TEAM
COMMITTED

Universitá degli Studi di Padova

Norme di Progetto V2.0

Informazioni sul documento

Nome documento	Norme di Progetto
Data documento	2012/01/31
Redattori	• Giorgio Maggiolo • Alessandro Cornaglia
Verificatori	• Lorenzo Braghetto • Massimo Dalla Pietá
Approvazione	• Giorgio Maggiolo • Giacomo Quadrio
Uso documento	Interno
Lista distribuzione	*Team Committed*

Sommario

Questo documento vuol definire le norme volte alla regolamentazione del gruppo *Team Committed* necessarie al processo di sviluppo del progetto SecurityGame

Diario delle modifiche

Modifica	Autore	Data	Versione
Approvato.	Giacomo Quadrio	2012/01/31	V2.0
Verificato. In attesa di approvazione hline *Completato capitolo 8. In attesa di verifica*	Massimo Dalla Pietá Alessandro Cornaglia	2012/01/30 2012/01/30	V1.5 V1.4
Completato capitolo 7	Alessandro Cornaglia	2012/01/29	V1.3
Scritta la norma sul repository del codice sorgente	Alessandro Cornaglia	2012/01/28	V1.2
Iniziate modifiche dopo correzione Prof. Vardanega. Corretti capitoli 1, 5 e 9. Aggiunto nuovo capitolo 6	Alessandro Cornaglia	2012/01/28	V1.1
Approvato	Giorgio Maggiolo	2011/12/12	V1.0
Corretto sezione 8.6 gestione attività. Revisione completata.	Lorenzo Braghetto	2011/12/11	V0.14
Completata la prima stesura delle Norme di Progetto	Giorgio Maggiolo	2011/12/11	V0.13
Modificato quanto scritto per i casi d'uso e il versionamento dei requisiti.	Giorgio Maggiolo	2011/12/11	V0.12
Inserite le versioni degli strumenti di lavoro	Giorgio Maggiolo	2011/12/10	V0.11
Inseriti gli strumenti di lavoro nel capitolo sugli strumenti. Da scrivere in lungo	Giorgio Maggiolo	2011/12/10	V0.10
Corretti alcune imprecisioni nei capitoli 1,2,3,4,5	Giorgio Maggiolo	2011/12/09	V0.9
Finito capitolo 7	Giorgio Maggiolo	2011/12/09	V0.8
Cominciato capitolo 8 e sistemazione della struttura del capitolo 9	Giorgio Maggiolo	2011/12/09	V0.7
Capitolo 6 iniziato. Manca una parte che è da concordare. Capitolo 9 iniziato (anche questo da concordare).	Giorgio Maggiolo	2011/12/09	V0.6
Finita la scrittura del capitolo 5	Giorgio Maggiolo	2011/12/08	V0.5
Scrittura del capitolo 4 completata. Capitolo 5 in progressione	Giorgio Maggiolo	2011/12/07	V.04
Iniziata la scrittura del capitolo 5. Il capitolo 4 deve ancora essere finito. Dall'1 al 3 sono finiti.	Giorgio Maggiolo	2011/12/07	V0.3
Sistemazione del file sorgente per fare in modo che includa tutti i capitoli come file a parte, cosí come la prima pagina. Iniziata la stesura del capitolo 3.	Giorgio Maggiolo	2011/12/07	V0.2

Inizio creazione del documento. Creato lo schema base e iniziato a scrivere l'introduzione e il capitolo sulle comunicazioni. Impostato inoltre lo schema grafico iniziale del documento stesso.	Giorgio Maggiolo	2001/12/07	V0.1

Indice

1 Introduzione

1.1 Scopo del documento

Il documento "Norme di Progetto"[1] viene redatto allo scopo di definire tutto l'insieme di norme che regolamenteranno lo svolgimento del progetto. Le suddette norme verteranno su tutti gli aspetti del progetto:

- ***Relazioni interpersonali***: comunicazione fra le varie figure professionali all'interno del gruppo di progetto
- ***Redazione documenti***: stili di redazione dei vari documenti interni e/o esterni
- ***Codifica***: stili e convenzioni di scrittura del codice sorgente
- ***Procedure di automazione***: strumenti e procedure per l'automazione di attività tecniche
- ***Definizione dell'ambiente di lavoro***: programmi utilizzati dall'intero gruppo di progetto

Tutti i membri del gruppo di progetto sottoscrivono le norme ivi contenute e vi sottostanno, in modo da migliorare la coerenza fra i vari documenti e migliorare efficienza ed efficacia dei vari file prodotti.

Qualora si renda necessario, un qualsiasi membro potrà proporre all'*Amministratore di Progetto* una modifica alle NdP il quale, sentito il parere di tutti gli altri membri del gruppo, valuterà se effettuare la modifica o meno.

1.2 Ambiguità

Al fine di evitare ogni ambiguità relativa al linguaggio e ai termini utilizzati nei documenti formali, il glossario viene incluso nel file `Glossario-V2.0.pdf`, dove vengono definiti e descritti i termini marcati da una sottolineatura.

[1] D'ora in poi chiamato "NdP", v.d. sezione 5.2.4

2 Comunicazioni

2.1 Comunicazioni interne

Le comunicazioni interne dovranno passare attraverso i canali Facebook o tramite il servizio di ticket di Github. In particolare:

- **Nota su Facebook:** le note su Facebook dovranno essere solo ed esclusivamente per comunicare con tutti i membri del progetto (a.k.a. Mailing List) e dovranno essere pubblicate all'interno del gruppo privato nel social network. Tutti i membri del progetto dovranno confermare la lettura della nota apponendo un messaggio in risposta ad essa.

- **Ticket:** il servizio di ticket offerto da Github dovrà essere usato per tutte le comunicazioni "*point-to-point*", ovvero da persona a persona. Basterà creare un nuovo ticket apponendo all'oggetto dello stesso il tag **[MESSAGGIO]**, per evidenziare in questo modo la natura del ticket, senza indicare alcuna *milestone*.

2.2 Comunicazioni esterne

Il Responsabile di Progetto sarà colui dovrà mantenere i contatti con Committente, Proponente e qualsiasi altra figura esterna al gruppo.

