Piano dei tests

Release 1.1

December 22, 2009
Firenze

Approvazione, redazione, lista distribuzione

approvato da	**il giorno**	**firma**
Marco Tinacci	December 22, 2009	

redatto da	**il giorno**	**firma**
Manuele Paulantonio	December 22, 2009	
Daniele Poggi	December 22, 2009	
Massimo Nocentini	December 22, 2009	

distribuito a	**il giorno**	**firma**
Francesco Calabri	December 22, 2009	
Niccoló Rogai	December 22, 2009	
Marco Tinacci	December 22, 2009	

Contents

Part I

Planning

Chapter 1

Master Plan

1.1 Scope

Questo è il *Master Plan* per il progetto **PMango**. Questo piano considera solo gli elementi software relativi alle aggiunte/modifiche descritte nel documento di analisi.

L'obiettivo primario di questo piano di test è quello di assicurare che la nuova versione di **PMango** offrirà lo stesso livello di informazioni e dettagli reso disponibile della versione corrente e aggiungerà tutte quelle informazioni necessarie per raggiungere gli obiettivi modellati dal processo di analisi.

Il progetto avrà tre livelli di testing i cui dettagli verranno definiti nella sezione 1.7.1 e negli specifici *level plan*.

Il quanto temporale stimato per questo progetto è molto compatto, quindi *ogni* ritardo nella fase di progettazione, sviluppo, installazione e verifica possono avere effetti significati sul deploy finale.

1.2 Struttura del documento

Il documento del piano delle prove viene diviso in tre macro parti:

planning vengono identificati gli aspetti necessari per arrivare alla definizione della specifica dei singoli test. Si identificano tutte le funzionalità (sia *technical side* che *customer side*), le risorse necessarie per eseguire il processo di test (soprattutto software in quanto non esistono procedure che richiedono grande potenza di elementi hardware), la strategia con cui si vuole portare avanti il progetto, i criteri di *success, failure* per valutare l'esecuzione di un test e la divisione in livelli logici (*fasi*) dell'intero processo.

design specification per ogni insieme di funzionalità catturate nella parte *use cases* del documento di analisi, si crea un capitolo di design nel quale si specifica quali use case si vogliono esercitare, si raffina la strategia espressa nella parte di *planning* e si identificano l'insieme di documenti *case* che effettivamente implementano i test necessari per esercitare gli use cases scelti.

case specification ogni documento di questa parte mira ad esercitare un *singolo* use case (o funzionalità): si identificano l'insieme di triple *(input, environment, output)* che permettono di costruire un modello della batteria di test che verrà applicata sullo use case.

Nella prima fase di stesura, non avendo ancora la possibilità di riferirci ai sorgenti e quindi testare effettivamente il comportamento della funzionalità (non possiamo creare dei task di prova con criteri atti a testare il codice scritto), la componente *input* di ogni tripla contiene solo la quantità di task del *testing project* che verranno creati.

Quando inizieremo la parte di codifica, la componente *input* conterrà l'identificatore dei tasks che sono stati creati nel *testing project* per esercitare la tripla.

Possiamo vedere i livelli sopra descritti come un test a tutti gli effetti, e attribuire ad ognuno di essi un risultato, codificato in base ai *pass/fail criteria*. Per dare ogni singola valutazione, possiamo usare il concetto della composizione di documenti in modo da soddisfare queste relazioni

$$panning \sqsubset desing_{specification} \sqsubset case_{specification}$$

La relazione $\sqsubset$ definisce che nella coppia $a \sqsubset b$ la valutazione di a dipende da alcune valutazioni di b. Nella nostra catena abbiamo che il risultato atomico (non composto) dell'esecuzione di un test viene espresso dai relativi documenti *case*, mentre gli altri documenti si limitano a comporre questi risultati per esprimere a loro volta il loro risultato (il concetto è simile al *bubble up* dei risultati dai case dettagliati per tuple fino al piano totale).

1.3 Elementi software sottoposti a test

La seguente lista comprende oggetti software *technical side*.

