartelle:* non andranno utilizzati caratteri speciali e/o caratteri accentati; qualora si dovesse utilizzare il nome di un file senza riferirlo con l'indirizzo completo si impone che venga formattato con un carattere monospace
  - *Nomi propri*: si seguirà il formalismo "Cognome Nome"
  - *Nome del gruppo*: ci si riferirà con la dicitura "gruppo" o "gruppo *Clockwork*", con il nome del gruppo in corsivo
  - *Nome del proponente*: ci si riferirà con "Zucchetti SPA " o "proponente"
  - *Nome del progetto*: ci si riferirà con **MyTalk** in grassetto

- **Sigle:** le sigle dei documenti saranno utilizzate esclusivamente all'interno di diagrammi o tabelle secondo il seguente formalismo:

  - AdR = Analisi dei Requisiti
  - SdF = Studio di Fattibilità
  - PdP = Piano di Progetto
  - PdQ = Piano di Qualifica
  - NdP = Norme di Progetto
  - Gl = Glossario

[13] Nel caso il numero potesse essere rappresentato con una singola cifra, si impone l'anteposizione di uno zero alla cifra.

[14] Nel caso il numero potesse essere rappresentato con una singola cifra, si impone l'anteposizione di uno zero alla cifra.

- ST = Specifica Tecnica
- DdP = Definizione di Prodotto
- MS = Manuale Sviluppatore
- MU = Manuale Utente

9.3 Componenti visive

9.3.1 Immagini

Tutte le immagini utilizzate all'interno dei documenti dovranno essere in formato PDF, PNG o EPS e dovranno avere una breve didascalia, vedasi sezione 9.3.3.

9.3.2 Tabelle

Istruzioni e regole per la creazione delle tabelle:

- **Struttura**: saranno tutte longtable. Si posizionerà il cursore in una cella e si andrà su Edit/Table Settings/Longtable e spuntare "use longtable"
- **Intestazione**: sarà in grassetto e la prima lettera maiuscola
- **Didascalia**: ogni tabella sarà fornita di didascalia[15], vedasi sezione 9.3.3. Per fare ciò bisogna aggiungere una riga alla fine della tabella, selezionarla e spuntare "Caption: on" nella finestra indicata nel punto precedente. Per posizionarla alla fine della relativa tabella, bisogna inserire il codice \endlastfoot, all'interno della cella
- **Contenuto**: in ogni cella il testo dovrà iniziare con la lettera maiuscola

9.3.3 Didascalie

Ogni didascalia deve avere le seguenti caratteristiche:

- Numero identificativo incrementabile, in grassetto
- Iniziare con la prima lettera in maiuscolo e finire senza punteggiatura
- Posizionata alla fine della relativa immagine/tabella

9.4 Versionamento

Il formalismo utilizzato per esprimere l'avanzamento di versione sarà il seguente:

$$v\ \{X^{16}\}.\{Y^{17}\}$$

dove:

[15] Esclusi il "Diario delle modifiche" e le tabelle del capitolo Organigramma del PdP.
[16] Da ora in poi questo indice verrà denominato *indice maggiore*.
[17] Da ora in poi questo indice verrà denominato *indice minore*.

- **X**: indica il numero di uscite formali accumulate dal documento[18]
- **Y**: indica i cambiamenti sostanziali[19] susseguitisi dall'ultima uscita formale. Tale numero deve partire da 0 e non ha un limite superiore

Per il corretto utilizzo del formalismo si dovranno seguire le seguenti normative:

- Nei nomi dei file approvati: `Nome_del_Documento_vX.Y.estensione`
- A piè di pagina a seguire il nome dei documenti: Nome del Documento v X.Y
- All'interno di ogni documento in cui si nomina un altro documento: Nome del Documento vX.Y

9.4.1 Avanzamento di versione

Incremento dell'indice maggiore

L'indice maggiore viene incrementato con l'approvazione del Responsabile di Progetto dopo l'attività di Verifica, ovvero quando è consistente rispetto ai contenuti e soddisfa gli obiettivi qualitativi prefissati (`Piano_di_Qualifica_v4.0.pdf`, sez. Pianificazione Strategica e Temporale).
L'incremento di tale indice comporta l'azzeramento dell'indice minore.

Incremento dell'indice minore

L'indice minore viene incrementato nelle seguenti occasioni[20]:

- Stesura di una o più bozze effettuata all'interno della stessa sessione di lavoro[21]
- Stesura di uno o più capitoli effettuata all'interno della stessa sessione di lavoro
- Modifiche sostanziali a bozze e capitoli effettuate all'interno della stessa sessione di lavoro
- Verifica del documento

Si ricorda che l'indice minore ricomincerà da zero quando l'indice maggiore viene incrementato.

[18] Anche se il documento non ha subito sostanziali modifiche da un'uscita formale alla successiva, si impone che avanzi comunque di versione.
[19] Non si considera una modifica sostanziale un aggiornamento sulla punteggiatura o correzione ortografica.
[20] Non viene considerato avanzamento di versione la correzione ortografica.
[21] Si considera una sessione di lavoro le ore di lavoro consecutive effettuate da una stessa persona.

9.5 Formattazione documenti

In ogni pagina di ogni documento formale, ad eccezione della copertina, dovranno comparire:

- **Logo** (versione in bianco e nero) come miniatura, da posizionare nel frontespizio sulla sinistra
- **Nome del progetto** da posizionare nel frontespizio sulla destra
- **Nome e versione**[22] **del documento** da posizionare a piè pagina sulla sinistra
- **Numero di pagina**
  - **Numeri romani**: partiranno dalla pagina successiva a quella della copertina. Inizieranno dal numero I e termineranno alla fine dei vari indici
  - **Numeri arabi**: espresso nel formato:

    n di tot

    da posizionare a piè pagina sulla destra. I numeri arabi partiranno dalla pagina successiva a quella dell'indice e partiranno dal numero 1

Ogni documento prodotto deve contenere:

- **Copertina**:
  - *Nome del progetto* acquisito dal gruppo
  - *Logo del gruppo* sotto il quale deve comparire la mail per contattare il gruppo
  - *Nome e versione del documento*
- **Informazioni generali** del documento:
  - *Nome documento*
  - *Versione documento* [23]
  - *Data creazione* ovvero il giorno in cui ha avuto inizio il lavoro sul documento
  - *Data ultima modifica* ovvero il giorno in cui è stata effettuata l'ultima modifica sul documento
  - *Uso documento* ovvero se il documento ha visibilità interna o esterna
  - *Redazione* ovvero quali membri del gruppo hanno contribuito alla stesura del documento

[22] Il formato utilizzato per tracciare le versioni è definito nella sezione 9.4.
[23] Tale campo dovrà essere omesso nei documenti di tipo Verbale descritti nella sezione 9.6.1.