A tale proposito é stato creato un indirizzo e-mail tramite il quale il Responsabile potrà parlare per conto e nel nome dell'intero gruppo di progetto.
Tale mail é:

teamcommitted@gmail.com

Il Responsabile dovrà tenere informato il gruppo rispetto ad eventuali comunicazioni da e per gli esterni tramite quanto scritto nella sezione 2.1

3 Riunioni

3.1 Riunioni interne

Le riunioni interne sono indette dal *Responsabile di Progetto* e potranno essere indette:

- Su richiesta di un membro del gruppo (via ticket indirizzato al *Responsabile di Progetto*) e con approvazione di almeno due membri (escluso il proponente stesso)
- Su proposta dell'*Amministratore di Progetto* e con l'assenso del *Responsabile di Progetto* o viceversa

La proposta di riunione interna, da parte di uno dei membri del gruppo, dovrà contenere il **nome** del proponente e la **motivazione** di tale richiesta. Se almeno due persone sottoscriveranno la richiesta, il *Responsabile di Progetto* dovrà organizzare suddetta riunione entro 3 giorni proponendo due date possibili (questa comunicazione dovrà seguire quanto scritto nella sezione 2.1). La riunione si considererà valida solo nel caso in cui siano presenti il *Responsabile di Progetto* e almeno il membro proponente e i sottoscrittori della riunione[2].
Nel caso di proposta da parte dell'*Amministratore di Progetto* o del *Responsabile di Progetto* , si dovrà seguire quanto indicato nella sezione 2.1 e dovrà esserci solamente l'approvazione della controparte responsabile dell'approvazione.

3.1.1 Casi particolari

Viste e considerate tutte le esigenze dei membri del gruppo, si reputa necessario considerare i seguenti casi particolari:

- **Caso 1:** *vicinanza con una milestone*[3].
  In questo caso le regole che prescrivono l'accettazione di una richiesta di riunione non si applicano. In sostituzione: "*Nel caso di richiesta di riunione basterà l'approvazione del solo* Responsabile di Progetto *che dovrà indire una riunione di gruppo nel più breve tempo possibile (possibilmente il giorno successivo all'approvazione). Queste comunicazioni dovranno seguire l'iter descritto nella sezione 2.1*
- **Caso 2:** *impossibilità di riunione per mancanza del numero legale alla 2a riunione*
  In questo caso la riunione si considera annullata d'ufficio. Non ci sono ulteriori provvedimenti o modifiche alle regole sopra scritte particolari.

3.2 Riunioni esterne

Come **riunione esterna** si intende un qualsiasi incontro fra *Proponente* \ *Committente* e un gruppo di rappresentanza del gruppo di progetto[4].

[2] D'ora in poi questa condizione verrà chiamata "*numero legale*"
[3] **Vicinanza** = $(\Delta t \leq \mathbf{1}settimana)$
[4] É possibile che l'intero gruppo partecipi all'incontro

La richiesta di riunione dovrà pervenire attraverso il servizio di ticket con le stesse modalità indicate nella sezione 3.1. Faranno fede solo le richieste da parte di almeno un membro del gruppo di progetto abbinata ad almeno due sottoscrizioni.

Una volta che è stata approvata una richiesta, il *Responsabile di Progetto* dovrà contattare, con le modalità scritte nella sezione 2.2, il *Proponente \ Committente* e organizzare l'incontro. A questo incontro dovrà esserci il *numero legale*. Il *Responsabile di Progetto* si assume l'incarico inoltre di informare tutti i membri del progetto della riunione e di verificare il giorno prima la presenza del numero legale.

Di ogni incontro si dovrà redigere un verbale che dovrà essere spedito in copia a tutti i presenti alla riunione per verifica e conferma di quanto scritto. É richiesta conferma scritta di ogni presente attraverso le modalità descritte nella sezione 2.1.

4 Repository

Per un corretto e più rapido sviluppo delle risorse, sono inserite qui le specifiche del repository.

4.1 Struttura repository

4.1.1 Progetto

Il repository dell'intero progetto è possibile trovarlo all'indirizzo:

`https://github.com/TeamCommitted`

A questo indirizzo sarà possibile trovare una lista di tutti i repository creati.

4.1.2 Documentazione

L'indirizzo root del repository dei documenti è:

`https://github.com/TeamCommitted/Team-Committed---Documenti`

Il repository è organizzato in maniera gerarchica per revisione ed è strutturato con la seguente struttura[5]:

- `/root/` conterrà le cartelle delle varie revisioni[6] e la cartella *Modelli*
- `/root/Modelli/` conterrà tutti i file che **dovranno** essere applicati ad ogni documento LyX[7]. In particolare:
  - `frontpage.lyx` è la pagina che genera in automatico la copertina di ogni documento. Dovrà essere inclusa come documento figlio
  - `preface.tex` è il file di impostazioni di LaTeXche dovrà essere importato nella *"Prefazione di LaTeX"* e correttamente modificato come descritto al suo interno
  - `Diario_modifiche.lyx` conterrà il codice LaTeXper generare il *Diario delle modifiche.* La documentazione per il suo uso è all'interno del documento stesso
- `/root/Modelli/img/` conterrà le immagini che tutti i documenti dovranno avere

[5] In questa sottosezione ci si riferirà all'url del repository della documentazione con *"`/root/`"*

[6] v.d. nota 8

[7] Per ulteriori informazioni sulle modalità di applicazione di questi modelli, v.d. sezione 5.1 a pagina 13

- `/root/XX/` conterrà tutta la documentazione riguardante una specifica revisione[8]. All'interno della cartella i documenti dovranno essere divisi fra **interni** ed **esterni** (questa differenziazione dovrà ripercuotersi a livello di cartella) e ogni documento dovrà essere inserito in una sua cartella dedicata, che conterrà anche le immagini specifiche di quel documento

4.1.3 Codice

L'indirizzo root del repository del codice è:

`https://github.com/TeamCommitted/SafetyGame-Sources`

Il repository è organizzato a seconda del ed è strutturato con la seguente struttura[9]:

- `/root/` conterrà le cartelle dei vari prodotti realizzati o in fase di realizzazione
- `/root/pc/` conterrà i sorgenti dell'applicazione, lato *desktop*
- `/root/mobile/` conterrà i sorgenti dell'applicazione, lato *mobile*
- `/root/web/` conterrà i sorgenti dell'applicazione, lato *web*

L'interno di ogni cartella contenente i sorgenti dovrà invece essere suddivisa in questa maniera:

- `./src/` conterrà i sorgenti veri e propri dell'applicazione
- `./test/` conterrà il codice creato per testare quanto contenuto in `./src/`
- `./doc/` conterrà la documentazione riguardante il software

4.2 Note di utilizzo dei repository

4.2.1 Documentazione

- **Formati file**: il repository della documentazione potrà contenere solamente file di testo in formato LyX, file `.pptx` delle presentazioni e i file `PDF` sia dei documenti che delle presentazioni; le immagini dovranno essere in formato `.png` o `.jpg`
- **Documenti rilasciati**: una volta rilasciato un documento, si dovrà allegare il file `PDF` del documento all'interno della stessa cartella del documento

[8] **XX** è acronimo di una revisione(v.d. sezione 5.2.4)
[9] V.d. nota 5

4.2.2 Codice

- **Formati file**
  - **Codice sorgente:** il repository del codice sorgente potrà contenere solamenti file sorgenti nei seguenti formati: `.java`
    - `.class`
    - `.html`
    - `.css`
    - `.xml`
    - `.sql`
    - `.js`
    - `.apk`
    - `.bluej`
    - `.ctxt`
  - **Immagini:** le immagini invece dovranno essere in formato `.jpg` e/o `.png`
  - **Documentazione:** la documentazione prodotta dovrà essere in formato `.html` e/o `.txt`

5 Documenti

In questo capitolo verranno elencate tutte le convenzioni che il *Team Committed* ha adottato riguardo alla redazione di tutta la documentazione.

