Disegno del sistema

Release 0.6

December 8, 2009
Firenze

Approvazione, redazione, lista distribuzione

approvato da	**il giorno**	**firma**
Marco Tinacci	December 8, 2009	

redatto da	**il giorno**	**firma**
Francesco Calabri	December 8, 2009	
Niccolò Rogai	December 8, 2009	
Marco Tinacci	December 8, 2009	

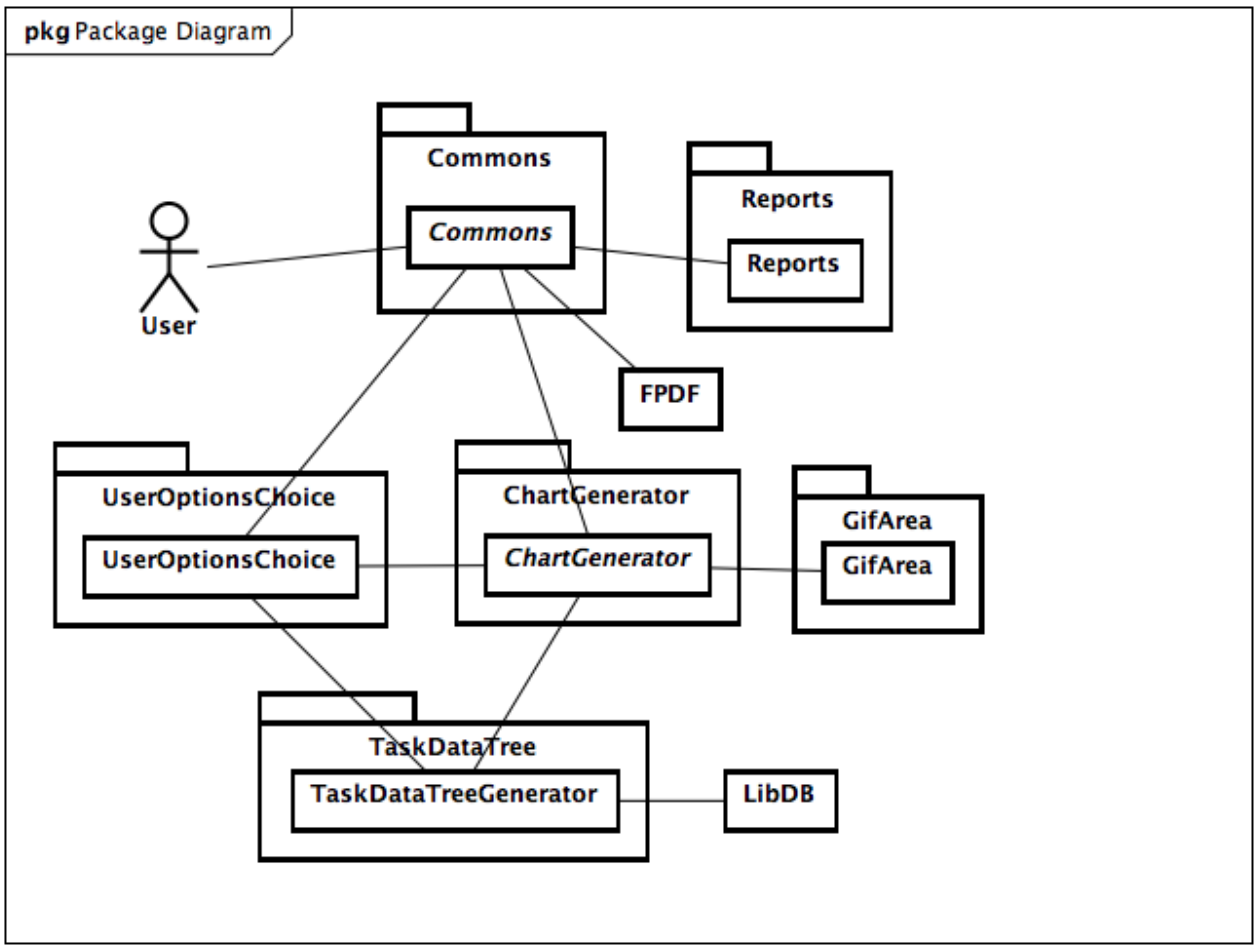
distribuito a	**il giorno**	**firma**
Manuele Paulantonio	December 8, 2009	
Daniele Poggi	December 8, 2009	
Massimo Nocentini	December 8, 2009	

Contents

0.1 Introduzione

In questo documento viene illustrata l'architettura del modulo tramite le viste dei *class diagrams* e i *sequence diagrams*, come evoluzione naturale, rispettivamente di *domain model* e *use cases*.
I diagrammi delle classi servono quindi a individuare le strutture dei dati e i metodi assiociati mentre quelli di sequenza evidenziano le comunicazioni che intercorrono tra le entità.
Per semplificare la rappresentazione del diagramma delle classi abbiamo deciso di scrivere gli attributi privati con la notazione di quelli pubblici "+ attribute" dando per assunto che siano provvisti dei metodi pubblici di "getAttribute()" e "setAttribute()".
Quando si fa riferimento a un file, a meno che non sia espressamente specificato, è sempre un path che inizia implicitamente dalla root path dove PMango è installato.

0.2 Descrizione architetturale



Il diagramma propone una visuale ad alto livello delle componenti principali del modulo.