- la classe *ChartGenerator* e le sue specializzazioni, pacch etto *Chart*, definite nella sezione **1.2** del documento *disegno del sistema*
- la classe *Commons* e le sue specializzazioni, pacchetto *commons*, definite nella sezione **1.1** del documento *disegno del sistema*

1.4 Funzionalità che verranno testate

La seguente lista comprende oggetti software *customer side*.

- processo per la generazione di *Gantt chart*
- processo per la generazione di *WBS chart*
- processo per la generazione di *Task Network chart*
- interfaccia grafica per la selezione delle nuove *UserOption* per ogni *Chart*
- aggiunta di ogni *Chart* alla sezione di reportistica

1.5 Software Risk Issues

Ci sono alcuni punti che ci portano a definire questa sezione:

- Reverse engineering di codice sorgente esistente, documentato nei sorgenti ma non ha documenti ufficiali (apparte le tesi) che descrivino in modo chiaro la struttura statica e dinamica di tutto il lavoro esistente.
- Uso di librerie esterne per la generazione delle immagini e dei documenti pdf.

1.6 Strategia

1.6.1 Testing machines

Identifichiamo due classi di macchine sulle quali viene condotto il processo di testing:

- *develop machine* macchine dove avviene sia lo sviluppo del codice che parte del testing.
- *acceptance machine* macchine (molto probabilmente unica, a meno di guasti) dove avviene solo il processo di testing.

1.6.2 Testing projects

testing project Costruiamo un istanza di progetto contenente un insieme di tasks, con le relative dipendenze sia di composizione che di "finish to start", necessari per esercitare ogni tripla definita nella parte III di ogni *case specification*.

Questo progetto conterrà tasks costruiti in modo da catturare la maggior parte dei contesti che potrebbero verificarsi nell'utilizzo a regime dell'applicazione:

- task con informazioni coerenti e ben formate, per avere un contesto in cui le informazioni appartengono al dominio e dovrebbero avere un output corretto

- task con informazioni non coerenti (avendo differenza fra le *planned, actual* data per esempio), per avere un contesto in cui l'applicazione dovrà generare la notazione definita per i contesti anomali
- relazioni tra task ben formate (sia di composizione che di dipendenza)
- relazioni tra task non ben formate, in modo da verificare la bontà dell'adattamento dell'algoritmi di generazione per i casi anomali

Questo progetto viene creato nella *acceptance machine* e su questa macchina sarà presente la versione completa del progetto necessaria per soddisfare tutte le triple.

Nelle *develop machine* invece non è richiesto necessariamente che sia presente l'istanza completa del progetto, bensi sarà sufficiente riportare solo un sottoinsieme del progetto sufficiente per testare il case che si vuole esercitare.

La struttura che possiamo dare al progetto al primo livello di dettaglio (quindi immediatamente successivo la *root*), è di creare una macro attività relativa ad ogni tripla definita in un documento *case specification*, in modo da esercitare il test in un contesto limitato, non avendo relazioni di composizione. In questo modo possiamo trattare ogni tripla in modo atomico; lo svantaggio di questo approccio è che avremo un *project plan* degenere ad un solo livello.

acceptance project Questo progetto permette di avere una istanza di progetto che si avvicina piú ad una istanza che potrebbe essere creata nella realtá, differenziandosi dalla precedente istanza di test soprattutto nella quantitá di informazioni che costituiscono il progetto.

Per creare questo progetto possiamo utilizzare la nostra istanza di *Project Plan*, creando un backup e caricandolo sulla nostra macchina di *acceptance*. Appena la fase di implementazione fornisce insiemi di elementi sufficienti per esercitare test su questo progetto, eseguiamo il backup del nostro *project plan*. Quindi come limite inferiore per questa data, possiamo assumere la data odierna 15/12/2009.