5.1 Formattazione e copertina

Ogni documento dovrà essere redatto seguendo queste indicazioni[10]:

- **Copertina:** ogni file LyX dovrà includere come primo oggetto il file `frontpage.lyx`[11]. Il file non dovrà essere spostato dalla sua posizione iniziale in quanto sarà Lyx, assieme alle impostazioni contenute nel file `preface.tex`, a modificare automaticamente il file.
- **Diario delle modifiche:** il codice del *diario delle modifiche* è possibile trovarlo nel file `Diario_modifiche.lyx`. Il file contiene il codice LaTeXche dovrà essere copiato ed incollato subito dopo l'inclusione della copertina. All'interno si troveranno le istruzioni su come utilizzare il file e come deve essere modificato il codice per mostrare correttamente il diario delle modifiche. Non è necessario copiare e incollare il file all'interno della cartella del documento in lavorazione, in quanto si dovrà invece copiare-incollare il suo contenuto e modificarlo di conseguenza all'interno del documento che si sta modificando.
- **Impostazioni del "*Preambolo di LaTeX*":** le impostazioni che dovranno essere applicate ad ogni documento è possibile trovarle all'interno del file `preface.tex`. Questo file dovrà essere copiato-incollato all'interno della cartella del proprio documento in lavorazione. Inoltre dovrà essere modificato (la copia "*locale*") secondo quanto scritto al suo interno. All'interno di LyX, invece, 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 `/root/Modelli/img` si troveranno i file `Logo_team_Label.jpg` e `LogoTeam.jpg`[12]. Si dovrà creare all'interno della cartella del documento in lavorazione una nuova cartella chiamata *img* e copiarvi all'interno il *logo piccolo*

Nel caso si rendessero necessarie delle modifiche al codice di uno di questi due file, si dovranno seguire le indicazioni descritte nella Sezione 2.1 a pagina 7, con destinatario del ticket l'*Amministratore di Progetto* .

[10] Ogni file citato in questa sezione è possibile trovarlo nel repository all'interno della cartella `/root/Modelli/`

[11] Il percorso del file fornito a LyX non dovrà essere assoluto, ma **relativo** (ex. "`../../../Modelli/frontpage.lyx`")

[12] rispettivamente indicati come "*logo grande*" e "*logo piccolo*"

5.2 Norme tipografiche

Allo scopo di uniformare lo stile tipografico utilizzato dai vari redattori nei vari documenti, si elencano qui una serie di regole riguardanti lo stile ortografico, tipografico e le varie assunzioni che possono essere riassunte in uno *"stile uniformato applicato ai vari documenti"*

5.2.1 Stile del testo

- **Grassetto:** il grassetto può essere usato in differenti situazioni:
  - **Elenchi puntati:** in questi casi è d'obbligo usare il grassetto per evidenziare il concetto esplicato poi nella continuazione del punto trattato. Gli *elenchi numerati* non sono soggetti a questa norma
  - **Descrizioni:** v.d. sopra
  - **Altre situazioni:** in altri contesti, il grassetto va esclusivamente utilizzato per evidenziare parole chiave o passaggi all'interno della frase
- **Corsivo:** il corsivo deve essere utilizzato nei seguenti casi:
  - **Citazioni:** quando si deve citare una o più frasi, queste devono essere scritte in corsivo[13]
  - **Nomi particolari:** il corsivo dovrà essere usato quando si parla di figure particolari piuttosto che di aziende (ex. *Team Committed* o *Amministratore di Progetto*)
  - **Altre situazioni:** in altre situazioni, il corsivo va utilizzato per enfatizzare parole importanti all'interno della frase. Differisce dal grassetto perché mentre il primo evidenzia parole o frasi chiavi del discorso, il secondo mette in luce delle parole o frasi importanti, ma non essenziali ai fini della frase stessa
- **Sottolineato:** le uniche parole sottolineate saranno quelle per le quali è presente una corrispondenza nel glossario. Ogni occorrenza[14] di ogni parola presente nel glossario dovrà essere sottolineata
- **Monospace:** il carattere monospace dovrà indicare 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[15].

[13] Attenzione! Le virgolette che contrassegnano una citazione non devono essere anch'esse in corsivo, ma in stile standard

[14] Non si considerano valide come occorrenze le parole all'interno dei titoli delle sezione o all'interno dei percorsi

[15] Fanno esclusione tutti i casi in cui la grammatica italiana preveda l'uso della maiuscola

5.2.2 Punteggiatura

- **Punteggiatura:** qualsiasi segno di punteggiatura non deve seguire un carattere di spaziatura, ma deve essere seguito da un carattere di spazio
- **Lettera maiuscola:** una lettera maiuscola deve seguire esclusivamente un punto, un punto esclamativo o un punto interrogativo
- **Parentesi**[16]**:** una qualsiasi frase racchiusa fra parentesi non deve iniziare con un carattere di spaziatura e non deve chiudersi con un carattere di punteggiatura e/o di spaziatura

5.2.3 Composizione del testo

- **Elenchi:** la prima parola di ogni punto dell'elenco deve avere l'iniziale maiuscola. La fine di ogni punto dell'elenco deve essere priva di punteggiatura
- **Note a piè di pagina:** ogni nota a piè di pagina dovrà cominciare con l'iniziale della prima lettera maiuscola, e non deve essere preceduta da alcun carattere di spaziatura. Ogni nota a piè di pagina non dovrà terminare con alcun carattere di punteggiatura e/o carattere di spaziatura

5.2.4 Formati ricorrenti

- **Path:** gli indirizzi web ed e-mail dovranno essere scritti utilizzando il percorso assoluto. Non sono ammessi percorsi relativi (a meno dell'uso di convenzioni [per un esempio v.d. 4.1.2 a pagina 10])
- **Date:** ove non specificato diversamente, le date dovranno essere espresse seguendo la notazione definita dalla ISO (ISO 8601), ovvero:

  AAAA/MM/GG

  dove:

  - **GG:** rappresenta il giorno scritto utilizzando esattamente due cifre
  - **MM:** rappresenta il mese scritto utilizzando esattamente due cifre
  - **AAAA:** rappresenta l'anno scritto utilizzando esattamente quattro cifre

