Norme

Release 1.0

December 10, 2009
Firenze

Approvazione, redazione, lista distribuzione

approvato da	**il giorno**	**firma**
Marco Tinacci	December 10, 2009	

redatto da	**il giorno**	**firma**
Massimo Nocentini	December 10, 2009	

distribuito a	**il giorno**	**firma**
Francesco Calabri	December 10, 2009	
Manuele Paulantonio	December 10, 2009	
Daniele Poggi	December 10, 2009	
Massimo Nocentini	December 10, 2009	
Niccoló Rogai	December 10, 2009	
Marco Tinacci	December 10, 2009	

Contents

0.1 Introduzione

Le norme di progetto sono divise in categorie (comunicazioni interne, documentazione, strumenti) e sotto categorie.

Nelle norme si avanza di major release sull'aggiunta di categorie significative mentre si avanza di minor per le modifiche e gli aggiornamenti minori di categorie e sotto categorie.

0.2 Struttura del documento

Chapter 1

Gestione del progetto

1.1 Modello di processo

Come modello di riferimento usiamo il modello agile **ICONIX**, eventualmente adattando caso per caso le fasi principali. La fase di analisi dei requisiti prevede la produzione dei documenti di domain model, use cases, metriche e mokup. La fase di progettazione del sistema prevede la produzione dei diagrammi di sequenza e di classe, con descrizioni delle interfacce e degli algoritmi non banali. La fase di progettazione delle prove prevede la produzione dei robustness diagrams al fine di produrre un documento contenente i test che si vogliono eseguire.

