

# Domain Model

Release 0.6

November 15, 2009

Firenze



---

## Approvazione, redazione, lista distribuzione

approvato da	il giorno	firma
Marco Tinacci		

redatto da	il giorno	firma
Francesco Calabri		
Manuele Paulantonio		
Massimo Nocentini		

distribuito a	il giorno	firma
Daniele Poggi		
Niccoló Rogai		
Marco Tinacci		

## Contents

1	Overall diagram	5
2	Task	6
3	TaskBox	6
4	Strip	7
5	Chart	9

## **Introduzione**

# 1 Overall diagram

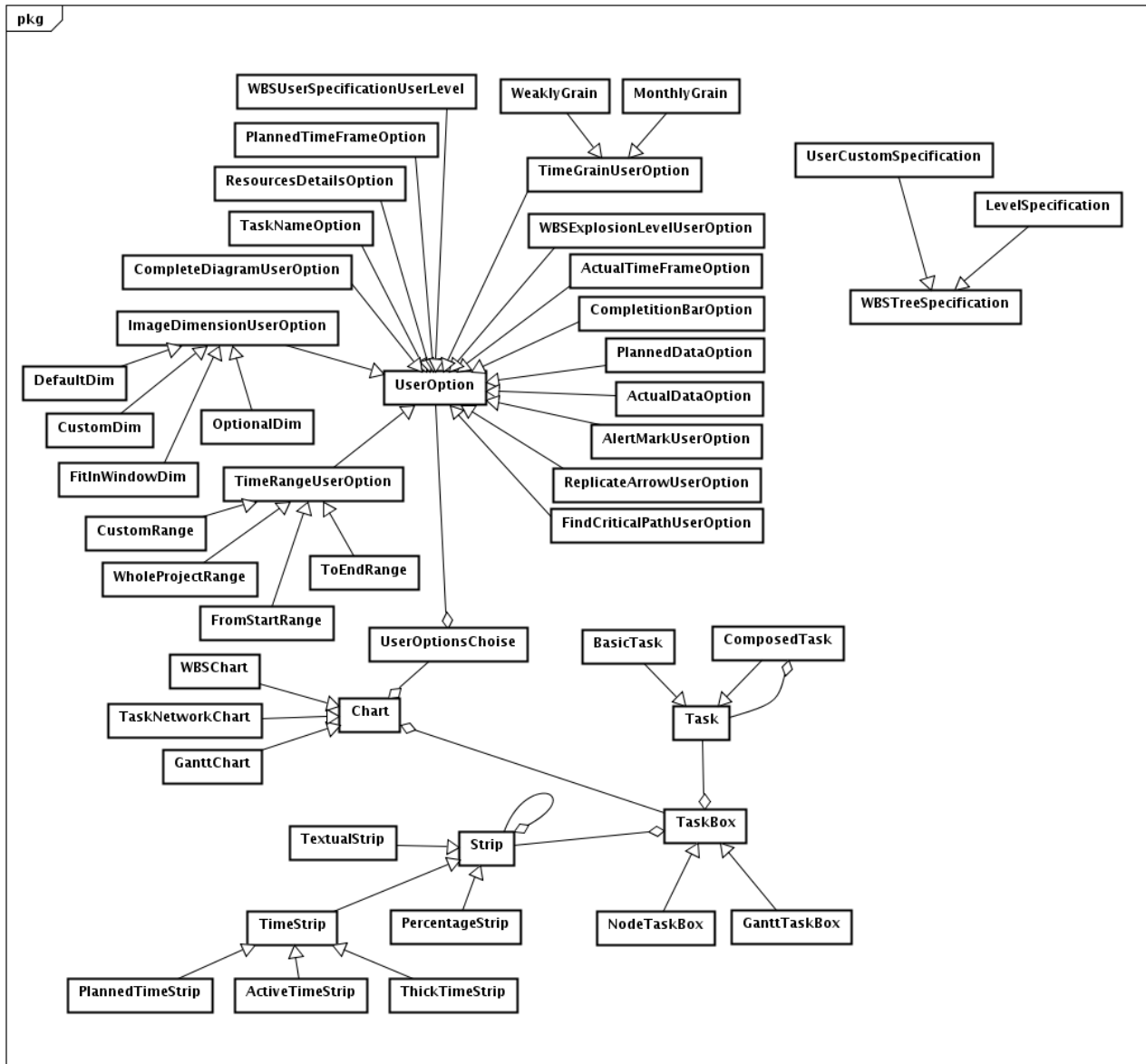


Figure 1: Overall UML diagram

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.