- **Nomi ricorrenti:**

  - **Ruoli di progetto:** vari ruoli di progetto dovranno essere formattati utilizzando la prima lettera maiuscola di ogni parola che non sia una preposizione (ex. *Responsabile di Progetto*)

[16] Intendendo sia tonde, che quadre o graffe

  - **Nomi dei documenti:** i nomi dei documenti dovranno essere formattati utilizzando la prima lettera maiuscola di ogni parola che non sia una preposizione (ex. *Norme di Progetto*)
  - **Nomi dei file:** si seguono le regole descritte al punto 5.2.1 a pagina 14 per quanto riguarda lo stile monospace
  - **Nomi propri:** l'utilizzo dei nomi propri dovrà rispettare la struttura "**Nome Cognome**"
  - **Nome del gruppo:** il gruppo dovrà sempre essere nominato per intero ("*Team Committed* "). A questo proposito è stato istituito il comando LaTeX `\TeamCommitted` che è già formattato secondo questa norma
  - **Nome del proponente:** ci si riferirà al proponente come *Dr. Amir Baldissera* . Anche a questo proposito è stato creato un comando LaTeX, `\propProg`
  - **Nome del progetto:** ci si riferirà al progetto come *Progetto SafetyGame*. Il comando LaTeXrelativo è `\nameproject`

- **Sigle:** sono ammesse le sigle sia dei nomi dei documenti che dei nomi delle revisioni. Dovranno sottostare alle regole per gli acronimi descritte nella sezione 5.2.1. In particolare:
  - **AR** = Analisi dei Requisiti
  - **PdP** = Piano di Progetto
  - **PdQ** = Piano di Qualifica
  - **NdP** = Norme di Progetto
  - **GL** = Glossario
  - **RR** = Revisione dei Requisiti
  - **RP** = Revisione di Progettazione
  - **RQ** = Revisione di Qualifica
  - **RA** = Revisione di Accettazione
  - **SF** = Studio di Fattibilità
  - **ST** = Specifica Tecnica

5.3 Componenti visive

5.3.1 Immagini

Tutte le immagini dovranno rigorosamente essere in formato `PNG` o `JPG`.

Ogni immagine dovrà avere una breve didascalia in cui dovrà necessariamente comparire un nome identificativo per la tracciabilità di tale immagine nel testo. A tale scopo si impone di utilizzare numeri incrementali.

5.3.2 Tabelle

Ogni tabella dovrà essere fornita di didascalia, in cui dovrà comparire un numero identificativo incrementale per la tracciabilità di tale tabella all'interno del documento[17].

5.4 Versionamento

Come già descritto nel capitolo 1 del seguente documento, al fine di uniformare lo stile di versionamento si useranno le seguenti norme sul versionamento:

- **Notazione:** per esprimere le varie versioni dei documenti si userà la seguente notazione:

  `V{X}.{Y}`

  dove:

  - **{X}:** indica il numero di uscite formali "accumulate" dal documento
  - **{Y}:** indica il numero di modifiche effettuate considerate importanti dal redattore del documento[18]

- **Utilizzo:** la notazione di versionamento dovrà sempre essere utilizzata nei seguenti casi:

  - **Citazione di un documento:** ogni volta che si cita un documento, bisognerà inserire il suo nome completo seguito dalla sua versione: `Documento di Esempio - V{X}.{Y}`
  - **Intestazione documenti:** la copertina di ogni documento dovrà avere il nome per esteso seguito dalla sua versione attuale

5.5 Formattazione documenti

Per essere considerato formattato bene, ogni documento dovrà avere:

- **Nome documento, versione e nome del team**: posizionati nel frontespizio sulla destra, seguendo lo schema "*NomeDocumento V{X}.{Y} - Team Committed*"
- **Logo del Team**: posizionato nel frontespizio sulla sinistra. Dovrà essere la versione ridotta del logo, come specificato in sezione 5.1
- **Numero di pagina:** posizionato a piè pagina al centro, espresso nel formato:

[17] É escluso il "*Diario delle modifiche*" per ovvi motivi

[18] Non viene normata la "*modifica effettuata considerata importante*" in quanto pensiamo che sia palese il suo significato

`n di tot`

Ogni documento prodotto, inoltre, prima dell'indice dovrà contenere nelle pagine di apertura:

- **Nome del documento**
- **Versione del documento**, espressa nel formato definito nella sezione 5.4
- **Informazioni generali del documento:**
  - **Nome del documento**
  - **Versione del documento**
  - **Data redazione**, ovvero data di rilascio della versione *{X}* del documento
  - **Redattori**, ovvero coloro che hanno collaborato alla redazione del documento[19]
  - **Verificatori**, ovvero i nomi dei membri del gruppo che hanno collaborato alla verifica del documento[20]
  - **Approvatori**, ovvero chi ha approvato la versione finale del documento[21]
  - **Uso**, se interno od esterno al gruppo
  - **Distribuzione**, ovvero la lista di persone che dovranno ricevere il documento[22]
- **Diario delle modifiche**, tabella che riporti le varie modifiche apportate al documento per giungere alla versione finale. Il *Diario delle modifiche* è una tabella ordinata in maniera decrescente, ovvero le prime righe contengono informazioni sulle ultime modifiche, mentre l'ultima riga conterrà le informazioni riguardanti le prime modifiche.
  Ogni riga dovrà contenere:
  - **Informazioni sulle modifiche**: breve descrizione sulle modifiche apportate al documento
  - **Autore**: dovrà contenere il nominativo di chi ha apportato la modifica in oggetto
  - **Data modifica**: dovrà indicare la data delle modifiche apportate
  - **Versione**: indicherà la versione a cui si è arrivati al momento dell'aggiornamento del *diario delle modifiche*

Per quanto riguarda le note di formattazione e di inclusione automatica dei seguenti documenti, si prega di vedere la sezione 5.1 a pagina 13

[19] Nel caso di più persone si dovrà utilizzare un elenco puntato per elencarle tutte
[20] v.d. nota 19
[21] v.d. nota 19
[22] v.d. nota19

5.6 Classificazione documenti

5.6.1 Documenti informali

Tutti i documenti che non siano stati approvati dal *Responsabile di Progetto* dovranno essere considerati dei **documenti informali**. Questi documenti saranno considerati **sempre** ad uso interno. Dovranno essere depositati nel repository dei documenti seguendo le norme descritte nella sezione 4.2.1 a pagina 11.

5.6.2 Documenti formali

I documenti formali sono tutti quei documenti che hanno raggiunto una versione pronta per il rilascio, ovvero quando il *Responsabile di Progetto* ha dato il suo assenso alla distribuzione.

