Documento di Analisi

Release 1.1

November 21, 2009
Firenze

Approvazione, redazione, lista distribuzione

approvato da	**il giorno**	**firma**
Marco Tinacci	November 21, 2009	

redatto da	**il giorno**	**firma**
Francesco Calabri	November 21, 2009	
Manuele Paulantonio	November 21, 2009	
Massimo Nocentini	November 21, 2009	

distribuito a	**il giorno**	**firma**
Daniele Poggi	November 21, 2009	
Niccoló Rogai	November 21, 2009	
Marco Tinacci	November 21, 2009	

Contents

Introduzione

Descrizione dell'acronimo: **cdns** sta per "**c**ome **d**escritto **n**ella **s**pecifica".

Part I

Requisiti del prodotto

Release 1.1

Chapter 1

Requisiti obbligatori

1.1 Generali (2.1)

Come requisiti fondamentali, **PMango 3.0** sarà visualizzabile e usabile con le ultime versioni *Internet Explorer 8* e *Mozilla Firefox 3.0*.

Le nostre modifiche e aggiunte saranno distribuite senza costi di licenza, in quanto si tratta di estensioni di un progetto GPL

Ci assumiamo la responsabilità di essere conformi ai punti *d), e)*.

1.2 Diagrammi WBS, Gantt e Task Network (2.2)

a) implementato nello use case **6.3**.

b) implementato negli use case **6.2** e in **7.2**.

c) implementato nello use case **6.5**

1.3 Diagrammi specifici (2.3)

a) implementato nello use case **6.5** [1]

b) implementato nello use case **9.2**

c) implementato nello use case **6.5**

d) implementato nello use case **6.5** e descritto in modo dettagliato nella sezione **4.6.1** del documento **Domain Model**

[1] replace every ShowUserOption reference with the relative generalization

e) realizzato nello use case **6.5**

1.4 Generazione di immagini e doc (2.4)

a) realizzato negli use case **6.10, 6.9**

b) realizzato nello use case **6.8**

c) realizzato nello use case **6.5** e descritto in modo dettagliato nella sezione **4.6.1** del documento **Domain Model**

d) vedi punto *a)*

e) realizzato nello use case **6.12**

1.5 Documentazione (2.5)

Ci assumiamo la responsabilità di essere coerenti a quanto richiesto nei punti *a), b), c)* nei momenti in cui verranno effettivamente implementati.

Chapter 2

Semplificazioni, Metriche

2.1 Semplificazioni e requisiti aggiuntivi

a) il nostro gruppo **non** prevede lo sviluppo di requisiti aggiuntivi, preferendo implementare correttamente il processo di sviluppo adottato per raggiungere i requisiti richiesti dal committente.

b) abbiamo deciso di creare **solo** oggetti **gif** per usarli in modo interscambiabile sia nella visualizzazione da browser web, sia per aggiungerli in documenti PDF. Questo ci porta alcuni vantaggi:

   – ci interfacciamo con una sola libreria, avendo cosi modo di capirne a fondo il comportamento e eventualmente aggiungere quelle funzionalità di helper che potrebbero servirci, ma che attualmente non vengono fornite.

   – ci riduce il carico di lavoro, questo non preclude che se arriviamo in anticipo con un prodotto finito e che rispetta la specifica richiesta, potremo proporre una integrazione dell'offerta sviluppando le funzionalità native per la rappresentazione in PDF.

2.2 Metriche

Part II

Domain Model

Release 1.1

Chapter 3

Overall diagram

Questo diagramma comprende tutti i concetti che abbiamo identificato durante la prima iterazione del blocco di analisi.

Nella figura abbiamo una visione di insieme che può essere utile a fini di codifica e progettazione del piano delle prove. Finchè si rimane invece nella sfera della progettazione (analisi inclusa) potrebbe produrre dei dubbi in quanto propone molti concetti; mentre si sta cercando di raffinare le varie relazioni secondo noi è necessaria una vista più in dettaglio di composizioni di pochi concetti che sono legati tra loro, lasciando tutti gli altri ad una loro commento separato.

Procediamo nel seguito del documento nella descrizione di piccole composizioni in modo da chiarire i motivi per cui sono stati creati concetti e relazioni fra essi.