1.6.3 Strumenti

Durante il processo di testing usufruiamo dei seguenti strumenti:

- controllo visivo umano per quanto riguarda il confronto dell'output con l'output atteso (descritto nei documenti *Test case specification*) per quanto riguarda immagini
- utilizzo di verifica automatica di batterie di test usando l'insieme di oggetti contenuti nella distribuzione di *phpUnit* oppure attraverso classi di test non appartenenti al precedente framework ma che hanno la stessa idea di verifica (controllo automatico tra output e risultato previsto).

1.6.4 Test failure's metrics

Modelliamo il concetto di *failure* aggiungendo delle informazioni, in modo da specializzarlo per il nostro processo di testing. Ogni specializzazione ha il significato di esprimere la gravità (importanza) del fallimento. Quindi, l'esito di un test può essere *success* o *failure*, nel secondo caso aggiungiamo queste specializzazioni:

minor il fallimento del test non è da considerarsi un evento grave.
Gli oggetti software che hanno prodotto questa failure possono essere comunque inseriti nella release di **PMango 3**, non impediscono l'avanzare dello sviluppo. Possiamo identificare di questa failure come un *defect*.

critical il fallimento del test è da considerarsi un evento grave.
Gli oggetti software che hanno prodotto questa failure non possono essere inseriti nella release di **PMango 3**, necessitano di ricontrollare il codice relativo a tali oggetti; non impediscono l'avanzare dello sviluppo.

blocking il fallimento del test è da considerarsi un evento grave.
Gli oggetti software che hanno prodotto questa failure non possono essere inseriti nella release di **PMango 3**, necessitano di ricontrollare il codice relativo a tali oggetti e, se necessario, ricontrollare il relativo documento di progettazione; impediscono l'avanzare dello sviluppo.

Queste misure sono valide per tutte le failure di test appartenenti a ogni level plan descritto in 1.7.

1.7 Level plans

1.7.1 Definitions

Il processo di testing per il progetto **PMango** consiste nei livelli seguenti.

unit questo livello viene effettuato da tutti gli sviluppatori e stilato dal team dei verificatori con un rapprensentante degli sviluppatori.

Ogni motivazione riguardo ogni singolo unit test deve essere resa disponibile e documentata in modo chiaro o in un documento apposito[1] oppure nel codice nel caso che

- viene utilizzato un strumento automatico indicato nelle sezione 1.6.3
- la motivazione ha una dimensione ragionevolmente corta che è possibile inserirla come commento nel codice

[1] inserire un riferimento al relativo documento di test case specification

Questo livello viene esercitato su macchine di tipo *develop machine*.

system/integration questo livello viene eseguito dal team dei verificatori in presenza di un rappresentante degli sviluppatori se necessario.

Ogni motivazione e descrizione di questi test deve essere esposta nei documenti *Test case specification*

Questo livello viene esercitato su macchine di tipo *acceptance machine* e *develop machine*.

acceptance questo livello viene eseguito dal cliente in presenza di un rapprensentante dei verificatori.

Una volta terminato il livello di *acceptance* il prodotto viene rilasciato al cliente il quale può continuare la fase di testing in parallelo alla fase di utilizzo.

Questo livello viene esercitato su macchine di tipo *develop machine*.

1.7.2 Precedence Relation

Possiamo costruire la relazione di precedenza $\sqsubset$ fra coppie di level plan che permette ad un oggetto software di avanzare nel processo di testing, in modo da garantire il suo corretto funzionamento durante la fase di revisione congiunta oppure nel collaudo.

Definiamo $\sqsubset$ in questo modo:

unit $\sqsubset$ system/integration ogni oggetto software entra nel processo di testing dal level plan *unit*.

Quando soddisfa tutti i suoi *unit* test oppure, per ogni fallimento, la metrica misura *minor*, allora può essere disponibile per il level plan *system/integration* se è richiesto da qualche *system/integration* test.

system/integration $\sqsubset$ acceptance quando un oggetto (o gruppo di oggetti) superano tutti i test oppure, per ogni fallimento, la metrica misura *minor*, allora l'oggetto (o gruppo di oggetti) può avanzare nel level plan *acceptance*

La relazione $\sqsubset$ è riflessiva, in quanto un test permane in un level plan finche non soddisfa i requisiti per passare nel successivo; non è ne simmetrica ne transitiva, in quanto vogliamo garantire al committente la sequenzialità del processo di testing.