Raggiunto questo stadio, il documento dovrà essere considerato per l'uso descritto nelle *informazioni generali del documento*. Questi documenti sono gli unici che possono (e devono) essere distribuiti alla propria lista di distribuzione. Il responsabile dovrà seguire quanto scritto nella sezione 4.2.1 a pagina 11 per quanto concerne la creazione del documento PDF del documento formale.

Ogni volta che si appone una modifica ad un documento già approvato (quindi dichiarato formale), si dovrà inserire tale modifica nel *diario delle modifiche*, avvisando nel frattempo il *Responsabile di Progetto* che provvederà, nel caso non siano necessarie ulteriori modifiche, a riapprovare il documento e rilasciare una nuova versione formale di esso.

5.7 Procedura di formalizzazione dei documenti

Ogni volta che i *redattori* considerano pronto un documento, dovranno avvisare il *Responsabile di Progetto* , il quale assegnerà tale documento ai *verificatori*.

Se il documento dovesse rispettare tutte le norme ivi descritte, il *Responsabile di Progetto* potrà approvare o meno il documento, oppure rimandarlo ai *redattori* per correzioni non normative.

5.8 Procedura di 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 V2.0.
I *verificatori*, una volta presa in consegna i documenti che hanno raggiunto una versione candidata al rilascio, dovranno applicare la seguente procedura:

- **Controllo ortografico:** il *verificatore* procederà al controllo ortografico del documento tramite **Aspell**, andando su *Strumenti->Correttore Ortografico*. Prima però dovrà controllare che la lingua del documento[23] sia correttamente impostata in *italiano*. Per fare ciò, dovrà andare su

[23] Poichè i documenti saranno formati da un file per ogni capitolo, il *verificatore* dovrà effettuare questa procedura per ogni singolo file

Documento->Impostazioni e, nel menù laterale, selezionare la voce *Lingua*. Una volta selezionata la voce, dovrà guardare la lingua impostata e, nel caso, correggerla in "*italiano*".

- **Controllo lessicale:** il *verificatore* procederà ad un'attenta lettura dell'intero documento alla ricerca di errori lessicali.
- **Grafici UML:** nel caso siano presenti dei grafici UML, il *verificatore* dovrà controllare che tali grafici rispettino le norme descritte in sezione 10.1 procedendo tramite attenta osservazione di essi

6 Analisi dei Requisiti

L'Analisi dei Requisiti dovrà essere redatta dagli analisti.

In tale documento dovranno comparire ordinatamente tutti i requisiti evinti dal Capitolato e/o dagli incontri effettuati con il *Proponente* / *Committente* .

Ogni requisito dovrà essere il più completo e non ambiguo possibile. Per tale motivo si richiede che per ogni requisito venga specificato:

- **Fonte:** ovvero da dove è emerso il requisito
- **Classificazione:** ovvero raggruppare i requisiti in insiemi coerenti in modo da poter definire una segnatura specifica, univoca e gerarchica secondo il seguente formalismo

  `R{X}{Y}`[24]

  dove:

  - **{X}:** indica la categoria del requisito. In particolare:
    - **F:** requisito funzionale
    - **Q:** requisito di qualità
    - **P:** requisito di prestazionale
    - **V:** requisito di vincolo
  - **{Y}:** indica la classe del requisito. In particolare:
    - **OB:** requisito obbligatorio
    - **D:** requisito desiderabile
    - **OP:** requisito opzionale
- **Descrizione grafica:** utilizzare il formalismo UML per creare diagrammi di casi d'uso che facilitino la comprensione di tali requisiti.
  I diagrammi dei casi d'uso utilizzeranno una nomenclatura simile alla precedente:

  `UC{XX}{YY}`

  dove:

  - **UC:** Use Case
  - **{XX}:** iniziali dell'attore principale del caso d'uso[25]
  - **{YY}**: numero sequenziale che contraddistingue il caso d'uso[26]

[24] Queste sigle dovranno essere seguite dal numero del requisito. Qualora si trattasse di un sottorequisito, si dovrà usare il sistema di numerazione per indici/sotto-indici

[25] Nel caso si tratti di un caso d'uso generale, **{XX}=G**

[26] Se è un'*esplosione* di un caso d'uso, sarà formato dal numero del caso d'uso che espande seguito da `.{ZZ}`

- **Descrizione didascalica:** al fine di non lasciare ambiguità ai casi d'uso, si dovranno correlare con una descrizione narrativa in cui dovranno comparire le seguenti informazioni:
  - **Attori coinvolti**
  - **Scopo e descrizione**
  - **Precondizione**
  - **Postcondizione**
  - **Flusso base degli eventi**
  - (Possibile) **Flussi alternativi**

Per ogni requisito dovrà essere inoltre definito un metodo per verificare se tale requisito, all'interno del sistema, sia stato soddisfatto o meno. Tale verifica potrà essere di tipo:

- **Statica**
- **Dinamica,** tramite test

6.1 Classificazione dei requisiti

Ogni requisito dovrà essere classificato in una di queste categorie:

- **Funzionale:** requisiti che determinano le capacità richieste al sistema
- **Non funzionale:** proprietà interne al sistema
- **Di qualità:** requisiti volti a portare valore aggiunto al sistema
- **Di interfacciamento:** requisito che svolge la funzionalità di connessione fra sistema e utente che vuole interagire con il sistema stesso
- **Di vincolo:** requisiti espressamente indicati nel capitolato d'appalto o nei verbali d'incontro con il *Proponente* o *Committente*

Inoltre ogni requisito dovrà appartenere ad una classe di requisiti:

- **Obbligatorio:** requisito da considerarsi irrinunciabile per il cliente. Senza di esso l'applicazione è da considerarsi **non soddisfacente** per il cliente
- **Desiderabile:** requisiti non strettamente necessari, ma che apportano valore aggiunto importante al prodotto
- **Opzionale:** requisito relativamente utile/importante o che potrebbe essere soggetto di ulteriore contrattazione

7 Progettazione

Durante la fase di progettazione, i progettisti dovranno attenersi alle seguenti specifiche.

7.1 Diagrammi

Si dovrà utilizzare il linguaggio UML per definire:

- **Diagrammi dei package:** dovranno essere presenti per l'architettura generale. Sarà fondamentale definire in maniera univoca i vari package, in modo da poter distribuire a più codificatori, in fase di *codifica*, package differenti che interagiscono fra loro
- **Diagrammi delle attività:** dovranno essere presenti per l'architettura generale e di dettaglio, qualora il comportamento dell'architettura di dettaglio non sia stato esplicato precedentemente. Si dovrà presentare con questo diagramma un generico flusso di esecuzione dell'intera applicazione. Qualora l'intero flusso dovesse risultate troppo approssimativo o complicato per un solo diagramma, si dovranno creare dei sotto-diagrammi per ampliare o suddividere il diagramma generale

7.2 Design Pattern

Si dovrà ricorrere alla seguente specifica per descrivere i vari design pattern utilizzati:

- **Descrizione generale:** descrivere brevemente la struttura generale del design pattern
- **Motivazione:** descrivere il motivo della scelta di tale design pattern
- **Contesto applicativo:** elencare i contesti ove il design pattern è stato applicato

7.3 Classi di verifica

Si dovranno sviluppare, quando possibile e sopratutto per le classi più generali, delle classi fittizie da utilizzare durante le fasi di verifica.