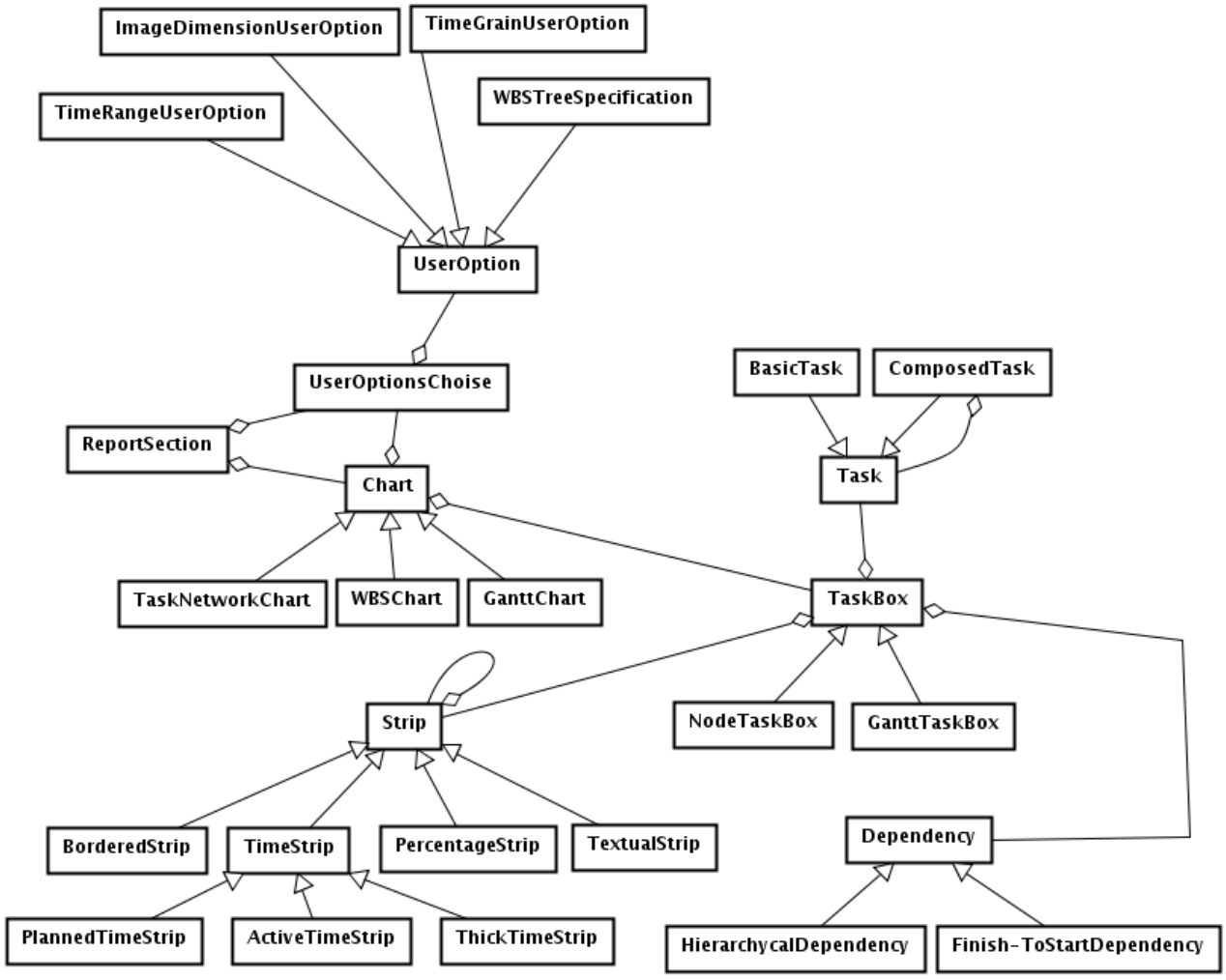

Figure 3.1: Overall UML diagram

Chapter 4

Aspects descriptions

4.1 Task

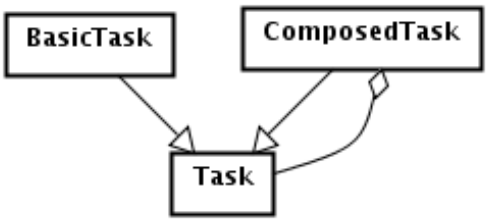

Figure 4.1: task and its relations

Molto probabilmente il concetto di *Task* esiste gia nell'attuale versione di **PMango 2.2.0**. Quello che abbiamo pensato è di introdurre un *glue layer* che ci permette di non apportare modifiche al codice esistente di mango, ma lavorare con uno strato di intermezzo per essere il meno intrusivi possibile e poter portare avanti il lavoro dipendendo solo dalle nostri oggetti, facendo il minor riferimento al codice gia esistente.

Vogliamo rendere trasparente il concetto che un *Task* sia un attività singola (non scomponibile in sottoattività) che una attività scomposta.

Costruiamo la relazione → che lega questi due concetti:

- *BasicTask* → attività di base, non ulteriormente scomponibili
- *ComposedTask* → attività che sono composte da sotto attività

In questo modo possiamo trattare questi due tipi di attività in modo interscambiabile e del tutto trasparente. Usando l'astrazione *Task* non ci importa se abbiamo una attività base o composta, in quanto cosi le abbiamo portate ad avere interfaccie compatibili.

4.2 TaskBox

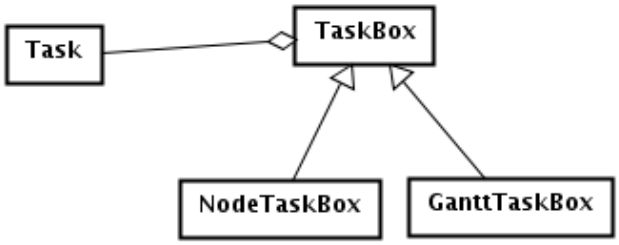

Figure 4.2: taskbox and its specializations

Il *TaskBox* è la rappresentazione grafica di un *Task* (Figure 4.1). Questo concetto astrae su queste specializzazioni:

- *GanttTaskBox* che ci permetterà di costruire la rappresentazione in un *GanttChart* conformi alle norme fissate nel documento di specifica.
- *NodeTaskBox* che ci permetterà di costruire la rappresentazione in un *WBSChart* e in *TaskNetworkChart* alle norme fissate nel documento di specifica.

Abbiamo usato il principio di incapsulare il concetto che varia, modellando il concetto astratto di *TaskBox* per avere questi vantaggi:

- non legare un *Chart* specifico a una rappresentazione specifica
- aggiungere una nuova rappresentazione consiste nel modellarla e dichiarare che si tratta di una specializzazione di *TaskBox*
- potremo cambiare a runtime il tipo di rappresentazione voluta nel disegno di un *Chart*, magari inserire in un *WBSChart* una rappresentazione pensata per i *GanttChart*

4.3 Strip

La *Strip* modella il concetto di ”striscia”: lo possiamo vedere come il building block di più basso livello di tutta la nostra analisi. Si possono osservare queste relazioni:

TaskBox composition *TaskBox* contiene delle *Strip*, indipendentemente dalla rappresentazione dedicata ad uno specifico *Chart*. Abbiamo costruito questa relazione in quanto per costruire un *TaskBox* sarà sufficiente comporre un insieme di strip, tante quante sono necessarie per la corretta visualizzazione del *Chart* che si sta disegnando.

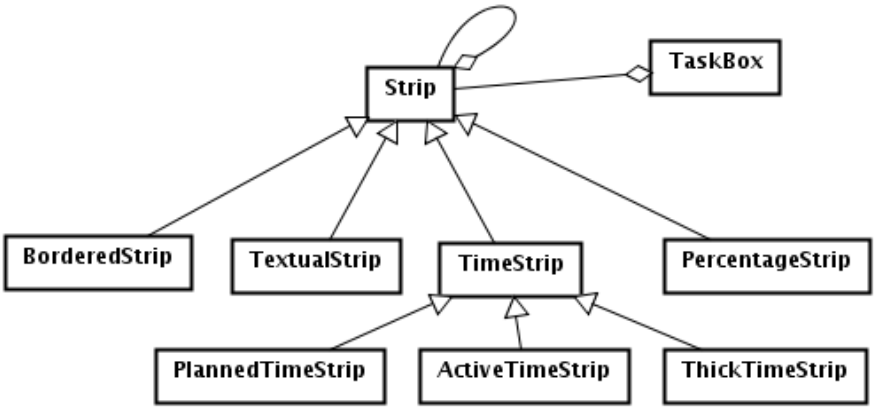

Figure 4.3: kinds of strips

russian doll *Strip* contiene a sua volta delle *Strip*: questo è un concetto che pensiamo posso essere molto potente. Vogliamo rendere la *Strip* un contenitore trasparente rispetto ad oggetti del suo stesso tipo. Questo ci permetterà di disegnare *Strip* annidate, decidendo a runtime sia la **profondità** di annidamento sia l'**ordine** con cui vengono annidate. Otteniamo cosi l'effetto di *russian doll*, dedicandoci a modellare solo alcune semplici specializzazioni di *Strip*, necessarie per l'implementazione delle notazioni richieste nel documento di specifica, limitandoci poi ad ottenere rappresentazioni complesse annidando quelle semplici.

specializations abbiamo un primo livello di specializzazione:

- *PercentageStrip* rappresenta un quantità (l'intera *Strip*) e la percentuale di comletamento (una parte di *Strip* di colore diverso). Può essere composta da un *GanttTaskBox* per indicare la percentuale di completamento relativa al *Task* rappresentato.
- *TextualStrip* permette di inserire delle stringhe di caratteri all'interno della *Strip*. Questa possiamo utilizzarla ad esempio nel *GanttChart* sulla destra della relativa *TaskBox* per indicare l'effort oppure le risorse.
- *BorderedStrip* permette di costruire un bordo, in modo che possiamo implementare la notazione per *field* come richiesto per *NodeTaskBox*
- *TimeStrip* permettono di rappresentare informazioni relative a un intervallo di tempo. Queste sono i building blocks per *GanttChart*. Possiamo specializzare ulteriormente questo concetto:
  - *PlannedTimeStrip* per costruire la parte superiore del *GanttTaskBox* cdns
  - *ActualTimeStrip* per costruire la parte inferiore del *GanttTaskBox* cdns

– *ThickTimeStrip* per costruire la parte superiore del *GanttTaskBox* nel caso la rappresentazione sia di un *ComposedTask* cdns

4.4 Chart

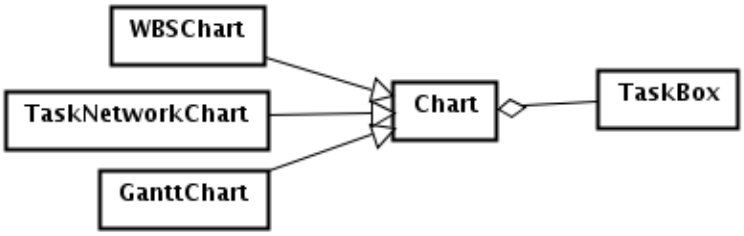

Figure 4.4: chart and building blocks

Il *Chart* modella il concetto di grafico generico. Per implementare la specifica abbiamo queste specializzazioni:

- *GanttChart* che permette di avere la rappresentazione delle attività nel tempo
- *WBSChart* per avere una vista gerarchica delle attività e di come sono state raffinate e decomposte in sotto attività
- *TaskNetworkChart* per rappresentare le dipendenze di tipo *finish-to start* fra coppie di attività

Per costruire un *Chart* è sufficiente assemblare *TaskBox* in base ad alcune preferenze del client che richiede la generazione.