Osservazione: quando un test passa da un livello al successivo deve essere continuo rispetto alla batteria dei test specificata nel livello che lascia. Ovvero: se avanza di livello deve continuare a soddisfare le condizioni che gli hanno permesso di avanzare fino al livello corrente.

1.8 Pass/fail criteria

Il risultato dell'intero piano delle prove è dato dalla seguente relazione:

risultato	**criteri**
success	numero di *minor* failures ≤ 10
minor failure	numero di *minor* failures > 10
critical failure	almeno una *critical* failure
blocking failure	almeno una *blocking* failure

Table 1.1: La colonna *criteri* si riferisce all'insieme dei *design specification* definite in II

Il processo di testing verrà completato nella data in cui avverrà il collaudo con il committente oppure quando questo piano da esito *success* in base alla relazione definita dalla tabella, prima del collaudo. Dalla fine del processo di testing, la nuova versione di PMango viene considerata *live*.

Nel caso in cui il nostro team non riuscisse a portare a termini gli impegni presi entro la data del collaudo, il processo di testing proseguirà fino alla data in cui si considera terminato il tempo a disposizione per eseguire l'esame.

1.9 Enviromental needs

I seguenti elementi sono richiesti per supportare l'intero processo di testing:

- Sia *develop machine* che *acceptance machine* devono avere installato una istanza di un server (L/W/M)AMP, con tutti i necessari permessi per la corretto funzionamento, relativamente al sistema operativo presente
- Sia *develop machine* che *acceptance machine* devono offrire tutte quelle *third party resources* necessarie per l'utilizzo della nuova versione di PMango (fonts microsoft, ...) compresi tutte quelle necessarie per la versione di PMango attuale.

Part II

Desing Specification

Chapter 2

Generate Chart Design

2.1 Features to be tested

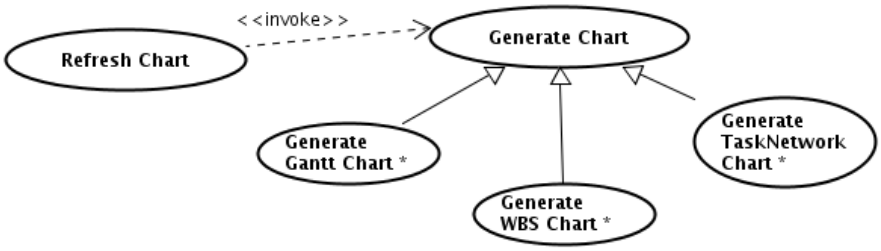

Figure 2.1: spaccato tratto dal package *Commons* del documento di analisi, parte *Use cases*

Verranno testati tutti gli use case espressi dalla figura tranne "Refresh Chart" in quanto è stato riportato solo per favorire la comprensione del contesto e perche la sua attività di costruire la *UserOptionChoice* e recuperare l'insieme dei *Task* vengono testate nel documento [1]

2.2 Raffinamento della strategia

confronto dell'output Per ogni use case di "generazione" si fa riferimento alle notazioni espresse nel documento di specifica e se il budget del progetto lo consente, verranno prodotti dei modelli, in notazione *mockup* disegnati a mano, per poter effettuare il confronto con le immagini di output generate dall'implementazione.

[1] make a test document for these use cases.

modalità di esecuzione Ogni use case verrà esercitato in modo *individuale*, non verranno esercitate combinazioni con altri use case, come la figura esprime.