7.4 Stile di progettazione

Per semplificare la lettura degli schemi e per la futura fase di verifica, sarà necessario prestare attenzione a:

- **Ricorsione:** si cercherà di evitare il più possibile l'utilizzo della ricorsione. Qualora il suo uso si dovesse rivelare strettamente necessario, si dovrà fornire adeguata dimostrazione di terminazione; nel caso la dimostrazione

sia adeguata e corretta, sarà altresì necessario stimare l'occupazione di memoria indotta dalla ricorsione: se l'occupazione di memoria sarà eccessiva, la ricorsione dovrà essere eliminata, anche se formalmente corretta

- **Concorrenza:** nel caso di utilizzo di flussi di esecuzione concorrenti, sarà necessario fornire, oltre ai diagrammi di flusso, anche una stima di risorse necessarie all'implementazione. Se i benefici ricavati dalla concorrenza[27] non saranno equivalenti o superiori alle risorse utilizzate, la concorrenza sarà eliminata
- **Annidamento di chiamate:** non dovranno esserci chiamate annidate con una profondità maggiore di 10(dieci)
- **Flussi condizionali:** per la chiarezza e comprensione del futuro codice e per migliorare la verifica di esso, qualora vengano utilizzati costrutti condizionali[28], l'annidamento di tali costrutti non potrà essere maggiore di 5(cinque). Questo parametro potrà essere aumentato previo accordo con il Responsabile di Progetto, nei casi in cui si renda necessario
- **Numero di parametri:** i metodi creati non potranno avere più di 10(dieci) parametri. Questo limite non si applica al numero di variabili interne alle singole classi

7.5 Tracciamento

Dovrà essere presente in forma tabellare il tracciamento di ogni requisito con l'entità che ne assicura la soddisfacibilità e viceversa.

7.6 Architettura generale

Il documento relativo all'architettura generale dovrà contenere al suo interno:

- **Design pattern:** trattazione dei vari design pattern utilizzati come specificato in sezione 7.2
- **Struttura generale dell'architettura**
  - Diagramma di package con una sintetica spiegazione
- **Struttura generale delle sottoparti dell'architettura**
  1. Diagramma delle classi
  2. Descrizione del diagramma delle classi ove, per ogni classe, dovrà comparire:
     (a) Nome completo comprensivo della gerarchia dei package

[27] Nel caso quindi non sia fondamentale per il corretto funzionamento dell'applicazione
[28] Per esempio costrutti del tipo `if-then-else`

   (b) Breve descrizione interna e motivo di esistenza
   (c) Funzionalità offerte
   (d) Utilizzatori ipotetici
3. Diagramma di attività della specifica sottoparte[29]
4. Diagramma di sequenza[30]

7.7 Architettura di dettaglio

Da definire.

[29] Si può omettere tale diagramma qualora la sua definizione risulti talmente banale da risultare immediata conseguenza delle descrizioni e dei diagrammi precedenti
[30] V.d. nota precedente

8 Codifica

Si elencano ora una serie di convenzioni per la struttura dei file sorgenti prodotti.

8.1 Intestazione file

Ogni file di codice sorgente dovrà iniziare con un'intestazione che dovrà rispettare il seguente schema:

```
/*
 * Name: {Nome del file}
 * Package: {Package di appartenenza}
 * Author: {Autore del file}
 * Date: {Data di approvazione del file}
 * Version: {Versione del file}
 *
 * Changes:
 * +------------+--------------+-------------------------
 * |    Date    | Programmer   | Changes
 * +------------+--------------+-------------------------
 * | AAAAMMGG   | NomeCognome  | - [label]method1
 * |            |              | - [label]method2
 * |            |              | - ....
 * +------------+--------------+-------------------------
 *
*/
```

dove:

- **Name:** nome del file comprensivo di estensione
- **Package:** nome completo della gerarchia del package
- **Author:** deve contenere i nomi di tutti coloro che hanno creato e modificato il file. Non devono essere inclusi coloro che hanno verificato e approvato il file
- **Date:** indica la data di approvazione del file
- **Version:** indica la versione attuale del file
- **Changes:** rappresenta la tabella di avanzamento del file. Per convenzione:
  - **NomeCognome:** dovrà contenere le prime 4 lettere del nome seguite dalle prime 4 lettere del cognome

– **Changes:** rappresenta la lista dei cambiamenti. Per ogni riga dovrà esserci un solo cambiamento[31]. *Label* potrà essere:
[+]: indica la creazione del metodo
[-]: indica l'eliminazione del metodo
[*]: indica la modifica del metodo

8.2 Convenzioni di codifica

Per rendere il codice leggibile, comprensibile e quindi, il più manutenibile possibile, si stabilisce che i programmatori debbano seguire la Java Code Convetion, il cui documento è reperibile all'indirizzo:

`http://java.sun.com/docs/codeconv/CodeConventions.pdf`

Di seguito si fa un sommario delle regole più frequentemente utilizzate:

8.2.1 Struttura interna delle classi

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

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

8.2.2 Struttura del codice

- **Dichiarazione di variabili:** al massimo una dichiarazione di variabile per linea
- **Blocchi di inizializzazione:** dichiarare tutte le variabili utilizzate in un blocco all'inizio del blocco stesso
- **Nomenclatura variabili:** non utilizzare variabili con lo stesso nome
- **Separazione dei metodi:** separare i metodi con una linea vuota
- **Separazione blocchi di codice:** separare con una linea vuota frammenti di codice logicamente indipendenti
- **Enfatizzazione parole chiave:** separare con uno spazio le parole chiave e le parentesi[32]

[31] Nel caso di più cambiamenti usare più righe indicando solo una volta la data e NomeCognome
[32] Fra cui, ma non solo, i costrutti `if`, `for`, etc...