4.5 Dependency

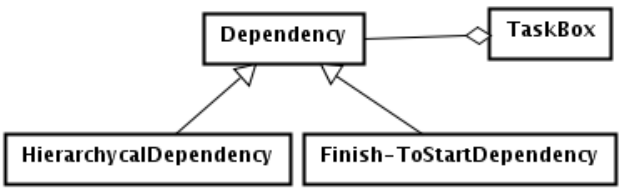

Figure 4.5: dependencies

Dependency modella il tipo di dipendenze che possiamo rappresentare in un *Chart*. Per implementare la specifica abbiamo bisogno di incapsulare queste varianti:[1]

- *Finish-ToStartDependency*: siano a, b due *Task* tali che b non può iniziare finché a non sia completato. Questa relazione è catturata da questa specializzazione.
- *HierarchycalDependency*: siano a, b_i con $i = 1, \ldots, n \in N$, *Task*s tali che a è scoposto in b_i *Task*. Questa relazione è catturata da questa specializzazione.

4.6 UserOption

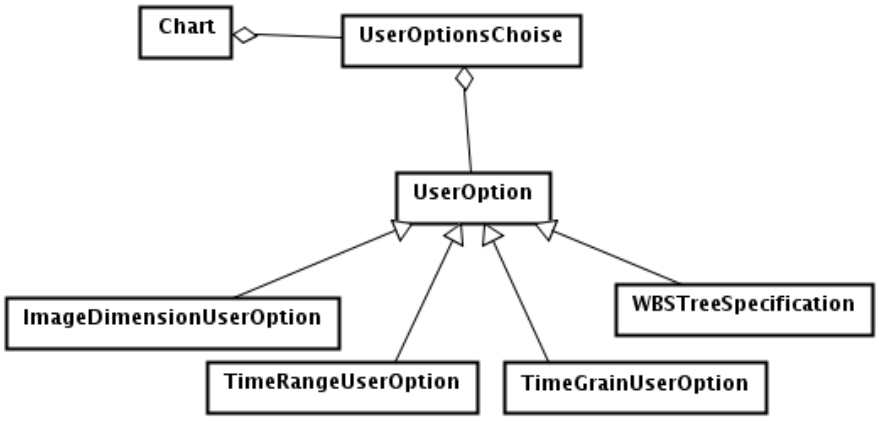

Figure 4.6: UserOptions and choice

Quando un generico client (potrebbe essere sia una persona fisica che un oggetto astratto) della nostra implementazione della specifica vuole generare un *Chart* può guidare la generazione decidendo alcuni fattori che sono di suo interesse. Questi fattori vengono modellati dal concetto di *UserOption*.

Ogni *Chart* espone una lista di *UserOption* per dare al client la possibilità di esprimere quali informazioni guidare. Questa lista varia da *Chart* a *Chart*[2].

Il concetto di lista di *UserOption* è catturato in *UserOptionsChoice*.

Procediamo per passi: nelle prossime subsection osserviamo due aspetti che trattarli insieme potrebbe non essere sufficiente per esporli in modo chiaro.

[1] dire in quali Chart vengono utilizzate

[2] creare un reference dove vengono mappate questa relazione: potrebbe essere un appendici di questo documento??

4.6.1 *UserOption*'s Instances

Questa è stata una decisione non molto facile da prendere. Il problema è questo: nella specifica abbiamo che per ogni *Chart* il committente ha dichiarato quali *UserOption* mostrare. Queste però non rappresentano un concetto che vogliamo catturare nel nostro modello, ma allo stesso tempo sono *istanze* (un insieme discreto quindi) di elementi che fissa il committente.

Per questo motivo decidiamo di codificare questo insieme discreto in questo documento e la successiva enumerazione è da considerarsi parte integrante del diagramma inserito come figura.

Rappresentiamo il concetto espresso sopra indicando due descrizioni con questa struttura di codifica:

- *istanze*, dove inseriamo tutte le possibili *UserOption* che non possono essere ancora raffinate
- *specializzazioni* dove inseriamo tutte le possibili specializzazioni di *UserOption* che possono essere ancora raffinate, ripetendo in modo ricorsivo questa struttura di codifica

Le successive descrizioni sono relative al concetto di *UserOption*:

istanze [3] *WBSExplosionLevelUserOption, ActualTimeFrameOption, CompletitionBarOption, PlannedDataOption, ActualDataOption, AlertMarkUserOption, ReplicateArrowUserOption, FindCriticalPathUserOption, WBSUserSpecificationUserLevel, WBSUserSpecificationUserLevel, ResourcesDetailsOption, TaskNameOption, CompleteDiagramUserOptions*

specializzazioni

- *WBSTreeSpecification*
  istanze *LevelSpecification, UserCustomSpecification*
  specializzazioni nessuna
- *TimeGrainUserOption*
  istanze *WeaklyGrain, MonthlyGrain*
  specializzazioni nessuna
- *ImageDimensionUserOption*
  istanze *CustomDim, FitInWindowDim, OptionalDim, DefaultDim*
  specializzazioni nessuna
- *TimeRangeUserOption*
  istanze *CustomRange, WholeProjectRange, FromStartRange, ToEndRange*
  specializzazioni nessuna

[3] inserire qui la il mapping sui vari Chart?

4.6.2 *UserOptionsChoice*

Questo concetto è, secondo la nostra analisi, molto importante in quanto ci permette di astrarre dal client che richiede una generazione.

Il motivo per cui abbiamo introdotto questo concetto è di poter lavorare lato server usando *UserOptionsChoice* per controllare quali informazioni il client vuole guidare. In questo modo non siamo vincolati ad accedere ai dati inviati per *POST, GET* dalla form HTML, ma possiamo direttamente guardare in *UserOptionsChoice*. Queste ci permette di disaccopiare il processo di generazione della maschera di input di una pagina HTML.

Se vogliamo utilizzare il processo di generazione (che comunque è server side) scrivendo un programma client (GUI o da riga di comando) che costruisce una HTML request ad hoc (dovremo definire una grammatica e attribuire la semantica ai contesti, questo è necessario, non chè scrivere un parser), usando *UserOptionsChoice* e il suo disaccoppiamento ci sarà possibile farlo.

Una volta ricevuta la response possiamo maneggiare la pagina inviata come una response HTML valida e usarla per i nostri obiettivi (possiamo rihiedere l'immagina generata, o il file PDF generato, salvandolo in locale, oppure visualizzando lo stesso con un browser, ma possiamo anche inserire in un db oppure farci dei test sopra...).