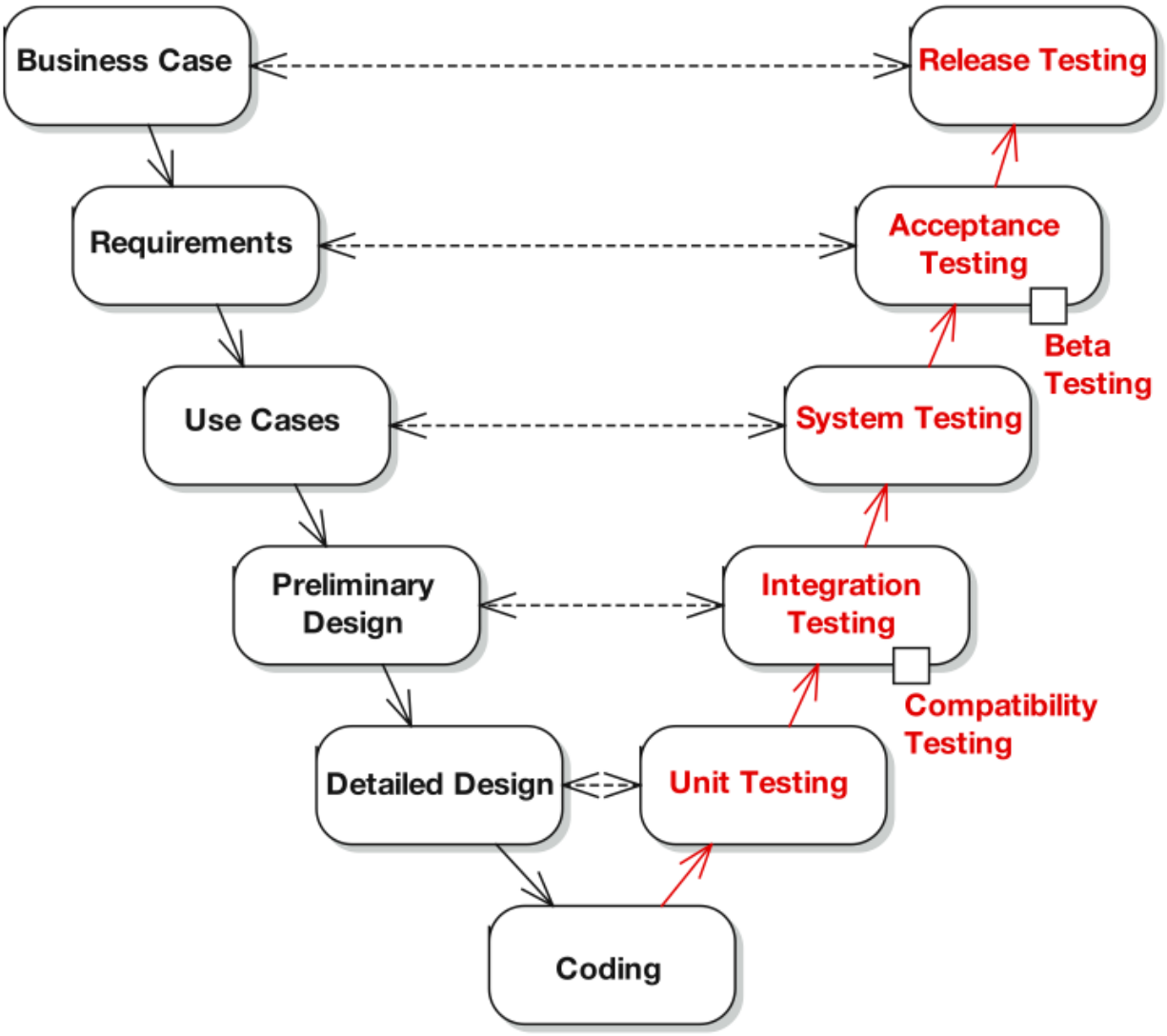

Figure 1.1: Modello a “V” che descrive le varie fasi di produzione e di testing

Chapter 2

Comunicazioni interne

2.1 Gruppo di discussione

Ogni comunicazione rilevante riguardo al progetto deve essere comunicata sul gruppo di discussione all'indirizzo web http://groups.google.it/group/kiwitp . I componenti del gruppo sono tenuti a controllare frequentemente le comunicazioni, dopo 24 ore dalla pubblicazione della comunicazione si presuppone che questa sia stata letta. Se la scadenza ricade in un giorno festivo si assume che la comunicazione sia stata letta dopo il termine del primo giorno lavorativo successivo. Eventuali ritardi o impossibilità devono essere comunicati sempre sul gruppo con almeno 24 ore di anticipo sull'evento interessato. Etichettare il titolo della discussione coi nomi delle persone interessate. Se la comunicazione è diretta a tutti i componenti del gruppo si può omettere i destinatari.

Chapter 3

Documentazione

3.1 Struttura delle directory e formato file

Tutti i documenti risiedono in formato digitale (.tex) nella cartella *"/documentazione_progetto"* del server SVN del progetto (specificato negli strumenti usati). Definiamo la relazione fra tipi di modello e posizione nel file system:

diario qui e' presente il file *[Kiwi-MGC]diario.tex* che compone le informazioni divise nelle successive directory. Inoltre viene committato anche il relativo *[Kiwi-MGC]diario.pdf*, generato a partire dal file .tex, ogni volta che il file sorgente viene committato.

archivio

consegne

verbali i file contenenti

riunioni_committente

collaudo

incontri_analisi

revisioni_congiunte

riunioni_interne

collettive

parziali

riunioni_SQ

norme

comunicazioni_interne

documentazione

strumenti

images

3.1.1 Sintassi valide esclusi i verbali

I nomi dei file che non appartengono alle foglie della precedente struttura devono rispettare la seguente sintassi:
<nome file> ::= [<team>-<proj>]<file>.<ext>
<file> ::= organigramma — diario — norme — requisiti — offerta — disegno_sistema — piano_prove — prova_<data>
<ext> ::= tex
<data> ::= AAAAMMGG (anno su quattro cifre, mese e giorno su due cifre per far coincidere l'ordinsamento alfabetico con l'ordinamento cronologico)
<team> ::= Kiwi
<proj> ::= MGC

3.1.2 Sintassi per i verbali

nome della cartella: verbale_<data>
nomi per i file:
[<team>-<proj>]verbale_body_<data>.tex
[<team>-<proj>]verbale_<data>.pdf
[<team>-<proj>]verbale_<data>.tex

3.2 Stringa di release

Le revisioni dei documenti versionati devono essere indicate con la dicitura **R.r** dove *R* indica la major release mentre *r* indica le minor release. Quando si parla di versione si intende quella appena specificata. Per risalire al numero di versione registrato nel server SVN si dovrà fare riferimento al diario di progetto. La prima release è fissata come **0.0**. Si ha una minor release di un prodotto quando si risolvono alcuni problemi significativi noti nell'ultima versione, e si scrive che migliorie si sono fatte nel messaggio di commit. Si ha una major release quando si risolvono tutti i problemi significativi noti. Ogni documento unico o major release deve essere stampato e firmato da tutti i responsabili seguendo la traccia della matrice di responsabilità. Inoltre ogni documento deve essere identificato dalla data di redazione e quella di accettazione di ogni singolo responsabile.

3.3 Dati comuni

- Ogni documento deve contenere la data di redazione e la data di presa visione di ogni persona coinvolta.
- Il formato della data deve essere AAAAMMGG nei nomi dei file (per esempio nei verbali) per facilitare l'ordinamento cronologico.
- All'interno dei documenti il formato delle date deve essere GG/MM/AAAA.
- I documenti stampati di più pagine devono essere spillati.

3.4 Diario

- Il diario dovrà contenere tutte le attività di archivio e di consegna in ordine per data in modo che i primi eventi notificati siano quelli più recenti.
- Per ogni prodotto o documento deve essere seguita la struttura:
  - *Documento non versionato*: titolo, data, ricevuto da persona, ruolo.
  - *Documento versionato*: titolo, data, versione, revision, ricevuto da persona, ruolo.
  - *Documento versionato fuori struttura*: titolo, path del repository, data, versione, revisione, ricevuto da persona, ruolo.
  - *Prodotto*: nome, data, versione, revision, ricevuto da persona, ruolo.