- **Parentesi graffe:** le parentesi graffe che aprono un blocco di codice devono essere nella stessa linea della parola chiave a cui appartengono, mentre le parentesi graffe che chiudono il blocco devono essere scritte in una nuova linea. Ad esempio:

```
while (condition) {
        statement1;
        statementN;
}
```

8.2.3 Convenzione sui nomi

- **Classi:** devono essere sostantivi la cui prima lettera deve essere maiuscola
- **Interfacce:** stessa nomenclatura delle classi
- **Metodi:** devono essere dei verbi la cui prima lettera deve essere minuscola. Nel caso siano composti da più parole, l'iniziale di ogni parola deve essere maiuscola
- **Variabili:** devono essere dei nomi brevi ed evocativi, la cui prima lettere deve essere minuscola. Nel caso siano composte da più parole, l'iniziale di ogni parola deve essere maiuscola
- **Costanti:** devono essere scritte in maiuscolo. Nel caso siano composte da più parole, si deve unire dal carattere underscore

8.3 Procedura di verifica

Per rendere uniforme la procedura di verifica, vengono qui elencate le norme che regolano la verifica del codice prodotto[33]

8.3.1 Verifica statica

- **Analisi del flusso di controllo**: si accerta che il codice segua il flusso aspettato, che non si possa entrare in porzioni di codice che possano non terminare, che non esista codice non raggiungibile
- **Analisi del flusso dei dati**: si accerta che il software non acceda mai a variabili non inizializzate o scriva inutilmente più volte prima di usare una variabile
- **Analisi del flusso di informazione**: verifica che gli input e gli output di ogni unità di codice o di più unità rientrino nelle specifiche del programma

[33] Si rimanda al `Piano di Qualifica - V2.0` che specifica in maniera dettagliata le tecniche e le modalità con cui verranno condotte le attività di verifica e validazione durante l'intero sviluppo del progetto

8.3.2 Verifica dinamica

- **Test di unità**: test che si effettuano per ogni unità del software con il massimo grado di parallelismo
- **Test di integrazione:** verifica dei componenti formati dall'integrazione delle varie unità che hanno passato il test di unità
- **Test di sistema e di collaudo**: verifica che il sistema in cui andrà installato il software rispetti i requisiti richiesti, o che il software riesca ad adattarsi correttamente al contesto dell'azienda proponente. Il collaudo sarà sul software installato, finito il quale avverrà il rilascio del prodotto
- **Test di regressione**: nel caso di una modifica ad un singolo componente, andranno effettuati nuovamente tutti i test di unità e, se necessario, di integrazione riferiti a quel componente

9 Glossario

Il Glossario dovrà essere formattato come definito in sezione 2.1 a pagina 7.

Il glossario sarà unico e riassuntivo per tutti i documenti.

Dovrà riportare in ordine lessicografico le definizioni delle parole che possono generare confusione o ambiguità all'interno dei documenti e che in essi appariranno sottolineate.

9.1 Inserimento vocaboli

Data l'universalità del glossario rispetto a tutti i documenti redatti dal gruppo *Team Committed* , viene nominato *Responsabile del glossario* il *Responsabile di Progetto* .

Per chiedere l'inserimento di un nuovo vocabolo all'interno del glossario, si dovrà seguire la procedura descritta in sezione 2.1 a pagina 7

10 Norme generiche

10.1 Grafici UML

Di seguito elenchiamo una serie di convenzioni adottate per tutti i grafici UML.

- **Standard:** lo standard adottato per tutti i grafici UML è lo standard *2.0*
- **Lingua:** la lingua di adozione per i grafici è l'*italiano*

10.1.1 Diagrammi Use Case

- **Use case:** ogni *use case* dovrà essere associato ad una particolare funzionalità del programma
- **Attori:** gli attori dovranno essere scritti con l'iniziale maiuscola e, nel caso di composizione di più parole, dovranno essere concatenate da un carattere di spaziatura, seguito da una maiuscola. Per gli acronimi vale quanto scritto nella sezione 5.2 a pagina 14
- **Funzioni:** tutte le funzioni dovranno essere scritte in maniera *atomica*[34], senza però lasciare spazio ad interpretazione al nome della funzione. Inoltre i verbi, se presenti, dovranno essere sostantivatizzati
- **Nomenclatura:** il nome di ogni *Use Case* dovrà essere un acronimo formato da:

  `UC{XX}{YY}`

  dove:

  - **UC:** Use Case
  - **{XX}:** iniziali dell'attore principale del caso d'uso[35]
  - **{YY}**: numero sequenziale che contraddistingue il caso d'uso[36]

10.2 Gestione delle attività

Per quanto riguarda la gestione delle attività e delle problematiche correlate al progetto, abbiamo deciso di adottare il **servizio di ticketing** messo a disposizione da *Github*[37].

[34] Ovvero inserendo il minor numero di termini possibili
[35] Nel caso in cui si parli di "Ambito Generale", **XX** = G
[36] Se è un'*esplosione* di un caso d'uso, sarà formato dal numero del caso d'uso che espande seguito da `.{ZZ}`
[37] Come scritto nella sezione 2.1 a pagina 7, questo servizio verrà usato **anche** per le comunicazioni fra membri singoli

Questo servizio è ottimale in quanto, assieme al servizio di milestone, permette di avere sott'occhio molto facilmente tutto lo stato dell'avanzamento dei lavori.

Gli strumenti di gestione delle attività dovranno essere utilizzati seguendo il seguente protocollo:

- **Creazione milestone:** il *Responsabile di Progetto* dovrà creare una **milestone** per la successiva revisione a cui il *Team Committed* ha intenzione di partecipare. Se ne potrà vedere lo stato di avanzamento a seconda nel numero di ticket completati rispetto al numero di ticket aperti.
  Per fare ciò si dovrà andare nel repository, andare su *Issue->Milestones->Create a new milestone*
- **Creazione ticket:** il *Responsabile di Progetto* dovrà creare un **ticket** per ogni compito. Tale compito dovrà essere assegnato ad un membro del *Team Committed* . Altri casi previsti per la creazione di un *ticket* sono le segnalazioni di *Anomalie e Discrepanze* da parte di un qualsiasi membro del gruppo come indicato nel documento `Piano di Qualifica - V1.0`. La creazione di un *ticket* va fatta definendo obbligatoriamente i seguenti campi:
  - **Title:** nome del compito assegnato
  - **Assegnee:** colui a cui è stato affidato il *ticket*
  - **Milestone:** la *milestone* per la quale il compito dovrà essere terminato
  - **Label**: le label verranno impostate a seconda di precisi argomenti nella fase di progettazione e alla creazione del ticket andrà scelta la label più appropriata
  - **Text:** il testo riassuntivo del compito
- **Esecuzione compito e risoluzione di un ticket:** ogni membro del gruppo dovrà visionare i ticket a lui assegnati e inserire un nuovo commento per ogni aggiornamento sullo stato del ticket. Una volta completato il compito o risolta l'*Anomalia* descritti nel ticket, dovrà aggiungere l'apposita label "*fixed*" e, se lo ritiene necessario, menzionare il ticket dal relativo commit. Sarà compito del Responsabile di Progetto confermare l'effettiva chiusura del ticket solo se il compito è stato eseguito o la modifica al codice ha risolto l'*anomalia*. L'utilizzo della label aiuta il *Responsabile di Progetto* a individuare i ticket presumibilmente da chiudere.
- **Ticket di verifica:** i **ticket di verifica** verranno creati secondo quanto scritto precedentemente
- **Chiusura milestone:** una *milestone* verrà considerata conclusa una volta che tutti i ticket creati sono stati chiusi. Alla chiusura di una *milestone*, il responsabile dovrà ripartire dal primo punto di questa procedura

11 Ambiente di progetto

Descriveremo ora l'ambiente che il gruppo *Team Committed* ha deciso di utilizzare per lo svolgimento dell'intero progetto.