  - *Verifica* ovvero quali membri del gruppo hanno verificato il documento
  - *Approvazione* ovvero chi ha approvato la versione formale del documento
  - *Lista distribuzione* ovvero la lista di persone alle quali dovrà pervenire il documento[24]

  Inoltre in questa pagina è presente il *Sommario*, dove è presente una breve descrizione del documento

- **Diario delle modifiche**: tabella che riporta le varie modifiche apportare al documento per giungere alla versione finale. Si tratta di una tabella ordinata in maniera decrescente: la prima riga dovrà contenere le informazioni riguardanti le ultime modifiche, mentre l'ultima riga dovrà contenere le informazioni riguardanti le prime modifiche effettuate al documento. Ogni riga dovrà contenere le seguenti informazioni:
  - *Autore*: nominativo di chi ha apportato la modifica in oggetto [25]
  - *Descrizione*: descrizione dei cambiamenti apportati
  - *Data*: quando la modifica è stata effettuata[26]
  - *Versione*: numero della versione corrispondente al documento in cui è stata applicata la modifica
- **Indice dei contenuti**[27]
- **Indice delle figure**[28]
- **Indice delle tabelle**[29]
- **Inclusione dei capitoli**

9.6 Tipi di documenti

9.6.1 Verbali incontri

Per Verbali degli Incontri si intendono quei documenti redatti dal Responsabile di Progetto in occasione di incontri esterni. Per tali documenti è prevista una sola stesura come promemoria dell'incontro avvenuto e tematiche trattate: per tale motivo non è previsto versionamento. Nei verbali non sono presenti Diario delle Modifiche e indice. Il documento sarà denominato secondo il seguente criterio:

[24] Per i documenti interni nella lista saranno presenti il gruppo e il committente, mentre per i documenti esterni sarà presente anche il proponente.
[25] Notazione definita nella sezione 9.2.4.
[26] Notazione definita nella sezione 9.2.4.
[27] Ad eccezione del Glossario.
[28] Nei documenti in cui è presente almeno una figura.
[29] Nei documenti in cui è presente almeno una tabella.

Verbale{data incontro[30]}

Le pagine copertina e informazioni del documento devono sottostare alle regole descritte nella formattazione del documento[31]. Saranno presenti le seguenti informazioni dell'incontro:

- **Data**[32]
- **Luogo** espresso nel formato:

  {città}({provincia}) {indirizzo}{numero civico}

- **Ora ritrovo** espressa dal formalismo:

  {hh}.{mm}[33]

- **Durata incontro** espressa dal formalismo:

  Durata dell'incontro: {x} min

- **Partecipanti interni** indicante la lista dei membri del gruppo[34] presenti all'incontro. La lista deve essere preceduta dall'etichetta:

  Partecipanti del gruppo:

- **Partecipanti esterni** espressi dal formalismo:

  Partecipanti esterni: {Proponenti}

- **Oggetto dell'incontro** in cui si elencano i punti salienti dell'incontro con eventuali chiarimenti e/o decisioni prese a riguardo

9.6.2 Documenti informali

Tutti i documenti non approvati dal Responsabile di Progetto saranno considerati informali. Verranno raccolti nel repository secondo la struttura definita alla sezione 7 e considerati ad uso interno.

9.6.3 Documenti formali

Tutti i documenti approvati dal Responsabile di Progetto si riterranno formali e pronti per essere distribuiti alla loro lista di distribuzione. Ogni volta che si appone una modifica ad un documento già approvato, si inserirà tale modifica nel diario delle modifiche, avvisando il Responsabile di Progetto che provvederà a riapprovare il documento e rilasciare una nuova versione formale. Verranno raccolti nel repository secondo la struttura definita alla sezione 7.

[30] In questo caso il formato della data dev'essere AAAAMMGG.
[31] Ad eccezione del versionamento.
[32] Notazione definita nella sezione 9.2.4.
[33] I campi hh e mm dovranno assolutamente essere espressi tramite due cifre.
[34] Secondo il formalismo indicato nella sezione 9.2.4.

9.7 Procedura di formalizzazione dei documenti

Ogni volta che i redattori considereranno pronto un documento, avviseranno il Responsabile, che lo assegnerà ai verificatori. Se il documento dovesse rispettare tutte le norme, il Responsabile di Progetto potrà approvare il documento, o rimandarlo ai redattori per correzioni non normative.

9.8 Rotazione dei ruoli

Per consentire la rotazione dei ruoli, il gruppo *Clockwork* verrà diviso in 2 sottogruppi, a loro volta divisi[35], così da permettere la presenza di verificatori e analisti/progettisti/programmatori[36] in maniera tale da permettere sia l'avanzamento dei compiti affidati ai due sottogruppi, sia la possibilità di verificare continuamente i compiti affidati.
Quando si raggiungerà il momento di rotazione dei ruoli, ci sarà un'inversione dei ruoli all'interno dei sottogruppi, ovvero i componenti del gruppo minore che hanno svolto il ruolo di verificatore svolgeranno i ruoli di chi era nell'altro gruppo minore (analista, progettista o programmatore[37]) e viceversa.

9.8.1 Incompatibilità

Nel caso in cui si presenteranno incompatibilità fra ruoli, vengono poste le seguenti norme:

- **Redattori-verificatori:** i redattori di una versione di un documento non possono esserne anche i verificatori[38], tranne nei seguenti casi:
  - **Mancanza di verificatori:** verranno nominati più verificatori dal Responsabile, ai quali verranno assegnate la verifica delle parti del documento non redatti da essi

9.9 Informazioni aggiuntive sui documenti

In questa sezione verranno inserite delle informazioni aggiuntive sulla redazione dei documenti e/o su alcune componenti dei documenti.

- Sulle informazioni generali dei documenti: come scritto precedentemente all'interno delle informazioni generali di un documento dovranno essere inserite la lista dei redattori, dei verificatori e degli approvatori del documento. Queste liste non dovranno essere cancellate da una versione all'altra, ma dovranno essere integrate con i nomi delle persone coinvolte nell'aggiornamento di versione. Potrebbe succedere di conseguenza che persone coinvolte nella redazione di un documento compaiano anche fra i verificatori[39]

[35] Da ora in poi questa porzione di gruppo verrà definita *gruppo minore*.
[36] Questi ruoli saranno ricoperti nelle fasi adatte dello svolgimento del progetto.
[37] Questi ruoli verranno ricoperti nelle fasi adatte dello svolgimento del progetto.
[38] Potranno verificare versioni future del documento.
[39] Vedere sezione precedente.

10 Ambiente di progetto

Nella seguente sezione verrà descritto in dettaglio l'ambiente di sviluppo che il gruppo *Clockwork* andrà ad utilizzare per lo svolgimento del progetto.

10.1 Ambiente generale

Questa sezione è dedicata a caratteristiche dell'ambiente di progetto che si ripercuotono su tutti gli altri ambienti.

10.1.1 Sistema operativo

Il sistema operativo utilizzato durante lo sviluppo del progetto non è vincolante, i vari membri del gruppo avranno libertà di scelta sulla piattaforma di sviluppo. Inoltre l'utilizzo di SO diversi permetterà di verificare la portabilità su più SO.

10.2 Ambientale documentale

10.2.1 Scrittura documenti

Avverrà tramite codice LaTeXutilizzando l'editor LyX ($\geqslant$ 2.0), (`http://www.lyx.org`). Verrà creato un documento per ogni capitolo di ogni documento. Tali capitoli saranno uniti tramite LyX (9.1) generando un documento in formato PDF.

10.2.2 Verifica ortografica

Per la verifica ortografica dei documenti si utilizzeranno Hunspell[40] ($\geqslant$ 1.3.2-4build1) e Enchant[41] ($\geqslant$ 1.6.0-7build1), l'utilizzo dei quali avverrà tramite apposito plugin per LyX.
Questa verifica verrà effettuata per ogni capitolo.

10.2.3 Pianificazione

Come supporto alla pianificazione del progetto e realizzazione dei diagrammi di Gantt si utilizzerà il software proprietario Microsoft Project 2010. Il software, fornito dal servizio MSDNAA di Microsoft in collaborazione con l'Università degli studi di Padova, permette la pianificazione delle attività legate allo sviluppo del prodotto e la gestione delle risorse. Tale strumento è compatibile solo con sistemi operativi Windows; in caso di necessità si potrà utilizzare una macchina virtuale Windows XP (fornita dalla MSDNAA) con installato il programma.