2.3 Tests identification

Sotto vi sono i reference a tutte le *case specification* relative a questo *desing*, definite in III:

- Generate Gantt Case
- Generate WBS Case
- Generate Task Network Case

2.4 Pass/fail criteria

Il risultato di questo desing è espresso dalla seguente relazione rappresentata da questa tabella:

risultato	**criteri**
success	numero di *minor* failures ≤ 5
minor failure	numero di *minor* failures > 5
critical failure	almeno una *critical* failure
blocking failure	almeno una *blocking* failure

Table 2.1: La colonna *criteri* si riferisce all'insieme di tests identificati nella sezione 2.3

Part III

Case Specification

Chapter 3

Generate Gantt Case

[1]Ogni successiva sezione rappresenta la tripla di testing *(Input, Environment, Output)*. Ogni tripla ha l'obiettivo di identificare un oggetto del diagramma che si vuole testare. Quando la componente *Environment* non viene specificata significa che la funzionalità non richiede condizioni particolari per il suo avvenimento (ad esempio le dipendenze vengono mostrate solo se l'utente seleziona *ShowDependenciesUserOption*).

Level plan relation Le seguenti triple sono da considerarsi appartenenti ad un level plan in questo modo:

unit se si stanno esercitando le triple 3.1, 3.2, 3.3, 3.4, 3.5 sul *testing project*, allora appartengono al *unit level*; la componente su cui viene eseguito lo unit testing è la classe *GifArea* con le sue specializzazioni

system/integration tutte le triple possono considerarsi incluse al *system/integration level* in quanto i dati su cui vengono costruite le rappresentazioni vengono recuperate secondo quanto descritto nel *sequence diagram* nella sezione **1.1 Commons, 1.3 Task Data Tree** del documento *disegno del sistema*.

acceptance se si stanno esercitando le triple 3.1, 3.2, 3.3, 3.4, 3.5 sul *acceptance testing project* allora appartengono al *acceptance level*. Questo perchè nel *acceptance testing project* ci possono essere task candidati per esercitare una tripla, però inseriti in un contesto più grande.

3.1 Basic task representation

Testare la rappresentazione grafica di un task atomico, non composto da sotto task.

[1]put this paragraph within a section inside a test plan introduction

input almeno 4 esempi

environment le *UserOption FromStartRange, ToEndRange* devono definire un intervallo di tempo nel quale l'input è almeno compreso.

output come descritto nella sezione 1. del documento di specifica.

3.2 Composed task representation

Testare la rappresentazione grafica di un task composto.

input

- si vogliono visualizzare i sottotask di livelli successivo al task composto. Per questa configurazione almeno 2 esempi
- si vuole una rappresentazione collassata dei sottotask del task composto. Per questa configurazione almeno 2 esempi

environment valorizzazione della user option *UserCustomSpecification = checked*, per permettere all'utente di collassare alcuni task

output come descritto nella sezione 1

3.3 Actual time representation

Testare la rappresentazione grafica delle informazione del contesto *actual* di un task.

input

- un task terminato in base alle informazioni *actual* prima della linea verticale che indica *today*. Almeno 4 esempi
- un task non terminato in base alle informazioni *actual* prima della linea verticale che indica *today*. Almeno 4 esempi

environment valorizzazione della user option *UserCustomSpecification = checked*, per permettere all'utente di collassare alcuni task

output come descritto nella sezione 1. e nella figura 2. del documento di specifica.

3.4 Task identifier and name representation

Testare la rappresentazione grafica dell'identificativo e del nome di un task sulla colonna di sinistra.

input

- il nome non supera i limiti espressi nel documento di specifica. Almeno 2 esempi
- il nome supera i limiti espressi nel documento di specifica. Almeno 4 esempi

environment valorizzazione della user option *ShowTaskName*, per permettere all'utente mostrare oppure no il nome del task

output come descritto nella sezione 1. del documento di specifica.

3.5 Task effort/resources representation

Testare la rappresentazione grafica dell'effort e delle resources sulla destra della *Taskbox*.

input

- Supponiamo task composti. Almeno 3 esempi
- Supponiamo task base. Almeno 4 esempi

environment valorizzazione della user option *ShowEffort*

output come descritto nella sezione 1. del documento di specifica.

3.6 Dependency relations representation

See 5.3.