Poiché il progetto *C03* richiede espressamente delle applicazioni multi-piattaforma, abbiamo deciso di uniformare dove possibile l'ambiente di sviluppo con applicazioni multipiattaforma[38].

11.1 Sistema operativo

Tutte le attività legate allo sviluppo del progetto verranno svolte su ambienti **Windows 7**, **Mac OsX** e **GNU/Linux**[39]

11.2 Ambiente documentale

11.2.1 Scrittura documenti

La scrittura dei documenti avverrà per mezzo di **LaTeX**[40] e come editor utilizzeremo **LyX** ($\geqslant$ 2.0.2), editor WYSIWYM[41] che permette una rapida stesura di documenti avendo una minima, se non nulla, conoscenza di *LaTeX*.

Per la compilazione di ogni sorgente ci si affiderà alla libreria `pdflatex`, libreria *LaTeX* che compilerà i file prodotti con *LyX* in formato `PDF`; la compilazione avverrà direttamente all'interno di *LyX*, tramite *File -> Esporta -> PDF(pdflatex)*

11.2.2 Verifica ortografica

Per la verifica ortografica dei documenti scritti in LaTeXsi userà il plugin per *LyX* **Aspell** ($\geqslant$ 0.60.6). L'utilizzo di Aspell avverrà tramite *LyX*, tramite *Strumenti -> Correttore ortografico*.

11.2.3 Pianificazione

Per quanto riguarda la pianificazione delle attività legate allo sviluppo del progetto e la gestione delle risorse verrà utilizzato lo strumento **Microsoft Project 2010**, strumento fornito dal servizio *MSDNAA* di Microsoft in collaborazione con il Dipartimento di Matematica Pura ed Applicata dell'Università degli Studi di Padova.

[38] Questa, assieme al fatto che abbiamo cercato di utilizzare il maggior numero possibili di applicazioni rilasciate in licenza *open-source* e/o *free software*, è una motivazione ricorrente sulla scelta delle applicazioni sotto esposte. Per questo motivo le motivazioni sopra esposte verranno omesse

[39] Con GNU/Linux intendiamo Ubuntu 10.04

[40] `http://www.latex-project.org`

[41] What You See Is What You Mean

Abbiamo scelto questo programma perché la controparte open-source, *Gantt-Project*, non è funzionale quanto *Project*, consci del fatto che il programma funzioni solo su ambiente Windows. Per questo motivo l'amministratore ha predisposto una macchina virtuale Windows XP (anch'esso fornito dal *MSDNAA*) con installato il programma.

11.2.4 Grafici UML

Per la definizione dei grafici UML è stato deciso di utilizzare **Bouml** ($\geqslant$ 4.23) per via della facilità di utilizzo e, sopratutto, per il fatto che il programma aderisce agli standard di *UML2*

11.2.5 Documentazione semi-automatica

Lo strumento di documentazione semi-automatica impiegato sarà **Javadoc** ($\geqslant$ 1.5), funzionalità di Java che permette di creare tutta la documentazione relativa ad una porzione di codice a partire dai commenti.

11.3 Ambiente di sviluppo

11.3.1 Strumento di versionamento

Come strumento di versionamento si è deciso di utilizzare **git**.

git è uno strumento di versionamento veloce e di facile apprendimento che rappresenta uno dei migliori strumenti attualmente esistenti. Per lo sviluppo collaborativo abbiamo deciso di appoggiarci al servizio **Github**[42] che fornisce non solo un repository *git*, ma anche strumenti utili alla collaborazione fra più persone, come il servizio di *ticket*, *wiki* e *milestone*.

Per quanto riguarda l'uso di *git* sui computer di sviluppo, si è deciso l'uso della versione ufficiale rilasciata dal team di sviluppo di *git* ($\geqslant$ 1.7.8)

11.3.2 Ambiente di codifica

Per la scrittura del codice abbiamo deciso di impiegare gli IDE **Eclipse** ($\geqslant$ 3.7.1) e **BlueJ** ($\geqslant$ 3.0.6) per la codifica in *Java* e per lo sviluppo di applicazioni *Android*, mentre **Geany** ($\geqslant$ 0.21) per la codifica *HTML*

11.3.3 Front-end grafico

Oltre alle **API native** di *Android*, abbiamo deciso di utilizzare **Swing** e **AWT** per il front-end grafico dell'applicazione *Java*.

11.3.4 Ambiente mobile

Per sviluppare l'applicazione mobile useremo il **Android SDK**, con il plugin per *Eclipse* **ADT** ($\geqslant$ 15.0.1)

[42] `http://www.github.com`

11.4 Ambiente di verifica e validazione

Di seguito vengono elencati gli strumenti scelti per le verifiche e la validazione[43].

11.4.1 Analisi statica

L'analisi statica di tutto il codice verrà effettuata con **FindBugs** ($\geqslant$ 2.0.0), mentre l'analisi metrica del codice verrà effettuata con **Metrics** ($\geqslant$ 1.3.6), plugin per *Eclipse*

11.4.2 Test

Per effettuare test useremo il plugin di *Eclipse* **JUnit** ($\geqslant$ 4.10) e **EclEmma** ($\geqslant$ 1.5.3). Per quanto riguarda i testi di carico sul database useremo **ApacheBench** ($\geqslant$ 2.0) tramite linea di comando, mentre per i test su browser useremo il plug-in per *Mozilla Firefox* **Selenium IDE** ($\geqslant$ 1.4.1).

11.4.3 Analisi dinamica

L'analisi dinamica verrà effettuata con il plug-in per *Mozilla Firefox* **Firebug** ($\geqslant$ 1.8.3). Per i test di velocità dell'applicazione web useremo il plugin per *Chrome* **SpeedTracer** ($\geqslant$ 2.4).

11.4.4 Validazione

La validazione del codice *HTML* e *CSS* dell'applicazione da noi sviluppata verrà fatta tramite il servizio **W3C Validator**[44] del *W3C*.

[43] Per maggiori dettagli si rimanda al documento Piano di Qualifica - V1.0
[44] `http://validator.w3.org/`