[40] `http://hunspell.sourceforge.net/`
[41] `http://abisource.com/projects/enchant/`

10.2.4 Diagrammi UML

Per la definizione dei grafici UML sarà utilizzato il software ArgoUML ($\geqslant 0.34$), che è stato scelto poiché è open source, multipiattaforma ed è in grado di esportare i diagrammi in formato vettoriale.
Si ricorda che ArgoUML non supporta ufficialmente lo standard UML2, dunque verrà prestata attenzione nell'espressione delle notazioni grafiche affinché rispettino le specifiche 2.0.

10.2.5 Mockup

Per la realizzazione di mockup dell'interfaccia grafica si è scelto di utilizzare il software open source Pencil ($\geqslant 2.0.3$).

10.2.6 Documentazione semi-automatica

Si userà Javadoc ($\geqslant 1.5$), funzionalità di Java che permette di creare documentazione relativa ad una porzione di codice partendo dai commenti.

10.3 Ambiente di sviluppo

10.3.1 Strumenti di versionamento

Si è deciso di adoperare Git, perché risulta veloce da utilizzare e di facile apprendimento. Ci appoggeremo al servizio Github (`https://github.com/`) che fornisce un repository *git*, e gli strumenti utili alla collaborazione fra più persone, come i servizi di *ticket*, *wiki* e *milestone*. Per quanto riguarda l'uso di *git* e *Github* sui computer di sviluppo, si è deciso l'uso della versione ufficiale rilasciata dal team di sviluppo di *git* ($\geqslant 1.7.8$) e le interfacce grafiche per i rispettivi sistemi operativi:

- *Github for Windows* ($\geqslant 1.0$)
- *Github for Mac* ($\geqslant$The Snappy, Actually)
- *Git-Cola* ($\geqslant 1.4.3.5-1$) (Linux)

10.3.2 Ambiente di codifica

Si userà Geany ($\geqslant 0.21$) per la scrittura del codice HTML5, CSS3 e Javascript. Si utilizzerà l'IDE multi-linguaggio e multi-piattaforma, Eclipse($\geqslant 3.8.0$) per Java, che fornisce, nella versione base, alcune funzionalità utili di debugging, come l'esecuzione del codice step-by-step, l'impostazione di breakpoint, visualizzazione dei valori di variabili e strutture dati durante l'esecuzione, sospensione e riavvio di thread in esecuzione, ecc. (`http://www.eclipse.org/`).

10.4 Ambiente di verifica e validazione

Di seguito vengono elencati gli strumenti scelti per la verifica e la validazione. Il gruppo potrà avvalersi dei seguenti strumenti per effettuare i processi di verifica:

- **Hunspell ($\geqslant$ 1.3.2-4build1):** vedasi sezione 10.2.2
- **Enchant ($\geqslant$ 1.6.0-7build1):** vedasi sezione 10.2.2
- **Yasca ($\geqslant$ 2.1):** programma open source che permette l'analisi statica del codice HTML (`http://www.scovetta.com/yasca.html`)
- **Chromedriver ($\geqslant$ 23.0.1240.0):** plugin per Google Chrome esegue test direttamente sul browser (`http://code.google.com/p/chromedriver`)
- **Firebug Lite ($\geqslant$ 1.4):** plugin per Google Chrome per l'analisi dinamica (`http://getfirebug.com/releases/lite/chrome`)
- **Eclipse ($\geqslant$ 3.8.0):** vedasi sezione **10.3.2**
- **FindBugs ($\geqslant$ 2.0.0):** plugin per Eclipse, in grado di individuare ed evidenziare staticamente, nel codice sorgente, alcuni tra i più comuni errori nella scrittura di codice Java (come l'uso di worst practice). Ogni errore riscontrato verrà analizzato e discusso per valutarne l'eventuale modifica correttiva (`http://findbugs.sourceforge.net/`)
- **EclEmma ($\geqslant$ 2.2.0):** plugin per Eclipse, in grado di verificare automaticamente la copertura del codice, sia durante la normale esecuzione, sia durante l'analisi dinamica tramite test. I test devono coprire il codice sorgente per intero (`http://www.eclemma.org/`)
- **Metrics Plugin for Eclipse ($\geqslant$ 1.3.6):** plugin per Eclipse che fornisce informazioni riguardanti le misure statiche del codice (vedi sezione 21.9.1 per chiarimenti) come:
  - Complessità ciclomatica
  - Numero di parametri
  - Volume di Halstead
  - Numero livelli di annidamento
  - Indice di utilità
  - Indice di dipendenza

  (`http://metrics.sourceforge.net/`)
- Strumenti di validazione W3C:
  - **Markup Validation Service:** per effettuare test su tutte le pagine navigabili del sistema e verificare l'aderenza HTML5 (`http://validator.w3.org/`)

  - **CSS Validation Service:** per effettuare test sui fogli di stile di tutte le pagine del sistema e verificare l'aderenza allo standard CSS versione 3 (`http://jigsaw.w3.org/css-validator/`)

- **Speed Tracer (⩾2.4):** plugin per Google Chrome che permette di verificare l'efficienza di un'applicazione web durante la sua esecuzione (`http://code.google.com/intl/it-IT/webtoolkit/speedtracer/`)
- **JUnit (⩾4.11):** framework per effettuare test di unità per il linguaggio di programmazione Java (`http://www.junit.org/`)
- **QUnit (⩾1.11.0):** framework per effettuare test di unità per il linguaggio di programmazione JavaScript (`http://qunitjs.com/`)

Parte III
Norme di attività e redazione

11 Procedure di Verifica

La verifica, attività svolta dal verificatore, ha il fine di accertare che i prodotti sviluppati dal gruppo *Clockwork* siano conformi alle norme e convenzioni riportate in questo documento, indipendentemente dalla loro tipologia. Il verificatore, dovrà assicurare, sotto la propria responsabilità, che i prodotti in questione soddisfino i requisiti di qualità descritti nel documento `Piano_di_Qualifica_v4.0.pdf`.

Tipologia errore	Gravità	Priorità dell'intervento
Ortografia e formattazione dei documenti	Bassa	Differibile
Codifica	Media	Breve
Stesura documento	Alta	Urgente
Progettazione	Molto alta	Urgentissimo

Tabella 1: Gravità e priorità degli errori

- **Errori ortografici e formattazione dei documenti:** viene riscontrato un errore ortografico, grammaticale o di formattazione in un documento a seguito dell'attività di verifica descritta nella sezione 11.1. Per risolvere tali errori si seguiranno i seguenti punti:

  1. Il Verificatore risolverà l'errore ed aprirà un ticket (vedasi sezione 8) al Responsabile, specificando che bisognerà effettuare una verifica sul documento in questione
  2. Il Responsabile sceglierà un altro Verificatore al quale far controllare il documento
  3. Il Verificatore scelto effettuerà la verifica[42]

- **Errore di codifica:** viene riscontrato un errore nel codice sorgente come descritto nella sezione 11.2. Per risolvere tali errori si seguiranno i seguenti punti[43]:

  1. I Verificatori dovranno aprire un ticket (vedasi sezione 8) destinato al Programmatore interessato, descrivendo l'errore.
  2. Il Programmatore risolverà l'errore
  3. I Verificatori verificherà le modifiche apportate[44]

[42] Nel caso in cui gli errori persistessero, verrà rieseguita tale procedura.
[43] Questa procedura verrà eseguita sia nel caso in cui l'errore sia logico, sia semantico, sia di sintassi.
[44] Nel caso in cui l'errore persistesse, verrà rieseguita tale procedura esponendo il problema anche al Responsabile.