Pass/fail criteria

La colonna *criteri* si riferisce alla rappresentazione grafica prodotta dall'implementazione.

risultato	**criteri**
success	• la costruzione della rappresentazione grafica è coerente con le notazioni espresse nel documento di specifica. Il posizionamento delle informazioni sull'*effort* e sulle *resources* può essere fissato dall'implementazione in quanto non vengono specificate dimensioni o gap predefiniti • costruzione delle testata contenente le informazioni sulla grana temporale può avere una formattazione diversa da quanto mostrato nella sezione *Mockups* del documento di analisi, basta che vengano mostrati i campi relativamente alla grana scelta (vedere relazione descritta nella sezione **2.2.1** del documento *Requisiti del prodotto*). • Le linee verticali rappresentanti un elemento appartenente alla *TimeGrain* possono essere rappresentate (oppure non comparire affatto) in modo diverso dall'implementazione attuale di PMango.
minor failure	le *Strip* che rappresentano le informazioni come descritte nel documento di specifica possono avere delle differenze con una tolleranza di **2mm** rispetto alla figura 2 del documento di specifica. (quindi anche riguardo alle differenti altezze)
critical failure	• i punti di stacco dei segmenti e le spaziature delle dipendenze possono essere fissati in maniera diversa da come viene eseguito nella versione attuale di PMango • i blocchi rappresentanti le informazioni *planned, actual* dei tasks sono staccati fra di loro. • non vengono rispettate le misure relative alla grana temporale scelta espresse nella sezione **2.2.1** del documento delle metriche
blocking failure	La costruzione delle componenti del diagramma non rispetta i vincoli definiti nella sezione **1** del documento di specifica.

Chapter 4

Generate WBS Case

Level plan relation Le seguenti triple sono da considerarsi appartenenti ad un level plan in questo modo:

unit se si stanno esercitando le triple 4.1, 4.2 sul *testing project*, allora appartengono al *unit level*; la componente su cui viene eseguito lo unit testing è la classe *GifArea* con le sue specializzazioni

system/integration tutte le triple possono considerarsi incluse al *system/integration level* in quanto i dati su cui vengono costruite le rappresentazioni vengono recuperate secondo quanto descritto nel *sequence diagram* nella sezione **1.1 Commons, 1.3 Task Data Tree** del documento *disegno del sistema*.

acceptance se si stanno esercitando le triple 4.1, 4.2 sul *acceptance testing project* allora appartengono al *acceptance level*. Questo perchè nel *acceptance testing project* ci possono essere task candidati per esercitare una tripla, però inseriti in un contesto più grande.

4.1 Task representation

Testare la rappresentazione grafica di un task.

input

- task atomico, non suddiviso in sottotask. Per questa configurazione almeno 3 esempi
- task composto, visualizzando i sotto task del livello successivo. Per questa configurazione almeno 2 esempi
- task composto, collassando i sotto task. Per questa configurazione almeno 2 esempi
- task con il contesto *actual* coerente con il contesto *planned*, almeno 2 esempi

- task con il contesto *actual* non coerente con il contesto *planned*, sia in modo lieve ("good news") sia in modo grave ("bad news"), almeno 2 esempi per entrambe i tipi di incoerenza

environment le user option *ActualTimeFrameOption, CompletitionBarOption, CompletitionBarOption, ActualDataOption, AlertMarkUserOption, ResourcesDetailsOption, TaskNameOption* determinano i fields che verranno mostrati nella rappresentazione grafica del task.

output come descritto nella sezione 2. del documento di specifica.