Dovremo quindi costruire un oggetto che si incarica di costruire *UserOptionsChoice* in base al tipo di richiesta ricevuta (da una pagina html come è il caso di PMango, oppure una richiesta da un client indipendente scritto in un qualche linguaggio). Una volta costruito l'insieme delle *UserOption* è possibile iniziare la generazione. Questo sarà delegato alla fase di progettazione.

4.7 ReportSection

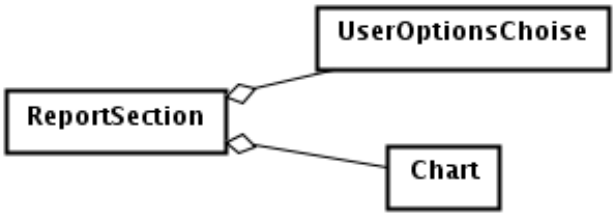

Figure 4.7: report section

Abbiamo da implemetare un requisito che vuole la possibilità di aggiungere alla reportistica un determinato *Chart* con le relative *UserOption* scelte dall'utente. Modelliamo quindi il concetto di *ReportSection* per realizzare questo requisito. Come si vede dalla figura, *Report-*

Section associa *Chart* e *UserOptionChoice*. Utilizziamo direttamente la lista delle scelte[4] in modo da non doverla costruire nella funzionalità di assemblamento del report.

[4]che viene costruita lato server

Part III

Use cases

Release 1.0

Chapter 5

Entire System UML diagram

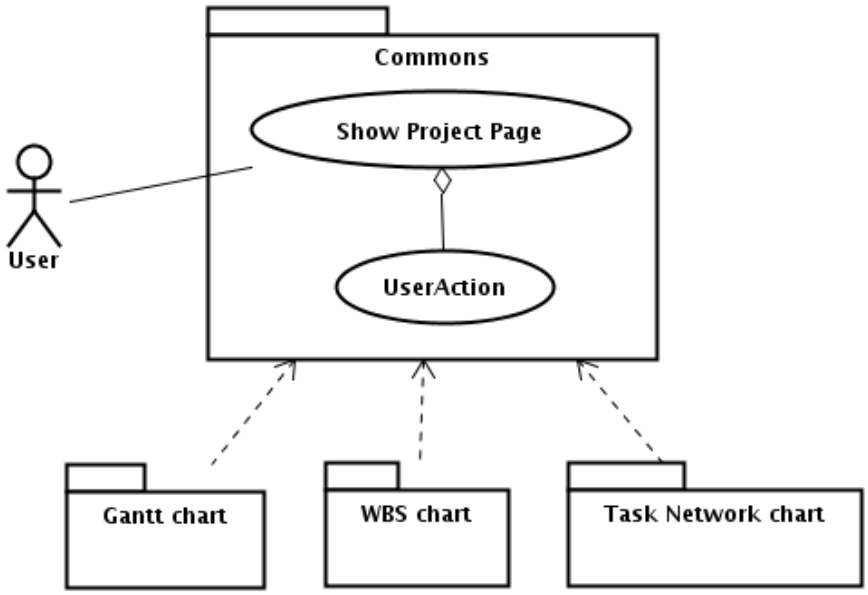

Figure 5.1: Entire system UML diagram

Chapter 6

Commons

6.1 Make Dependencies

6.1.1 Basic course

Si assume che l'insieme di *TaskBox* sia già costruito.

Il client richiede di rappresentare graficamente le *Finish-ToStartDependency* relative all'insieme di *TaskBox* già costruite.

Il sistema esegue questi passi:

1. costruisce un insieme Dep di coppie del tipo (a, b), con $a, b \in$*TaskBox*, tale che $(a, b) \in Dep \Leftrightarrow b$ non può iniziare finchè a non è stata completata.
2. per ogni coppia $(a, b) \in Dep$ costruisci una linea spezzata cdns.

6.1.2 Alternative course

not well formed project Se la struttura al albero WBS del progetto non è ben formata allora si deve cercare di dare la migliore euristica possibile per la rappresentazione delle dipendenze.

6.2 Make NodeTaskbox

6.2.1 Basic course

Il client prende un reference al *WBSChartGenerator* e domanda di creare la rappresentazione grafica di un *Task*.

Il sistema esegue questi passi:

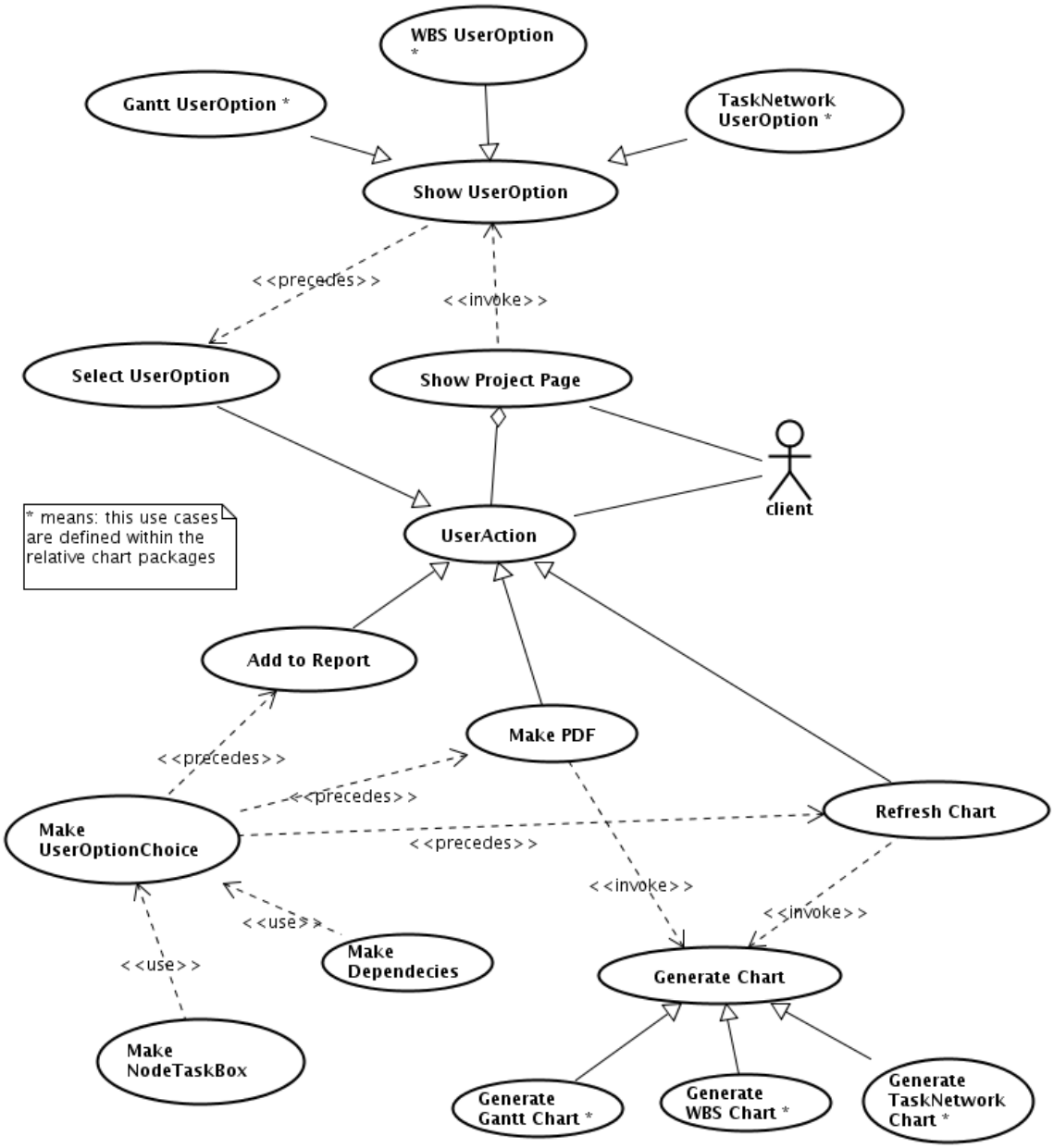

Figure 6.1: Overall Commons UML diagram

1. recupera il *Task* da rappresentare.