- **Errori nella stesura di documenti:** se il verificatore rileva una incongruenza, un'ambiguità o un errore logico nel documento si seguiranno i seguenti punti:

  1. I Verificatori dovranno aprire un ticket (vedasi sezione 8) destinato ai membri del gruppo interessati e al Responsabile, descrivendo l'errore
  2. I membri del gruppo richiamati dovranno risolvere l'errore e aggiornare la relativa documentazione
  3. I Verificatori dovranno verificare le modifiche apportate[45]

- **Errore di progettazione:** il verificatore riscontra un errore nella progettazione dell'architettura del sistema. Per risolverlo si seguiranno i seguenti punti:

  1. I Verificatori dovranno richiedere una riunione (vedasi sezione 5.1) con presenti *almeno* tutti i progettisti per sollevare l'errore.
  2. I Progettisti dovranno risolvere l'errore e aggiornare la documentazione
  3. I Verificatori dovranno verificare le modifiche apportate[46]

11.1 Verifica dei documenti

PREMESSA: in questa sottosezione si elencherà solamente la procedura che verrà utilizzata per verificare i documenti. Per maggiori dettagli si faccia riferimento al `Piano_di_Qualifica_v4.0.pdf`. I verificatori, una volta presi in consegna i documenti che hanno raggiunto una versione candidata al rilascio, dovranno applicare la seguente procedura:

- **Controllo ortografico:**

  1. Il Verificatore controllerà che la lingua del documento[47] sia correttamente impostata in italiano
     (a) Dovrà andare su Documento->Impostazioni ->Lingua
     (b) Se la lingua impostata non è "italiano", dovrà correggerla
  2. Il verificatore procederà al controllo ortografico del documento tramite Hunspell o Enchant, andando su Strumenti->Correttore Ortografico.

- **Controllo lessicale:** Il verificatore procederà ad un'attenta lettura dell'intero documento alla ricerca di errori lessicali

[45] Nel caso in cui l'errore persistesse, verrà rieseguita tale procedura richiamando tutti i membri del gruppo.
[46] Nel caso in cui l'errore persistesse, verrà rieseguita tale procedura richiamando tutti i membri del gruppo.
[47] Poiché i documenti saranno formati da un file per ogni capitolo, il verificatore dovrà effettuare questa procedura per ogni singolo file.

- **Diagrammi UML:** nel caso siano presenti dei diagrammi UML, Il verificatore dovrà controllare che tali diagrammi rispettino le norme descritte in sezione 15

Indipendentemente dal metodo di controllo utilizzato, il Verificatore dovrà accertare:

- L'assenza di errori di ortografia, compreso il corretto uso della punteggiatura, degli accenti e degli apostrofi
- L'assenza di errori grammaticali, logici e sintattici nei periodi
- La coerenza del documento con le norme descritte nella sezione 9

Il Verificatore dovrà, inoltre, prendere nota degli errori rilevati, suddivisi ed ordinati per sezione, e aprirà un ticket destinato al redattore del documento in esame affinché corregga gli errori riscontrati[48].

11.2 Verifica del codice

La verifica del codice verrà effettuata inizialmente in fase di sviluppo da ogni componente così da correggere errori banali segnalati in fase di compilazione. Lo sviluppatore deve assicurare che la propria componente software compili senza warning prima di inserirla nel repository.
Il Verificatore dovrà attenersi alle attività descritte nel documento `Piano_di_Qualifica_v4.0.pdf`, sez. Pianificazione strategica e temporale.
Inoltre dovrà aprire un ticket destinato allo sviluppatore indicando gli errori rilevati.

[48] Gli errori di ortografia, possono essere corretti direttamente dal Verificatore.

12 Norme per i Test

12.1 Identificazione dei test

Si seguiranno tali formalismi per indicare i test:

- TS: test di sistema, seguito dal nome identificativo del requisito a cui si riferisce. Se un requisito necessita di più test verranno differenziati aggiungendo una lettera in ordine alfabetico
- TI: test di integrazione, seguito dal nome identificativo della componenente.Se una componente necessita di piu test verranno differenziati aggiungendo una lettera in ordine alfabetico
- TU: test di unità, seguito da una coppia di valori $x.y$, ove
  - X: rappresenta l'identificativo dell'unità sottoposta a test
  - Y: identifica il caso di test

12.2 Alpha-test

Quando dovrà essere eseguito l'alpha-test:

1. L'alpha-test sarà supervisionato dal Responsabile, che annoterà i lavori richiesti per correggere gli errori riscontrati
2. Tutti i membri del gruppo saranno presenti[49]
3. I Verificatori dovranno eseguire i test di sistema descritti nel `Piano_di_Qualifica_v4.0.pdf`
4. Se individuato un malfunzionamento, il verificatore aprirà un ticket destinato al Programmatore responsabile della unità, descrivendo il malfunzionamento e dove si presenta

12.3 Beta-test

Quando dovrà essere eseguito il beta-test:

1. La data e il luogo saranno decisi dal Responsabile accordandosi coi proponenti
2. Tutti i membri del gruppo saranno presenti[50]e i proponenti
3. Verrà chiesto ai proponenti di effettuare i test di sistema definiti nel `Piano_di_Qualifica_v4.0.pdf`
4. Verranno descritti obiettivi e risultati dei test ai proponenti, e saranno fornite le istruzioni basilari per l'utilizzo del software

[49] Ogni assenza deve essere giustificata al Responsabile.
[50] Ogni assenza deve essere giustificata al Responsabile.

5. Verrà nominato un membro del gruppo per annotare commenti ed eventuali malfunzionamenti del software rilevati dal proponente. Al termine del test tale componente redigerà un verbale e aprirà un ticket per ogni malfunzionamento riscontrato[51]

[51] Tale ticket verrà assegnato al Programmatore responsabile

13 Studio di fattibilità

In questo documento si richiede lo studio di ogni capitolato analizzando i seguenti punti:

- **Fattibilità tecnica**: comprende la valutazione dei rischi legati alle tecnologie richieste per soddisfare i requisiti e alla difficoltà stimata da parte del gruppo di apprendere l'utilizzo delle stesse
- **Fattibilità socioeconomica**: comprende la valutazione dei rischi legati agli aspetti socioeconomici del progetto come, ad esempio, i costi da affrontare e la concorrenza di altri fornitori
- **Fattibilità implementativa**: comprende la valutazione dei rischi legati alla soddisfacibilità dei requisiti del capitolato e quindi la stima della capacità del gruppo di completare le richieste obbligatorie, desiderabili e opzionali
- **Altro**: qualsiasi altro fattore di rischio stimato dagli analisti non compreso nei casi precedenti

13.1 Rischi

Per ogni fattore enunciato precedentemente dovranno essere catalogati i rischi come viene descritto nel seguente protocollo:

- **Nome del rischio**: sintetizza il rischio individuato con un nome significativo
- **Descrizione del rischio**: descrive brevemente il rischio individuato
- **Stima di probabilità**: l'analista indica la probabilità che si verifichi la situazione di rischio in esame con un parametro compreso nell'insieme {*Certa, Alta, Media, Bassa, Nulla*}
- **Stima di incidenza**: l'analista indica quanto ritiene grave il verificarsi della situazione di rischio in esame con un parametro compreso nell'insieme {*Catastrofica, Alta, Media, Bassa, Trascurabile*}
- **Giustificazione della stima**: l'analista spiega le motivazioni dell'assegnamento alle stime di probabilità e di incidenza assegnate
- **Analista**: nome dell'analista che sarà contattato dal Responsabile di Progetto (o dal Verificatore per conto del Responsabile di Progetto) nel caso in cui le valutazioni appaiano inesatte o poco giustificate
- **Riferimenti**[52]: elenco puntato di tutte le fonti utilizzate, per consentire una futura verifica delle fonti

[52] Questo punto è facoltativo.