4.2 Dimension of Task representation

Testare la dimensione della rappresentazione grafica di un task.

input

- task aventi un nome la cui larghezza entra nella dimensione specificata nel documento di specifica, almeno 3 esempi
- task aventi un nome la cui larghezza supera la dimensione specificata nel documento di specifica, almeno 3 esempi
- task aventi un numero di resources diverso per qualche coppia di task, almeno 5 esempi

environment valorizzazione della user option *ResourcesDetailsOption, TaskNameOption*, considerando la generazione sia mostrando i nomi dei task, sia non mostrandoli.

output come descritto nella sezione 2

4.3 Composition relations representation

Testare la rappresentazione grafica delle dipendenze di composizione espresse nel *project plan*.

input

- almeno 10 task legati da relazioni di composizione

environment valorizzazione della user option *WBSTreeSpecification* con una delle sue due specializzazioni per esprimere il livello di dettaglio, più altre opzioni per avere la rappresentazione delle informazioni volute.

output come descritto nella sezione 2. e nella figura 4. del documento di specifica.

Pass/fail criteria

La colonna *criteri* si riferisce alla rappresentazione grafica prodotta dall'implementazione.

risultato	**criteri**
success	• la costruzione della rappresentazione grafica è coerente con le notazioni espresse nel documento di specifica. Il posizionamento del simbolo *delta* (con la sua estensione negativa "!") può essere fissato dall'implementazione in quanto non vengono specificate dimensioni o gap predefiniti • l'ordine con cui vengono mostrati i campi della *TaskNodeBox* non è significativo in quanto non esiste una sua definizione nel documento di specifica. Ogni soluzione proposta dall'implementazione è corretta.
critical failure	• i field che compongono la *Taskbox* sono staccati fra di loro. • la spaziatura, i punti di stacco dei segmenti rappresentanti le relazioni di composizione e i punti di uscita/ingresso di tali segmenti n elle *Taskbox* non rispettano i requisiti espressi nella sezione **3** del documento di specifica • non vengono rispettate le misure relative all gap minimo espresse nella sezione **2.2.2** del documento delle metriche
blocking failure	La costruzione delle componenti del diagramma non rispetta i vincoli definiti nella sezione **2** del documento di specifica rilasciato dal committente.

Chapter 5

Generate Task Network Case

Level plan relation Le seguenti triple sono da considerarsi appartenenti ad un level plan in questo modo:

unit se si stanno esercitando le triple 5.1, 5.2 sul *testing project*, allora appartengono al *unit level*; la componente su cui viene eseguito lo unit testing è la classe *GifArea* con le sue specializzazioni

system/integration tutte le triple possono considerarsi incluse al *system/integration level* in quanto i dati su cui vengono costruite le rappresentazioni vengono recuperate secondo quanto descritto nel *sequence diagram* nella sezione **1.1 Commons, 1.3 Task Data Tree** del documento *disegno del sistema*.

acceptance se si stanno esercitando le triple 5.1, 5.2 sul *acceptance testing project* allora appartengono al *acceptance level*. Questo perchè nel *acceptance testing project* ci possono essere task candidati per esercitare una tripla, però inseriti in un contesto più grande.

5.1 Task representation

See 4.1.

5.2 Dimension of Task representation

See 4.2

5.3 Dependency relations representation

Testare la rappresentazione grafica delle dipendenze di dipendenza espresse nel *project plan*.

input

- almeno 10 task divisi in (i seguenti sono tutti da intendersi come foglie dell'esplosione del *project plan* scelta per la generazione, altrimenti non si potrebbero disegnare le dipendenze come espresso nella sezione 4 del documento di specifica):
  - atomici
  - collapsed, non visualizzando i sotto task
  - not collapsed, visualizzando i sotto task

  legati da relazioni di dipendenza "finish to start".
- almeno 5 tasks (*A, B, C, D, E*) che appartengon alla relazione "finish to start"

$$\rightarrow = \{(A, C), (B, C), \ldots\}$$

- almeno 5 tasks (*A, B, C, D, E*) che appartengon alla relazione "finish to start"

$$\rightarrow = \{(A, B), (A, C), \ldots\}$$

environment si valorizzano queste user option *ReplicateArrowUserOption, CompleteDiagraUserOption, ShowTimeGapUserOption* e i precedenti input vengono ripetuti per ogni combinazione dei possibili valori possibili per l'ambiente (8 casi in totale)

output come descritto nella sezione 2. e nella figura 4. del documento di specifica.