2. costruisce la rappresentazione grafica, il *NodeTaskbox* in base alla scelta *WBSTreeSpecification*. Il *NodeTaskbox* è una composizione di *Strip*. Il sistema costruisce una biezione tra *BoxedStrip* e le scelte presenti in *UserOptionsChoice* in questo modo:

- se *UserOptionsChoice* contiene *TaskNameOption*, allora costruisce una *Strip* contenente il nome del *Task* cdns.
- se *UserOptionsChoice* contiene *ResourcesDetailsOption*, allora sul margine destro della *GanttTaskBox* appendi la stringa contenente la lista delle risorse cdns.
  Altrimenti appendi sul margine destro l'effort cdns.
- se *UserOptionsChoice* contiene *PlannedTimeFrameOption* allora costruisce due *Strip* adiacenti contenenti rispettivamente le date di inizio e fine *Task* cdns.
- se *UserOptionsChoice* contiene *ActualTimeFrameOption* allora costruisce due *Strip* adiacenti contenenti rispettivamente le date di inizio e fine *Task* reali cdns.
- se *UserOptionsChoice* contiene *PlannedDataOption* allora costruisce tre *Strip* adiacenti contenenti rispettivamente la durata pianificata, lo sforzo complessivo pianificato, il costo pianificato del *Task* cdns.
- se *UserOptionsChoice* contiene *ActualDataOption* allora costruisce tre *Strip* adiacenti contenenti rispettivamente la durata "dall'inizio ad oggi", lo sforzo complessivo "ad oggi" effettuato, il costo complessivo "ad oggi" del *Task* cdns.
- se *UserOptionsChoice* contiene *CompletitionBarOption* allora costruisce la barra di completamento del *Task* cdns.
- to complete with the missing options

6.2.2 Alternative course

troncamento del nome del *Task* se la stringa scritta supera la dimensione fissata nel documento di specifica, allora il sistema la tronca cdns.

6.3 Show Project Page

6.3.1 Basic course

Il client digita l'URL della *ProjectPage*[1].
Il sistema invia in risposta la pagina richiesta aggiungendo all'insieme dei tab presenti, tre tab relativi ai *Chart* descritti nel documento **Domain Model** e oggetto dell'appalto.
Il client fa click sul tab relativo al *Chart* che vuole generare.
Il sistema invoca la specializzazione dello usecase 6.5 relativa al *Chart* scelto, per costruire l'insieme delle possibili *UserOption*: dopo di che aggiunge questo insieme alla pagina di risposta.
Il client effettua una azione, invocando una specializzazione di 6.7.

[1] TODO:definire ProjectPage nel documento dei Mockup

6.4 Generate Chart

Questo use case, come rappresentato nel diagramma UML Figure 6.1, fornisce il punto di astrazione per altri use case. Per questo motivo la descrizione del comportamento che si vuole modellare viene descritta in ogni specializzazione.

6.5 Show UserOptions

Questo use case, come rappresentato nel diagramma UML Figure 6.1, fornisce il punto di astrazione per altri use case. Usiamo questo formalismo per permettere ad ogni specializzazione di esprimere solo quali sono le *UserOption* disponibili per il relativo *Chart*.

6.6 Make UserOptionsChoice

6.6.1 Basic course

Il sistema riceve una http request contenente una sequenza di *UserOption* che sono state selezionate dall'utente. Costruisce una *UserOptionsChoice* cosi: per ogni *UserOption* indicata nella request si aggiunge all'oggetto in costruzione.

Lato server abbiamo l'insieme di scelte disponibile per le azioni successive.

6.7 User Action

Questo use case, come rappresentato nel diagramma UML Figure 6.1, fornisce il punto di astrazione per altri use case. Per questo motivo la descrizione del comportamento che si vuole modellare viene descritta in ogni specializzazione.

6.8 Add to Report UserAction

6.8.1 Basic course

Si assume che il client stia interagendo con il tab relativo al *Chart* che vuole generare.

Il client fa click sul pulsante "Add to Report".

Il sistema invoca lo use case 6.6 per costruirsi la *UserOptionsChoice*.

Il sistema aggiunge una *ReportSection* per aggiungere alla reportistica il *Chart* richiesto, con la relativa *UserOptionsChoice*. Queste sezioni saranno elencate nella schermata della reportistica già esistente.

6.9 Make PDF

6.9.1 Basic course

Si assume che il client stia interagendo con il tab relativo al *Chart* che vuole generare.
Il client fa click sul pulsante “Make PDF”.
Il sistema esegue questi passi:

- invoca lo use case 6.6 per costruirsi la *UserOptionsChoice*.
- esegue una ricerca dei *Task* che devono essere inclusi nel *Chart*.
- invoca la specializzazione di 6.4 per la creazione del relativo *Chart*
- costruisce un file pdf aggiungendo al suo interno la rappresentazione generata
- invia al client una pagina di risposta con una **icona** accanto ai due pulsanti della reportistica, per segnalare che il file PDF è disponibile.

6.10 Refresh Chart

6.10.1 Basic course

Si assume che il client stia interagendo con il tab relativo al *Chart* che vuole generare.
Il client fa click sul pulsante “Refresh”.
Il sistema esegue questi passi:

- invoca lo use case 6.6 per costruirsi la *UserOptionsChoice*.
- esegue una ricerca dei *Task* che devono essere inclusi nel *Chart*.
- invoca la specializzazione di 6.4 per la creazione del relativo *Chart*
- invia al client una pagina di risposta inserendo la rappresentazione nel bottom della pagina.

6.11 Select User Option

6.11.1 Basic course

Si assume che il client stia interagendo con il tab relativo al *Chart* che vuole generare, quindi lo use case 6.3 è già stato completato.

Il client vuole selezionare alcune *UserOption* per guidare la rappresentazione delle informazioni che verranno codificate nel *Chart*.
Queste sono rappresentate tramite una form html, sotto forma di controlli grafici, dipendenti dal tipo e dalla semantica della *UserOption* che rappresentano. [2] Il sistema non effettua alcuna azione in quanto il fatto di memorizzare le scelte fino al "submit" viene tenuto dalla form html stessa.

6.12 Open in New Window

6.12.1 Basic course

Si assume che il client stia interagendo con il tab relativo al *Chart* che vuole generare.
Il client fa click sul pulsante "Open In New Window".
Il sistema esegue questi passi:

- apre il collegamento in una nuova pagina vuota
- invoca lo use case 6.10 per costruire il *Chart*
- visualizza il *Chart* nella nuova pagina

[2] inserire qua il mappaggio tra il tipo di UserOption e il controllo grafico che viene bindato.