13.2 Vantaggi

Per ogni fattore enunciato precedentemente dovranno essere catalogati i vantaggi come viene descritto nel seguente protocollo:

- **Nome del vantaggio**: sintetizza il vantaggio individuato con un nome significativo
- **Descrizione del vantaggio**: descrive brevemente il vantaggio individuato
- **Stima di incidenza**: indica quanto l'analista ritiene incidente nella valutazione del vantaggio complessivo il vantaggio in esame con un parametro compreso nell'insieme {*Ottimale, Alta, Media, Bassa, Trascurabile*}
- **Giustificazione della stima**: l'analista spiega le motivazioni dell'assegnamento alla stima di incidenza assegnata
- **Analista**: nome dell'analista che sarà contattato dal Responsabile di Progetto (o dal Verificatore per conto del Responsabile) nel caso in cui le valutazioni appaiano inesatte o poco giustificate
- **Riferimenti**[53]: elenco puntato di tutte le fonti utilizzate, per consentire una futura verifica delle fonti

Una volta terminato lo studio di fattibilità di tutti i capitolati, l'analista redattore del documento finale dovrà inserire una conclusione in cui si esporrà:

- Una **valutazione complessiva** dei rischi e dei vantaggi di ciascun capitolato
- Il **capitolato scelto**
- Un'**analisi sintetica** di vantaggi e rischi di ciascun capitolato

[53] Questo punto è facoltativo.

14 Attività di Analisi dei Requisiti

Lo scopo del processo di Analisi è di raccogliere tutti i requisiti che il software finale dovrà possedere. Le fonti dalle quali possono emergere tali requisiti sono in quest'ordine d'importanza:

- Capitolato d'appalto
- Incontri col proponente
- Valutazioni interne al gruppo

Per far emergere i requisiti si seguirà tale procedura:

1. Gli Analisti analizzeranno il capitolato d'appalto ed individueranno i casi d'uso
2. Gli Analisti richiederanno una riunione (vedasi sezione 5.1)
3. Si valuteranno i casi d'uso emersi
4. Si richiederà un incontro col proponente (vedasi sezione 5.2)
5. Si dovranno analizzare e valutare gli elementi emersi durante l'incontro col proponente[54]
6. Gli Analisti analizzeranno casi d'uso trovati e risalteranno i requisiti che soddisfano i vari casi d'uso

Al termine di questo processo sarà necessario che tutti i requisiti identificati vadano raccolti nel documento Analisi dei Requisiti. Saranno inoltre fornite le indicazioni di massima per verificare i requisiti individuati.
I requisiti dovranno:

- Rispettare i vincoli espressi dal proponente
- Coprire le necessità degli utenti
- Essere atomici
- Essere verificabili

[54] Gli elementi possono essere o modifiche ai casi d'uso emersi in precedenza, l'aggiunta di nuovi casi d'uso, o l'eliminazione di casi d'uso.

15 Analisi dei Requisiti

L'Analisi dei Requisiti sarà redatta dagli analisti. In tale documento dovranno comparire ordinatamente tutti i requisiti rilevati dal Capitolato e/o dagli incontri effettuati col proponente.
Il documento conterrà tali informazioni:

- **Descrizione generale:** descrizione del contesto d'uso del prodotto, della tipologia di utenti di destinazione, delle funzioni che il software fornisce all'utente ed i vincoli generali dettati dal capitolato d'appalto
- **Descrizione dei casi d'uso:** vedasi 15.2
- **Elenco dei requisiti:** insieme dei requisiti individuati. Sarà organizzato in maniera tabulare, e si suddividerà in *Ambito utente* ed *Ambito generale* (vedasi 15.1)
- **Tracciamenti:** tabelle di tracciamento tra requisiti ed altri oggetti. Nello specifico si indicherà:
  - Tracciamento dei requisiti con la loro provenienza
  - Tracciamento dei requisiti coi casi d'uso
  - Tracciamento dei casi d'uso coi requisiti

15.1 Requisiti

Ogni requisito dovrà essere il più completo e chiaro possibile. Per tale motivo si richiede che per ogni requisito siano specificati:

- **Fonti**: ovvero da che luogo è emerso il requisito
- **Classificazione**: si raggrupperanno i requisiti in insiemi coerenti in modo da poter definire una segnatura specifica, univoca e gerarchica secondo il seguente formalismo:

  R{AMBITO}{TIPO_REQUISITO}{PRIORITÀ} {GERARCHIA}

  dove:
  - AMBITO: distinguerà l'ambito per cui sarà definito il requisito:
    - **U** indicherà qualsiasi tipo di utente
    - **G** indicherà un requisito generale di sistema
  - TIPO_REQUISITO: distinguerà la tipologia del requisito:
    - **F** indica un requisito funzionale
    - **Q** indica un requisito di qualità
    - **V** indica un requisito di vincolo

  - PRIORITÀ: distinguerà la priorità associata al requisito:
    * **O** indicherà un requisito obbligatorio, ossia un requisito la cui implementazione è fondamentale per la riuscita dell'applicazione
    * **D** indicherà un requisito desiderabile, ossia un requisito la cui implementazione è gradita ma non fondamentale e la cui realizzazione è da intendersi di importanza secondaria rispetto ai requisiti facoltativi
    * **F** indicherà un requisito facoltativo, ossia un requisito la cui implementazione non è da ritenersi vincolante per la riuscita del progetto: saranno requisiti ricavati dal capitolato o dal proponente e che il gruppo si impegnerà ad implementare qualora le risorse disponibili saranno sufficienti
  - GERARCHIA: distinguerà la gerarchia presente tra i vari requisiti

15.2 Casi d'uso

Ogni diagramma contiene l'elenco degli attori coinvolti, le definizioni tecniche utilizzate per ragioni di sintesi nei diagrammi e l'insieme dei diagrammi dei casi d'uso realizzati per rappresentare le funzionalità del software.Ogni diagramma dei casi d'uso dovrà:

- Utilizzare tale formalismo:

  UC{GERARCHIA}

  dove:

  - GERARCHIA: distinguerà la gerarchia presente tra i vari casi d'uso

- Utilizzare il linguaggio di modellazione UML per la creazione dei diagrammi di casi d'uso che facilitino la comprensione di tali requisiti

- Contenere una descrizione testuale di:

  - Precondizioni
  - Scenari principali
  - Scenari alternativi
  - Postcondizioni
  - Attori

  Nel caso dei diagrammi gli scenari dovranno fare esplicito riferimento agli Use Case inseriti nel diagramma per chiarificarne la funzione

- **Descrizione didascalica**: si dovrà accompagnare ogni diagramma di caso d'uso con una descrizione dove dovranno comparire le seguenti informazioni:

- Attori coinvolti
- Scopo e descrizione del requisito
- Precondizione
- Postcondizione
- Flusso principale degli eventi
- Scenari alternativi