## 2 Task

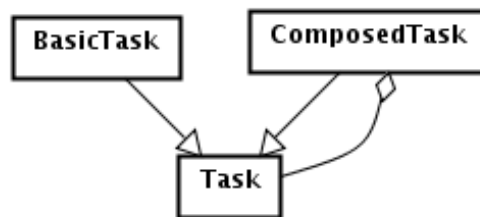


Figure 2: task and its relations

Molto probabilmente il concetto di *Task* esiste già 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 già esistente.

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

Costruiamo la relazione  $\rightarrow$  che lega questi due concetti:

- *BasicTask*  $\rightarrow$  attività di base, non ulteriormente scomponibili
- *ComposedTask*  $\rightarrow$  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 così le abbiamo portate ad avere interfacce compatibili.

## 3 TaskBox

Il *TaskBox* è la rappresentazione grafica di un *Task* (Figure 2). Questo concetto astrae su queste specializzazioni:

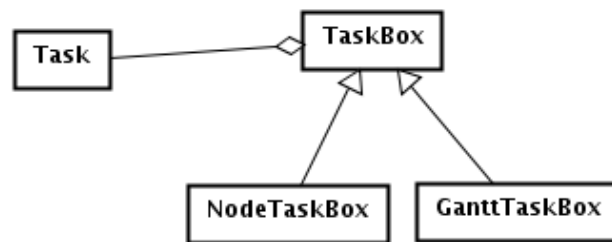


Figure 3: taskbox and its specializations

- *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 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.

**russian doll** *Strip* contiene a sua volta delle *Strip*: questo è un concetto che pensiamo possa 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 così l'effetto di *russian doll*, dedicandoci a modellare solo alcune semplici

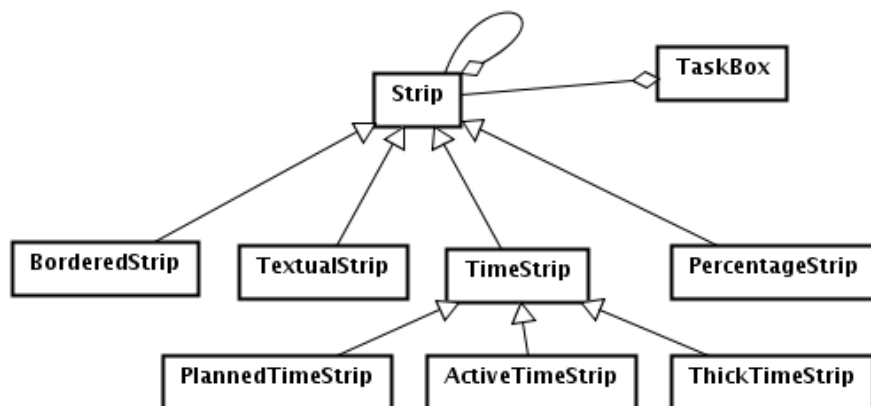


Figure 4: kinds of strips

specializzazioni di *Strip*, necessarie per l'implementazione delle notazioni richieste nel documento di specifica, limitandoci poi ad ottenere rappresentazioni complesse anidando quelle semplici.

**specializations** abbiamo un primo livello di specializzazione:

- *PercentageStrip* rappresenta una quantità (l'intera *Strip*) e la percentuale di completamento (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



## 5 Chart

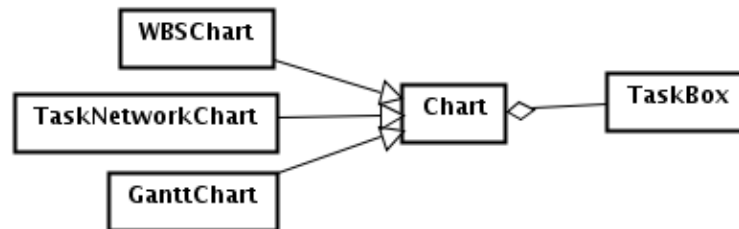


Figure 5: chart and building blocks