Chapter 7

Gantt chart

Overall UML diagram

7.1 Make Left Column

7.1.1 Basic course

Il client prende un reference al *GanttChartGenerator* e domanda di creare la colonna di sinistra del diagramma.

Il sistema esegue questi passi:

1. costruisce una colonna di larghezza di dimensione **fissa**, che viene calcolata cdns.
2. Il sistema effettua una ricerca per calcolare l'insieme dei *Task* necessari da scrivere nella colonna.
3. Per ogni *Task* trovato si scrive il WBS identifier
   (a) se *UserOptionChoises* contiene la scelta *TaskNameOption* allora accoda alla stringa scritta al punto prima il nome del task

7.1.2 Alternative course

troncamento dei TaskName se la stringa scritta supera la dimensione fissata nel documento di specifica, allora troncala secondo cdns.

troncamento del WBS identifier se avessi un identifier troppo lungo dovrei fare lo stesso??

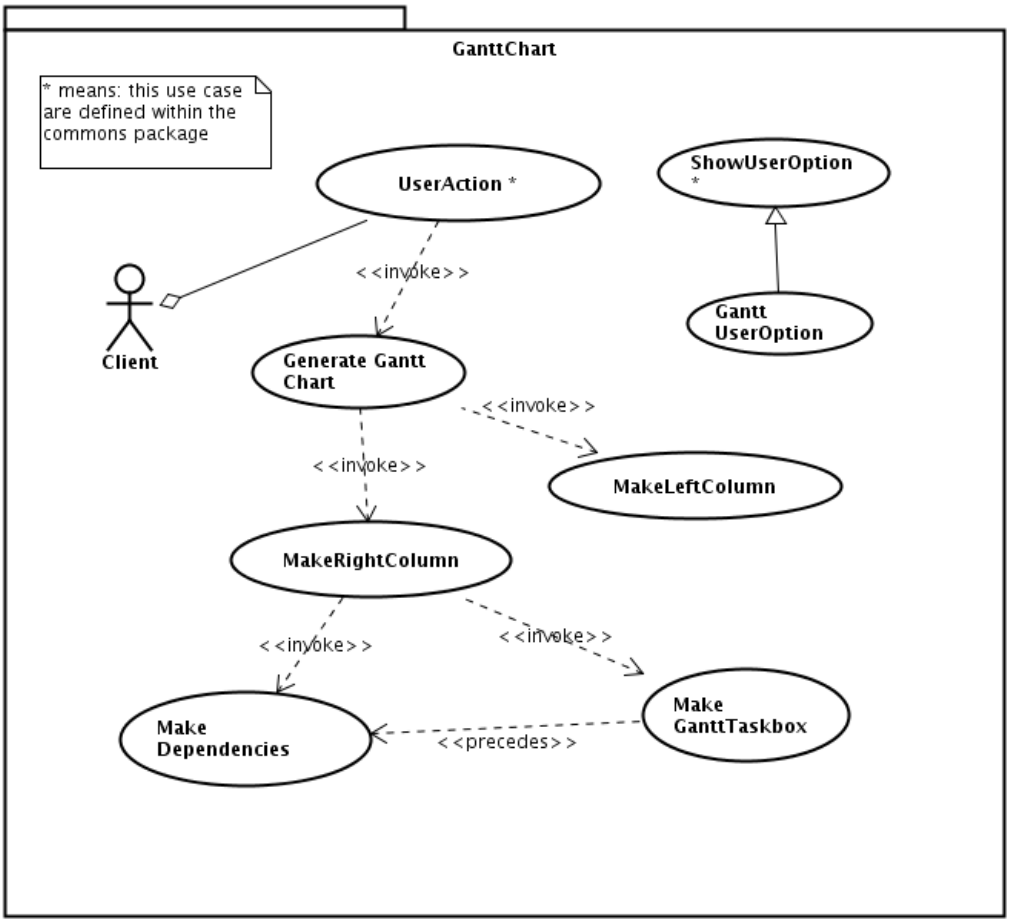

Figure 7.1: Gantt Overall UML diagram

7.2 Make GanttTaskbox

7.2.1 Basic course

Il client prende un reference al GanttChartGenerator e domanda di creare la rappresentazione grafica di un *Task*.

Il sistema esegue questi passi:

1. recupera il *Task* da rappresentare
2. costruisce la rappresentazione grafica, il *GanttTaskBox* in base alla scelta *WBSTreeSpecification*.
3. È possibile codificare delle informazioni aggiuntive:

- se *UserOptionChoice* contiene *ResourcesDetailsOption*, allora sul margine destro della *GanttTaskBox* appendi la stringa contenente la lista delle risorse cdns.
  Altrimenti appendi sul margine destro l'effort cdns.

7.2.2 Alternative course

troncamento delle resourses se le informazioni testuali aggiuntive superano la dimensione definita nel documento di specifica, allora si troncano cdns.

7.3 Make Right Column

7.3.1 Basic course

Il client prende un reference al GanttChartGenerator e domanda di creare la colonna di destra del diagramma.

Il sistema esegue questi passi:

1. costruisce una colonna di larghezza di dimensione *fissa* che viene calcolata cdns.
2. Il sistema effettua una ricerca per calcolare l'insieme dei *Task* necessari da rappresentare nella colonna.
3. Per ogni *Task* trovato:
   - invoca lo use case 7.2
   - si posiziona il *GanttTaskBox* creato nella giusta posizione temporale in base alla scelta *TimeGrainOption*.
4. se *UserOptionsChoice* contiene *ShowDependencies* allora per ogni *TaskBox* rappresentata, invoca lo use case 6.1

7.3.2 Alternative course

troncamento delle resourses se le informazioni testuali aggiuntive superano la dimensione definita nel documento di specifica, allora si troncano cdns.

7.4 Generate Gantt Chart

7.4.1 Basic course

Il client richiede di generare un *GanttChart*.
Il sistema costruisce un oggetto in questo modo:

- invoca lo use case 7.1 per creare la colonna a sinistra.
- invoca lo use case 7.3 per creare la rappresentazione nel tempo (colonna destra del diagramma).

Chapter 8

WBS chart

Overall UML diagram

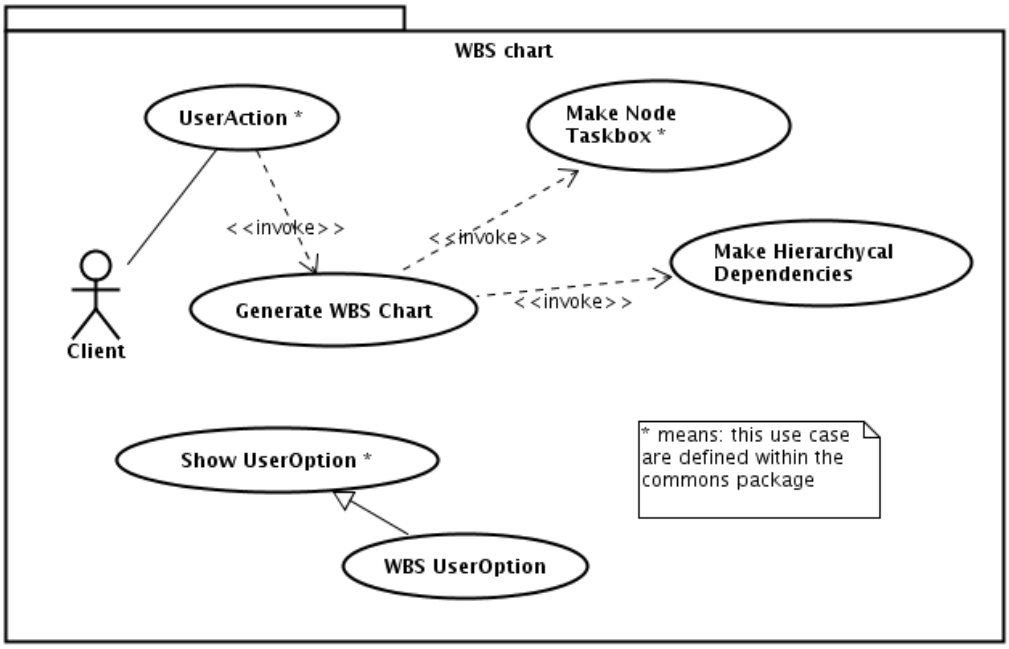

Figure 8.1: WBS UML diagram

8.1 Make Hierarchycal Dependencies

8.1.1 Basic course

Il client richiede la funzionalità di rappresentazione della relazione gerarchica di un task.
Il sistema esegue questi passi:

1. in base alle informazioni in *WBSTreeSpecification*, recupera i figli del task (non l'intera discendenza, solo quelli di livello successivo) e li dispone graficamente[1] al di sotto di esso (top-down)[2]
2. Il sistema collega il task ai suoi figli con linee spezzate cdns.

8.1.2 Alternative course

WBSExplosionLevel = 0 se il livello di visualizzazione della WBSStructure richiesto si limita alla root, crea il solo un *NodeTaskbox* rappresentante l'intero progetto.

8.2 Generate WBS Chart

8.2.1 Basic course

Il client richiede di generare un *WBSChart*.
Il sistema costruisce un oggetto in questo modo:

- invoca lo use case 6.2 per creare i *NodeTaskBox* del diagramma.
- invoca lo use case 8.1 per creare le dipendenze gerarchiche

[1] OSS. Non tutte le opzioni, in questa situazione hanno senso (resources detail?, alert marks?)
[2] effettivamente non e' che e' lui che li crea, dovrebbe solo aggiungerci le spezzate.

Chapter 9

TaskNetwork chart

Overall UML diagram

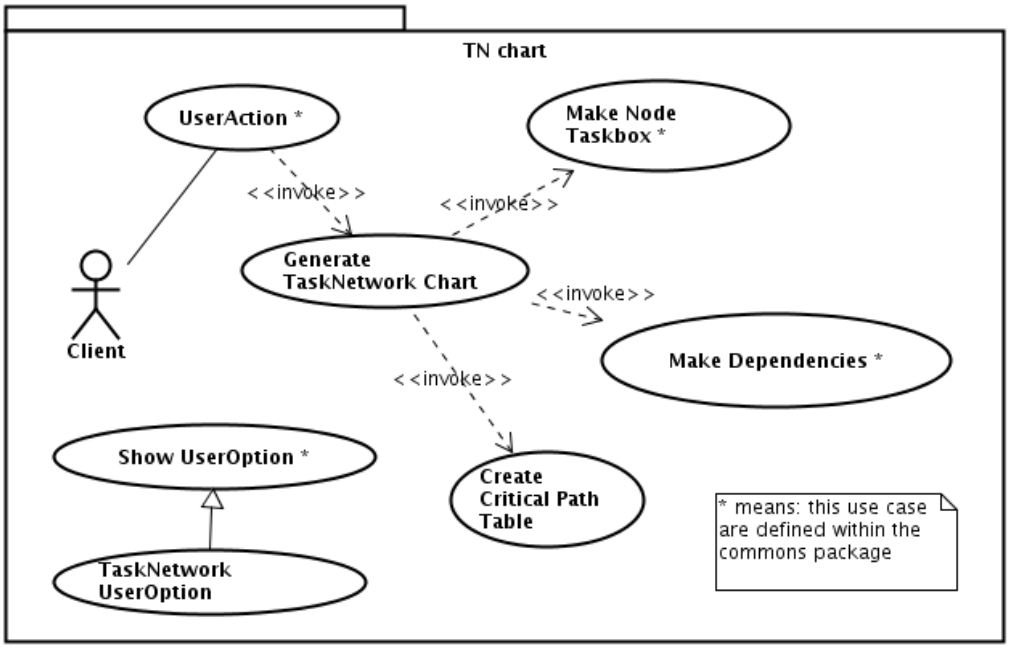

Figure 9.1: TaskNetwork UML diagram

9.1 Generate TaskNetwork Chart

9.1.1 Basic course

Il client richiede di generare un *TaskNetworkChart*.
Il sistema costruisce un oggetto in questo modo:

- invoca lo use case 6.2 per creare i *NodeTaskBox* del diagramma.
- si possono aggiungere le seguenti informazioni:
  - se *UserOptionsChoice* contiene *ShowCriticalPath* allora calcola il critical path rispetto al grafo rappresentato
  - se *UserOptionsChoice* contiene *ShowCriticalPathTable* allora invoca lo use case 9.2
  - se *UserOptionsChoice* contiene *ShowDependencies* allora per ogni *TaskBox* rappresentata, invoca lo use case 6.1

9.2 Create Critical Path Table

9.2.1 Basic course

Si assume che i critical path siano già stati calcolati e siano già rappresentati secondo le opzioni presenti in *UserOptionsChoice*.

Il client richiede di generare la tabella riassuntiva per i *critical path*.
Il sistema costruisce una tabella definita cosi: Ogni critical path calcolato rappresenta una

durata	**effort**	**ultimo gap**	**visualizzato**
...	...	...	*

entry nella tabella.