16 Attività di Progettazione Architetturale

Al fine di diminuire il livello di rischio dovuto da una progettazione errata e prevenire incongruenze logiche, l'attività di progettazione deve essere effettuata tenendo conto delle seguenti linee guida:

- **Decomposizione modulare**: si identificheranno componenti indipendenti tra loro riducendo il grado di accoppiamento del sistema e aumentando la coesione
- **Componenti terminali**: per evitare di esporre esternamente dettagli implementativi, che causerebbero eccessivo accoppiamento, bisogna riconoscere subito i componenti che non necessitano di ulteriori scomposizioni ottenendo un abbattimento di costi e tempi già dalle prime fasi
- **Accoppiamento e coesione dei componenti**: per attenuare la probabilità del rischio legato alla introduzione di complessità inutile, quindi aumentare la manutenibilità del sistema e abbassare i costi per l'aggiunta di componenti future, bisognerà mantenere un indicatore dei parametri in corso d'opera
- **Grado elevato di coesione e basso di accoppiamento**: al fine di garantire una corretta separazione logica e una buona caratterizzazione dei componenti
- **Principio di astrazione**: al fine di definire la radice della gerarchia delle classi, bisognerà individuare le caratteristiche comuni tra le componenti. Quello che si differenzia dall'insieme di funzionalità iniziale andrà definito tramite specializzazione
- **Astrazione e concretizzazioni**: ogni astrazione corrisponde ad una o più concretizzazioni
- **Componenti astratte**: dovranno essere sufficienti nella caratterizzazione dell'entita desiderata, completi nell'elencare le caratteristiche di interesse dell'utente ed atomiche
- **Integrità concettuale**: adoperare uno stile uniforme per la modellazione delle componenti, da applicare in ogni parte del sistema
- **Complessità delle componenti e delle iterazioni**: bisogna cercare di trovare un compromesso per non complicare il modo in cui una classe comunica col resto del sistema, ed avere una semplice modellazione di tale classe
- **Design Pattern**: individuare i casi in cui si dovranno utilizzare, ed applicarli di conseguenza

- **Enforce intention**: rendere non ambiguo il confine dei moduli e non lasciare spazio nella fase di codifica, rendendo immutabile ciò che non subisce variazioni nel tempo e utilizzando il design pattern Singleton per le classi con una sola istanza
- **Parallelizzazione**: raggiungere un livello di dettaglio tale da permettere una parallelizzazione della codifica

17 Specifica Tecnica

Durante la fase di progettazione i progettisti dovranno adeguarsi alle seguenti specifiche:

- **Diagrammi**: si andrà ad utilizzare il linguaggio UML per definire:
  - *Diagrammi delle classi*: dovrà essere presente all'interno dei documenti l'intera architettura generale e di dettaglio
  - *Diagrammi di flusso*: presente nel caso in cui l'azione di codifica dei programmatori dovesse portare aleatorietà e conseguentemente non garantire il corretto funzionamento dell'applicazione supposta dall'architettura
  - *Diagrammi di package*: presenti sia nell'architettura generale che di dettaglio. I vari package saranno definiti in maniera univoca per poterli distribuire a vari codificatori durante la fase di codifica
- **Design pattern**: per i vari design pattern andremo a specificare:
  - Una *descrizione generale* per presentare brevemente la struttura del design pattern
  - Una *motivazione* di tale design pattern
  - Il *contesto applicativo* dove andremo ad utilizzarlo
- **Classi di verifica**: da sviluppare quando possibile, soprattutto per le classi generali, sono classi fittizie da utilizzare durante la fase di verifica e come prototipo
- **Stile di progettazione**: per la semplificazione degli schemi e per la prossima fase di progettazione si cercheranno di rispettare le seguenti norme
  - *Ricorsione:* nel caso si utilizzasse la tecnica della ricorsione si dovrà stimare l'utilizzo di memoria che si andrà ad occupare ed in caso la memoria occupata fosse eccessiva si andrà ad eliminare la ricorsione
  - *Concorrenza:* si andranno a fornire i diagrammi di flusso ed una stima delle risorse necessarie. Nel caso in cui i benefici ricavati dalla concorrenza non siano equivalenti o superiori alle risorse utilizzate si andrà ad eliminare la concorrenza
  - *Annidamento di chiamata*[55]: la profondità massima di annidamento tollerata è 10. Un valore elevato determina alta complessità e riduce il livello di astrazione del codice
  - *Flussi di condizione:* per la chiarezza e la semplicità di verifica del codice, qualora vengano utilizzati costrutti condizionali (if-then-else) non ci dovrà essere un annidamento maggiore di cinque

[55] Indica il numero di livelli di annidamento dei metodi.

- *Numero di parametri*[56]: il numero massimo di parametri tollerati è 10. Se il numero di parametri supera questa soglia, deve essere ridotto, tramite la creazione di ulteriori classi che contengano più parametri tra loro correlati, aumentando la manutenibilità e l'astrazione del codice

[56] Indica il numero di parametri formali per metodo.

18 Attività di Progettazione in Dettaglio

L'attività di progettazione in dettaglio deve essere effettuata tenendo conto delle seguenti linee guida:

- Definizione dei moduli: il carico di lavoro per lo sviluppo di un modulo deve essere realizzabile per un singolo programmatore
- Ogni modulo deve essere un sottosistema definito, completo di tipi, dati e funzionalità fornite
- Ogni modulo deve essere associato ad uno o più requisiti. Questo è possibile tracciando i moduli coi rispettivi componenti definiti nel documento di Specifica Tecnica

19 Definizione di Prodotto

Tale documento deve contenere i risultati della progettazione di dettaglio (vedasi sezione 18), effettuata sulla base dettata dal documento Specifica Tecnica.
Lo scopo della Definizione di Prodotto deve definire l'architettura di dettaglio, individuando i moduli del sistema affinché raggiungano una dimensione, complessità, accoppiamento e coesione adeguati allo sviluppo del singolo Programmatore.

19.1 Struttura del documento

Il documento Definizione di Prodotto dovrà seguire questo schema[57]:

- **Server:** sarà suddiviso in:
  - **Package transfer:** sarranno descritte le unità relative al package transfer
  - **Package usermanager:** saranno descritte le unità relative al package usermanager
  - **Package functionmanager:** saranno descritte le unità relative al package functionmanager
  - **Package dao:** saranno descritte le unità relative al package dao
  - **Package shared:** saranno descritte le unità relative al package shared
- **Client:** sarà suddiviso in:
  - **Package communication:** saranno descritte le unità relative al package communication
  - **Package model:** saranno descritte le unità relative al package model
  - **Package collection:** saranno descritte le unità relative al package collection
  - **Package view:** saranno descritte le unità relative al package view
- **Diagrammi di sequenza**: saranno inseriti i diagrammi di sequenza nello standard UML per descrivere particolari interazioni del sistema
- **Tracciament**o: si assocerà ogni package definito nel sistema coi rispettivi componenti presenti (vedasi `Specifica_Tecnica_v3.0.pdf`, sez. Architettura Generale). Sarà associato l'elenco delle classi che realizzano la componente

[57] Sarà da tenere in considerazione anche delle norme descritte nella sezione 9.