  La data segnata sulle voci del diario è intesa come il giorno in cui il documento viene reso disponibile ai diretti interessati e non il giorno del particolare evento al quale il documento è riferito. Se per esempio la redazione di un verbale di una riunione in data A avviene in data B, il nome del documento deve contenere la data A mentre la voce del diario deve contenere la data di distribuzione su repository B.
- Se ci sono più rilasci in una stessa data di un documento versionato, si prende la release più aggiornata del giorno.
- Il diario avanza di major quando viene stampato e approvato, per le modifiche e gli aggiornamenti si avanza di minor.

Chapter 4

Strumenti

4.1 IDE

L'ambiente di sviluppo utilizzato è Eclipse PDT, provvisto del plugin di subversion.

4.2 Versioning

Il server SVN utilizzato è quello di Google Code,
all'indirizzo http://code.google.com/p/mangodiagrams/.

Quando si modifica un file versionato e le modifiche comportano un incremento della release, allora nel commento del commit si deve specificare il pattern:

```
release x.y of <versioned document name>, <other commit comments>
```

4.3 Project Management

L'applicazione per la gestione del progetto è PMango "http://pmango.cli.di.unipi.it/" e vi si accede con lo username "TP2xxxxx" (xxxxx = prime cinque lettere del cognome) mentre la password preimpostata è yyyzz (yyy = prime tre lettere del nome di battesimo, zz = ultime due cifre della matricola) I log delle ore registrate su PMango devono essere accompagnati da una descrizione sintetica del lavoro eseguito.

4.4 Document Editing

La documentazione del progetto deve essere scritta in latex e compilata in file PDF con il comando PDFlatex. La strumentazione latex usata può essere Texlive-2007 o superiori. In alternativa i documenti possono essere .odt scritti in Openoffice ed esportati in file PDF.

4.5 Diagrams

Per la creazione di diagrammi UML (casi d'uso, diagrammi di sequenza, diagrammi delle classi) utilizziamo la versione Community del software Jude alla release 5.5.2, i file .jude definiti e le rispettive esportazioni .png devono essere salvati nelle cartelle contenenti il documento a cui sono associate.

Chapter 5

Sorgenti

5.1 Files

I file sorgenti del modulo sono tutti contenuti nella cartella "/src/". La struttura di questa cartella ricalca quella descritta nel documento di disegno del sistema.

- commons
  - Commons.php
  - GanttTab.php
  - WBSTab.php
  - TaskNetworkTab.php
- useroptionschoice
  - UserOptionsChoice.php
  - OptionName.php
- reporthandler
  - ReportHandler.php
  - Report.php
- taskdatatree
  - TaskDataTreeGenerator.php
  - TaskDataTree.php
  - TaskData.php
  - Task.php

- chartgenerator
  - ChartGenerator.php
  - GanttChartGenerator.php
  - WBSChartGenerator.php
  - TaskNetworkChartGenerator.php
- gifarea
  - GifArea.php
  - GifTaskBox.php
  - GifGanttBox.php
  - GifProgressBar.php
  - GifBox.php
  - GifLabel.php
  - DrawHelper.php

5.2 Regole Sintattiche

La sintassi dei nomi segue lo stile java ed è la seguente:

- cartelle, package: lower case
- classi, files: upper camel case
- metodi e variabili: lower camel case

Per quanto riguarda i commenti, il programmatore è tenuto a commentare ogni metodo, ogni classe ed ogni variabile. Le uniche eccezioni sono i casi banali, cioè i metodi get e set con struttura tradizionale e le variabili che hanno un tempo di vita minore di quello dell'intera classe e che non sono globali. Si lascia a discrezione del programmatore il commentare le sezioni di codice difficilmente comprensibili. In generale i commenti devono precedere la dichiarazione del soggetto commentato e in particolare devono essere rispettate le seguenti regole:

- classe: commento di tipo multilinea $/* < comments > */$ contenente una descrizione generale delle funzioni che offre la classe.

- metodo: commento di tipo multilinea $/* < comments > */$ contenente la descrizione di ogni parametro in ingresso, del valore di ritorno, dei side effects, dell'algoritmo usato e se necessaria una descrizione informale. La descrizione di un parametro deve essere preceduta da $@param$, quella del valore di ritorno da $@return$ ed eventuali riferimenti da $@see$.
- variabili: commento di tipo multilinea $/* < comments > */$ contenente la descrizione di cosa rappresenta la variabile.
- altro: i commenti che non rientrano nelle categorie precedenti, come le spiegazioni delle parti di codice non facilmente comprensibili, vanno fatti con la notazione $// < comment >$ su riga singola, in modo che possano essere distinti dagli altri tipi di commenti.