Part IV

Mockups

Release 1.0

Chapter 10

Gantt chart

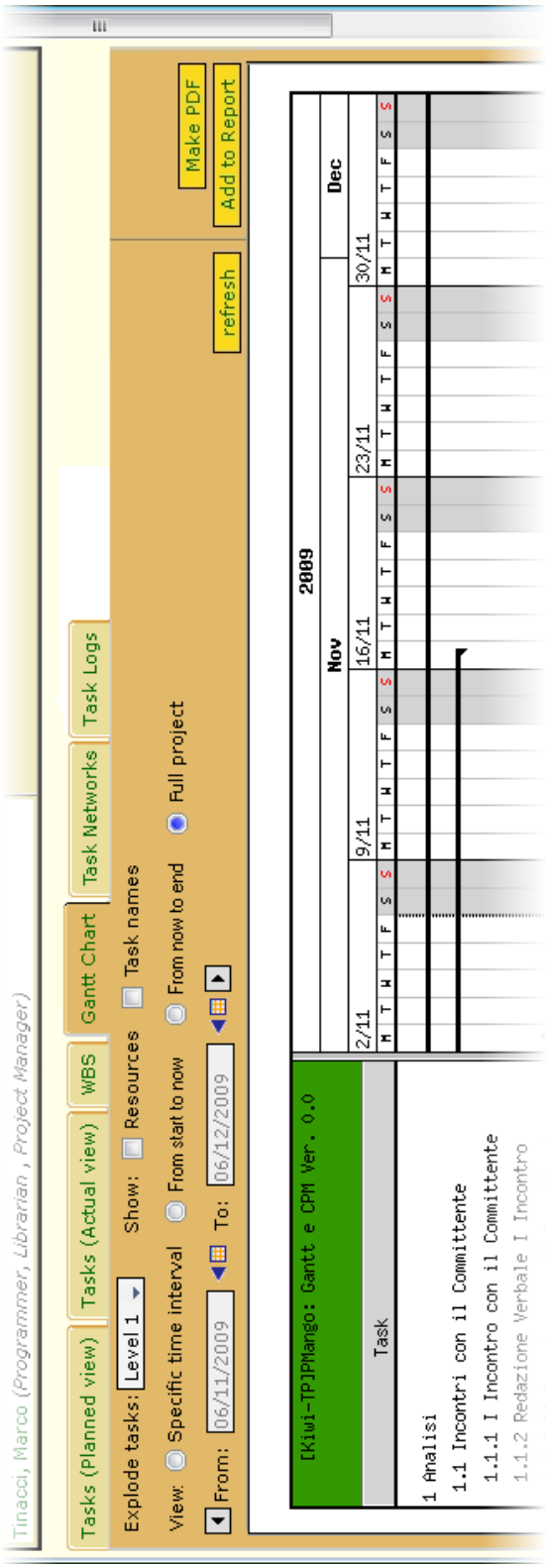

Figure 10.1: Gantt tab mockup

Chapter 11

WBS chart

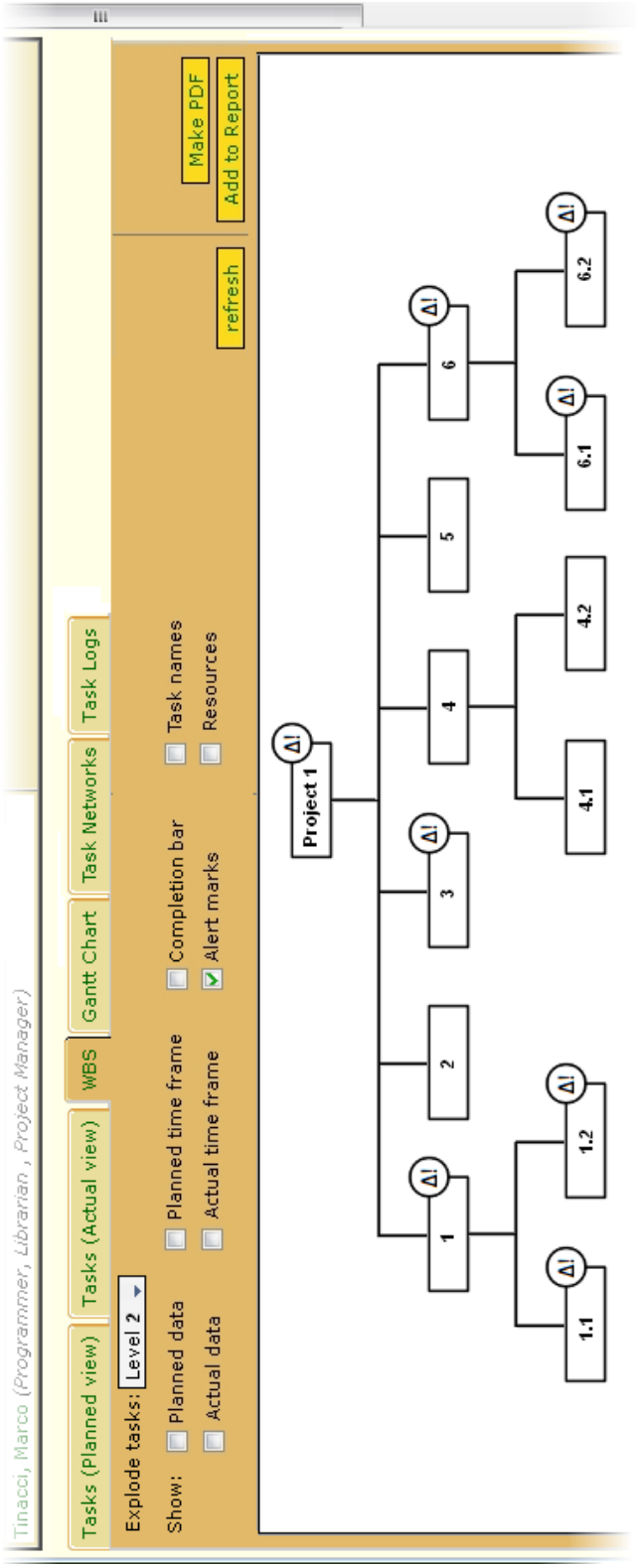

Figure 11.1: WBS tab mockup

Chapter 12

Task Network chart

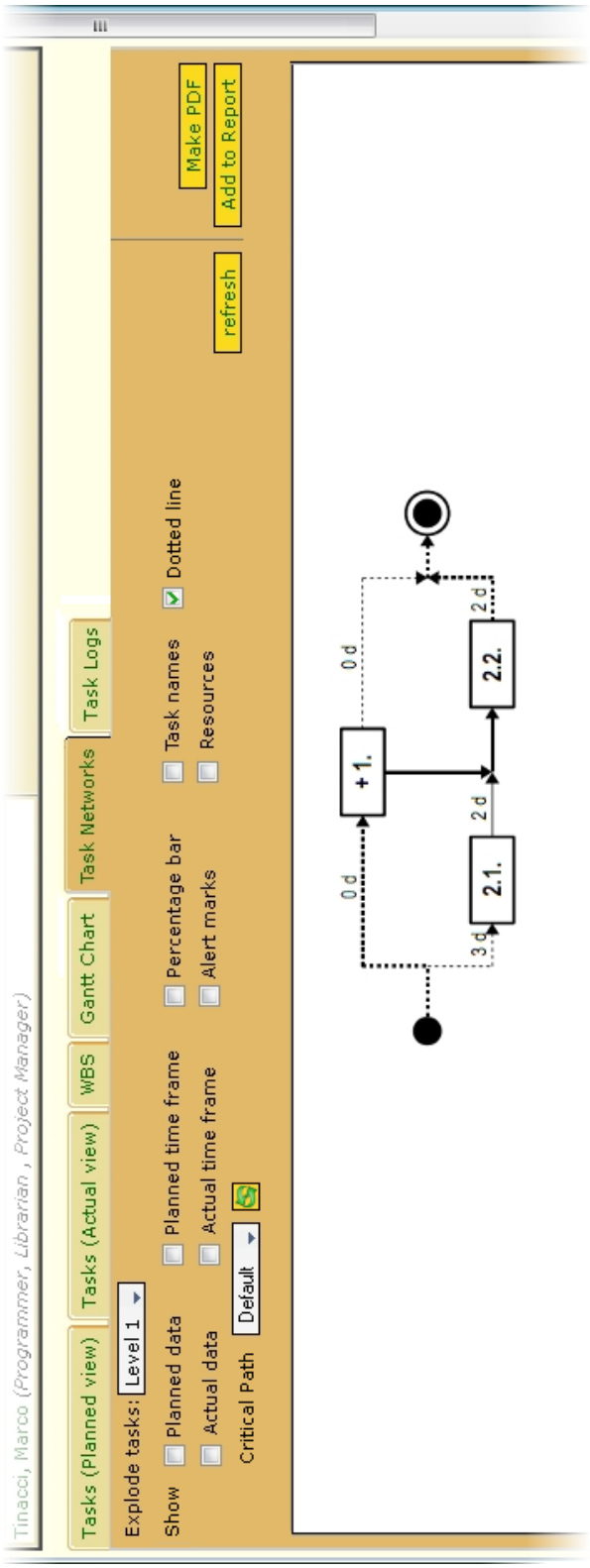

Figure 12.1: Task Network tab mockup