20 Attività di Codifica

Si dovranno seguire tali norme per aiutare l'attività di verifica del codice:

1. **Iterazioni:** ove si useranno, si imporrà un limite superiore statico
2. **Allocazioni di memoria:** non si useranno dopo l'inizializzazione
3. **Sottoprogrammi/sottoprocedure:** non dovranno superare le 60 righe di codice[58]
4. **Asserzioni:** si utilizzeranno almeno 2 asserzioni per sottoprogramma
5. **Data hiding:** si utilizzerà il massimo livello
6. **Variabili globali:** non si utilizzeranno
7. **Lettura di variabili:** sarà effettuata solo tramite metodi *get*
8. **Scrittura di variabili:** sarà effettuata solo dai metodi *set*
9. **Parametri in input e output:** ad ogni livello bisognerà verificare che i parametri siano validi, in caso contrario, agire di conseguenza
10. **Compilazione:** i file devono essere subito compilati; non dovranno esservi presenti warning, e utilizzeranno la pratica del *continuous integration*
11. **Return:** nei metodi dovrà essere presente un solo return

[58] Tali righe saranno contate escludendo la riga di comando e la riga di dichiarazione.

21 Codifica

21.1 Intestazione del file

Ogni file di codice corrisponderà esattamente ad ogni singola classe, ed inizierà con un'intestazione che dovrà rispecchiare il seguente standard:

```
/*
* Nome: {nome del file}
* Package: {package di appartenenza}
* Autore: {autore del file}
* Data: {data di creazione del file}
* Versione: {versione del file}
*
* Modifiche:
* +--------+----------------+--------------------+
* | Data   | Programmatore  | Modifiche          |
* +--------+----------------+--------------------+
* | AAMMGG | CognomeNome    | - [label]metodo1   |
* |        |                | - [label]metodo2   |
* |        |                | - ....             |
* +--------+----------------+--------------------+
*
*/
```

dove:

- **Nome**: sarà il nome del file comprensivo di estensione
- **Package**: sarà comprensivo della gerarchia del package
- **Autore**: sarà l'autore del file e non necessariamente il programmatore che sta modificando il file attualmente
- **Data**: sarà la data di creazione del file
- **Versione**: indica la versione attuale del file
- **Modifiche**: rappresenta la tabella di avanzamento del file. Per convenzione:
  - **Data**: rappresenta la data dell'avvenuta modifica nel formato AAMMGG con le cifre:
    - AA che rappresentano l'anno
    - MM che rappresentano il mese con eventuali zeri iniziali
    - GG che rappresentano il giorno con eventuali zeri iniziali

- **CognomeNome**: rappresenta il programmatore che ha effettuato le modifiche. Conterrà la prima lettera del cognome seguita dalla prima lettera del nome[59]
- **Modifiche**: rappresenta la lista dei cambiamenti. Per ogni riga dovrà esserci un solo cambiamento. *Label* potrà essere:

  [+]: per indicare la creazione del metodo
  [-]: per indicare l'eliminazione del metodo
  [*]: per indicare la modifica del metodo

21.2 Versionamento

Il versionamento dei file di codice sarà conforme al seguente formalismo:

$$\{X^{60}\}.\{Y^{61}\}$$

ove:

- **X**: è un numero intero incrementale corrispondente alla versione completa e stabile del file[62]
- **Y**: è un numero intero incrementale corrispondente alla modifica Y-esima effettuata dall'ultima versione stabile del file

21.2.1 Avanzamento di versione

Incremento dell'indice maggiore

Ogni volta che un file è completo e stabile, viene incrementato l'indice maggiore. Questo comporta ad impostare l'indice minore a zero (0).
Il file in questione potrà cominciare a sostenere i test per confermare l'effettivo funzionamento delle funzionalità.

Incremento dell'indice minore

Ogni volta che un file subisce una modifica viene incrementato l'indice minore. Si ricorda che tale indice viene azzerato ad ogni incremento dell'indice maggiore.

[59] Nel caso in cui il programmatore abbia un cognome o un nome composto da più parole, si prenderà la prima lettera di ogni parola.
[60] Da ora in poi questo indice verrà denominato come *indice maggiore*.
[61] Da ora in poi questo indice verrà denominato come *indice minore*.
[62] Un file completo è stabile quando tutte le funzionalità pubbliche obbligatorie sono definite e si considerano funzionanti.

21.3 Convenzioni di codifica - Java

21.3.1 Struttura interna delle classi

All'interno di una classe dovranno comparire esattamente nel seguente ordine:

1. Variabili statiche
2. Variabili di istanza
3. Costruttori
4. Metodi

21.3.2 Struttura del codice

- **Dichiarazione di variabili**: per ogni linea ci sarà al massimo una dichiarazione di variabile
- **Blocchi di inizializzazione**: le variabili utilizzate in un blocco saranno dichiarate all'inizio del blocco stesso
- **Nomenclatura variabili**: non ci saranno variabili con lo stesso nome
- **Indentazione del codice**: non si userà il comando *tab*, ma tre caratteri di spaziatura
- **Separazione dei metodi**: i metodi saranno separati tramite una linea vuota
- **Separazione dei blocchi di codice**: i blocchi logici indipendenti saranno separati con una linea vuota
- **Enfatizzazione delle parole chiave**: le parole chiave e le parentesi saranno separate da uno spazio
- **Parentesi graffe**: le parentesi graffe che aprono un blocco di codice saranno posizionate nella stessa linea della parola chiave a cui appartengono; le parentesi graffe che chiudono il blocco saranno scritte in una linea indipendente, come nell'esempio:

```
if (condizione){
  blocco di codice;
}
```

21.3.3 Convenzioni sui nomi

- **Package**: sostantivi brevi ed evocativi la cui iniziale è minuscola. Nel caso siano composte da più parole queste saranno unite e tutte le iniziali saranno minuscole
- **Classi:** sostantivi la cui prima lettera sarà maiuscola. Se composti da più parole, queste saranno unite e l'iniziale di ogni parola sarà maiuscola
- **Interfacce:** stessa nomenclatura delle classi
- **Metodi:** verbi la cui prima lettera deve essere minuscola. Nel caso siano composti da più parole queste saranno unite e l'iniziale di ogni parola dopo la prima sarà maiuscola
- **Variabili:** nomi brevi ed evocativi la cui iniziale è minuscola. Nel caso siano composti da più parole queste saranno unite e l'iniziale di ogni parola dopo la prima sarà maiuscola
- **Costanti:** nomi brevi ed evocativi, scritte in maiuscolo. Nel caso siano composti da più parole queste saranno unite e l'iniziale di ogni parola dopo la prima sarà maiuscola

21.4 Convenzioni di codifica - HTML5

Anche se i moderni browser sono in grado di interpretare alcuni errori, è bene seguire le convenzioni sottoindicate (in caso di dubbio specifico non risolto nelle norme, si faccia riferimento a `http://www.html-5.com/cheat-sheet/`).

21.4.1 Tag

- **Scrittura dei tag**: tutti i tag devono essere scritti in minuscolo
- **Chiusura dei tag**: i tag devono essere sempre chiusi:
  - Tag non vuoti: saranno composti da un tag di apertura seguiti dal blocco e il tag di chiusura. Ad esempio:

    ```
    <p> Blocco1 </p>
    <p> Blocco2 </p>
    ```

  - Tag vuoti: l'apertura del tag e la rispettiva chiusura possono essere nella stessa parentesi uncinata (la chiusura del tag deve avvenire alla fine di questo). Ad esempio:

    ```
    Blocco <br/>
    ```

- **Tag innestati**: in questo caso, la chiusura dei tag potrà avvenire solo se i tag dentro ad esso sono stati chiusi (l'ordine di chiusura dei tag deve essere inverso a quello di apertura). Ad esempio:

  ```
  <p> L'ultima parola è in <b>grassetto</b>.</p>
  ```

21.5 Convenzioni di codifica - CSS3

Il codice CSS3 deve essere valido. Tutte le specifiche si possono trovare presso il sito `http://www.w3schools.com/cssref/css3_browsersupport.asp`.