5.4 Critical path representation

Testare la rappresentazione grafica dei possibili critical path nel *project plan*.

input

- almeno 10 task che producono un solo critical path
- almeno 10 task che producono più di un critical path (provare con 3??)

environment si valorizzano queste user option *ShowCriticalPath*

output come descritto nella sezione 2. e nella figura 4. del documento di specifica.

Pass/fail criteria

See the figure 5.1.

risultato	**criteri**
success	• vedi la relativa sezione *success* definita per il wbs case specification 4.3 • nel caso si vogliano disegnare i *critical path* la tabella può essere posizionata ovunque nello spazio disponibile, in quanto nel documento di specifica non viene fissata una posizione precisa. • la formattazione e la posizione dell'informazione *Time gap* può variare, basta che sia vicina con una tolleranza di **3mm** al segmento a cui è riferita.
minor failure	la posizione dei task che non hanno dipendenze con altri task, (ma vengono solo completati, necessaria la scelta dell'opzione *CompleteDiagramUserOption*, associandoli alla *start/end milestone*), possono essere piazzati seguendo l'euristica proposta dall'implementazione perchè nel documento di specifica non ne viene richiesta una disposizione particolare.
critical failure	• i field che compongono la *Taskbox* sono staccati fra di loro. • la spaziatura, i punti di stacco dei segmenti rappresentanti le relazioni di dipendenza non rispettano i requisiti espressi nella sezione **4** del documento di specifica • non vengono rispettate le misure relative all gap minimo espresse nella sezione **2.2.3** del documento delle metriche
blocking failure	La costruzione delle componenti del diagramma non rispetta i vincoli definiti nella sezione **2** del documento di specifica rilasciato dal committente.

Table 5.1: La colonna *criteri* si riferisce alla rappresentazione grafica prodotta dall'implementazione

Part IV

Reporting

Chapter 6

Items Transmittal report

In questo capitolo vengono registrati i componenti prodotti nella fase di implementazione, con i riferimenti ai sorgenti dove i componenti sono implementati. Per ogni componente possono essere registrate più informazioni, le quali rispettano questo pattern: data nella quale si ritiene che la codifica del componente permetta l'esercizio della funzionalità catturata dal test; numero di *revision* del commit sul repository; differenze significative rispetto alla versione precedente disponibile per testing.

Nessun componente disponibile per testing.

6.1 GanttChartGenerator

Source code in:

6.1.1 22/12/2009

revision

differences

Chapter 7

Tests logs

Questo capitolo è un contenitore di log. Ogni log è in relazione one-to-one con una tripla definita in un case specification, nella parte III, ed ogni log viene descritto in una sezione indipendente dalla altre.

7.1 Log of 3.1 triple

7.1.1 22/12/2009

input in

esito

confronto dei risultati con output attesi

commenti per correzione errori

Chapter 8

Summary report

Questo capitolo riassume l'esito dell'intero piano di test e delle sue componenti:

8.1 Case Specification 3

triples	**esito**	**causes' logs**
Basic task representation	*failure*	none
Composed task representation	*failure*	none
Actual time representation	*failure*	none
Task identifier and name representation	*failure*	none
Task effort/resources representation	*failure*	none
Dependency relations representation	*failure*	none

8.2 Case Specification 4

triples	**esito**	**causes' logs**
Task representation	*failure*	none
Dimension of Task representation	*failure*	none
Composition relations representation	*failure*	none

8.3 Case Specification 5

triples	**esito**	**causes' logs**
Task representation	*failure*	none
Dimension of Task representation	*failure*	none
Dependency relations representation	*failure*	none
Critical path representation	*failure*	none

8.4 Design Specification 2.1

case specification	**esito**	**causes' logs**
Generate Gantt Case	*failure*	none
Generate WBS Case	*failure*	none
Generate Task Network Case	*failure*	none

8.5 Master Plan

desing specification	**esito**	**causes' logs**
Generate Chart Design	*failure*	none