21.5.1 Selettori

I selettori devono:

- Essere scritti su una riga
- Essere divisi da uno spazio
- **Parentesi graffe**:
  - Di apertura: deve essere nella riga del selettore a cui appartiene
  - Di chiusura: deve essere in una riga a sé stante, senza alcuna indentazione
- Gruppi di selettori non legati logicamente saranno divisi da una linea vuota

```
.menu_navigazione_principale{
}
.menu_navigazione_secondario{
}

.content{
}
```

- **Selettori unici**: ogni selettore sarà su una linea singola. Ad esempio:

```
#form td.uno,
#form td.due{
    blocco codice
}
```

21.5.2 Proprietà

Le proprietà sono la parte interna alle parentesi graffe e descrivono come i selettori devono essere visualizzati.
Ogni proprietà deve:

- Essere su una riga
- Indentata con tre spazi
- Avere uno spazio dopo il nome della proprietà e uno prima del valore, ad esempio:

```
.selettore{
    proprietà : valore;
}
```

- Terminare con un punto e virgola (;)
- Le proprietà multivalore devono essere separate da una virgola seguita da uno spazio

21.6 Convenzioni di codifica - Javascript

Per tali convenzioni si farà riferimento alla guida che Google mette a disposizione, tramite il sito `http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml`.

21.7 Convenzioni di codifica - SQLite

Per SQLite si dovranno seguire tali convenzioni:

- Nomi delle tabelle: in minuscolo. Se composti da più parole, queste saranno unite con la prima lettera maiuscola
- Nomi delle colonne: in minuscolo. Se composti da più parole, saranno separate da un simbolo di *underscore*

21.8 Metriche

In base ai criteri elencati nella sezione 10.4 si definiscono le seguenti metriche per la composizione di codice di qualità:

- **Complessità ciclomatica:** rappresenta la complessità di un metodo, basata sulla misurazione del numero di cammini linearmente indipendenti che attraversano il grafo di flusso di controllo, dove:
  - Nodi: rappresentano gruppi indivisibili di istruzioni
  - Archi: connettono due nodi se le istruzioni di un nodo possono essere eseguite immediatamente dopo le istruzioni dell'altro nodo

  Un valore elevato di complessità riduce la manutenibilità e la possibilità di riuso del metodo: se dovesse risultare tale, parte delle sue funzionalità devono essere demandate ad altri metodi da richiamare.

  Particolare attenzione va posta sulla stima di questo valore: il costrutto switch, ad esempio, moltiplica i cammini linearmente indipendenti e quindi può comportare una misurazione di complessità ciclomatica molto elevata (e oltre i limiti imposti). Tuttavia potrebbe verificarsi l'eventualità che, al fine di garantire una velocità di esecuzione maggiore per un certo metodo, si decida di assumere limiti di complessità ciclomatica più laschi. I valori di

complessità ciclomatica misurati verranno trattati, dunque, con le dovute considerazioni. Il valore ideale di complessità ciclomatica massima posto come obiettivo è dieci

- **Numero di parametri:** vedasi sezione 17
- **Volume di Halstead:** rappresenta il contenuto informativo di un programma, definito da Halstead, viene calcolato come lunghezza del programma moltiplicata per il logaritmo in base due della somma di operatori unici. Il valore dovrebbe essere compreso tra 20 e 1000
- **Numero di variabili locali:** denota il numero di variabili locali interne a ciascun metodo. Come per il numero di campi dati per classe, potrebbe essere necessario incapsulare alcune di queste variabili in nuove classi coese qualora il numero sia troppo elevato
- **Numero di livelli di annidamento:** vedasi sezione 17
- **Grado di accoppiamento:**
  - **Indice di utilità**[63]**:** se troppo basso indicherà che il package non fornisce molte funzionalità al suo esterno, potrebbe essere scarsamente utile; se troppo alto indicherà che altre classi sono strettamente dipendenti dal package in questione, e quindi potrebbe accadere che eventuali modifiche ad esso comportino costi elevati di adattamento delle classi che vi dipendono, qualora non fosse stato progettato adeguatamente il sistema di interfacce
  - **Indice di dipendenza**[64]**:** va sempre minimizzato, aumentando le funzionalità proprie di un package, senza la necessità di affidarsi al servizio offerto da altre classi esterne

21.9 Procedura di Verifica e Validazione

Di seguito vengono riportate le norme che regolano le attività di verifica riguardanti al codice[65].

21.9.1 Analisi statica

- **Analisi del flusso di controllo:** si verificherà, analizzando i vari flussi possibili, che il codice segua la sequenza specificata. Si accerterà che il codice sia ben strutturato, che non esistano parti di codice che non vengono mai raggiunte o che non terminano

[63] Indica il numero di classi esterne al package che dipendono da classi interne ad esso.
[64] Indica il numero di classi interne al package che dipendono da classi esterne ad esso.
[65] Per una specifica dettagliata delle tecniche e delle modalità con cui verranno condotte tali attività si rimanda al `Piano_di_Qualifica_v4.0.pdf`.

- **Analisi di flusso dei dati:** bisogna assicurare che il flusso di dati non usi variabili non ancora inizializzate o prive di valore, oppure che si scriva più volte prima di usare una variabile
- **Analisi flusso d'informazione:** bisogna verificare che input ed output di ogni unità di codice (o più unità) rientrino nelle specifiche del programma
- **Verifica formale del codice:** bisogna verificare la correttezza del codice rispetto alla specifica dei requisiti

21.9.2 Analisi dinamica

- **Test di unità:** test applicati alle singole unità di sistema al fine di verificare la presenza di malfunzionamenti. Bisognerà fare tali test con il massimo grado di parallelismo
- **Test di integrazione:** test effettuato per verificare che i componenti formati dall'unione delle varie unità che hanno superato il test di unità cooperino nel modo corretto
- **Test di sistema e collaudo:** test eseguito sull'intero sistema allo scopo di accertare che il sistema prodotto adempia ai requisiti richiesti e che riesca ad adattarsi correttamente al contesto richiesto dal proponente. Il collaudo sarà sul software finito e, se superato, verrà seguito dal rilascio del prodotto
- **Test di regressione**: nel caso fosse necessario apportare delle modifiche ad un singolo componente, si ricominceranno i test a partire da quelli di unità

22 Glossario

Il glossario sarà unico e riassuntivo per tutti i documenti e riporterà al suo interno tutte le definizioni, in ordine lessicografico, delle parole che possono generare confusione o ambiguità all'interno dei vari documenti, che saranno sottolineate alla loro prima occorrenza nel documento.
Questo documento sarà formattato come descritto nella sezione 9.

22.1 Inserimento vocaboli

Data l'universalità del glossario rispetto a tutti i documenti redatti dal gruppo *Clockwork*, viene nominato Responsabile del Glossario il Responsabile di Progetto. Per poter chiedere l'inserimento di un nuovo vocabolo si dovrà quindi fare richiesta al Responsabile di Progetto che si appresterà a decidere se la parola presa in esame possa essere fonte di ambiguità e/o confusione. Per procedere all'inserimento di una nuova parola il Responsabile di Progetto dovrà seguire le seguenti regole:

- Verificare la presenza della parola presa in esame
- Inserire il vocabolo nella posizione che gli compete