- Commons: questa componente è quella che controlla il flusso principale e si interfaccia direttamente con l'utente (la pressione dei pulsanti dell'interfaccia utente corrisponde a una chiamata di un metodo). Utilizza la componente ChartGenerator per generare i diagrammi richiesti e UserOptionsChoice per gestire la visualizzazione e l'acquisizione delle opzioni utente presenti nella form della pagina. Per includere il diagramma in un PDF è necessario che comunichi con la libreria FPDF, mentre per aggiungere i diagrammi al report deve utilizzare la componente Reports.
- ChartGenerator: questa componente si occupa della generazione dei diagrammi. L'elaborazione grafica viene delegata alla componente GifArea, l'acquisizione dei dati a TaskDataTreeGenerator, e le opzioni utente vengono lette da UserOptionsChoice.
- TaskDataTreeGenerator: questa componente si occupa della generazione di una strut-

tura dati contenente tutte le informazioni dei task di cui è composto il progetto. Si interfaccia con il database per il reperimento dei dati utilizzando la classe già presente in PMango *DBQuery*. Le richieste di quali dati verranno richiesti al database dipendono dalle opzioni specificate in UserOptionsChoice.

- UserOptionsChoices: questa componente gestisce i campi di un array associativo che contiene le opzioni scelte dall'utente.
- GifArea: componente che contiene utilità per l'elaborazione di immagini gif. Queste funzionalità si collocano in un livello medio tra la generazione del diagramma completo e il disegno di segmenti e label.
- Reports: questo componente si occupa di aggiungere il diagramma prodotto nel report.

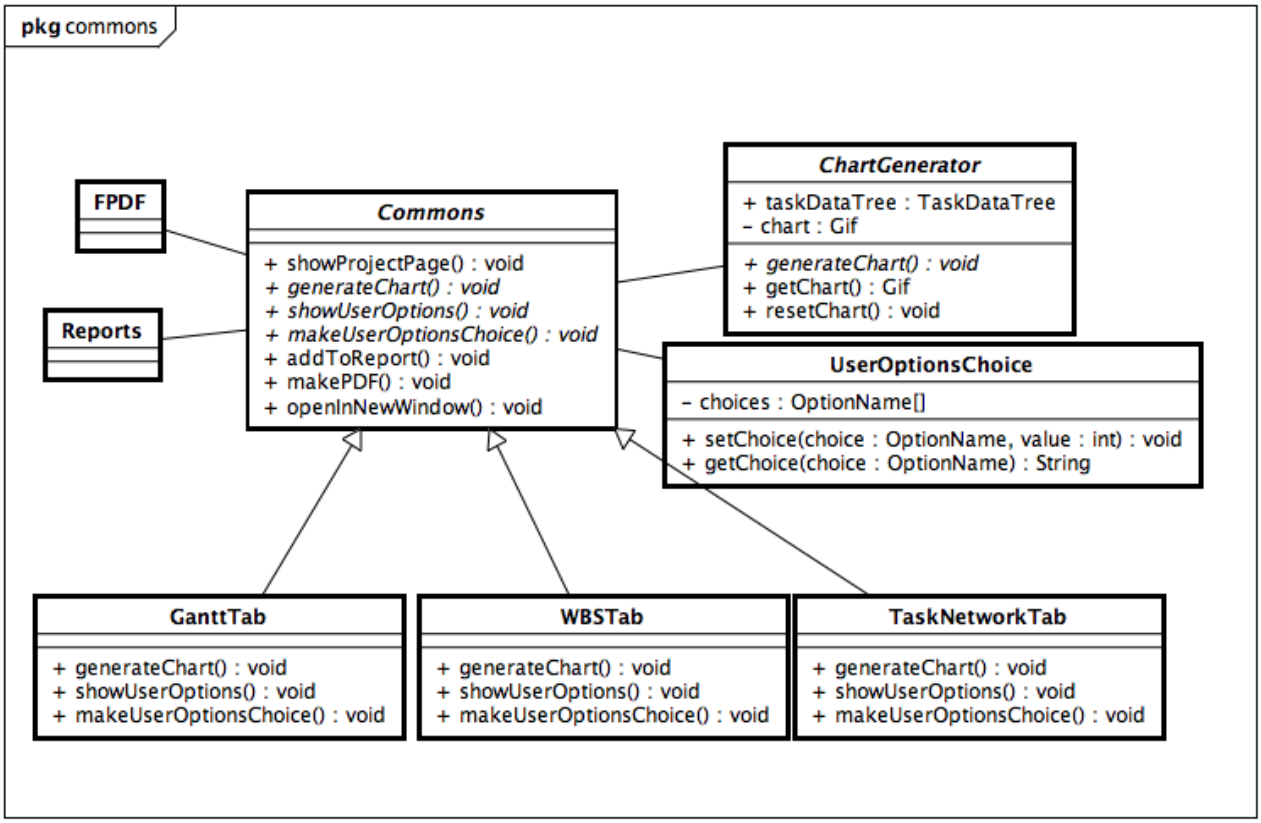
Le dipendenze esterne al modulo sono:

- FPDF: la libreria per la generazione dei PDF "lib/fpdf/fpdf.php"
- DBQuery: la classe per reperire le informazioni dal database "classes/query.class.php"

Chapter 1

Class Diagrams

1.1 Commons



La classe Commons racchiude tutte le funzioni principali del modulo. Utilizza le classi

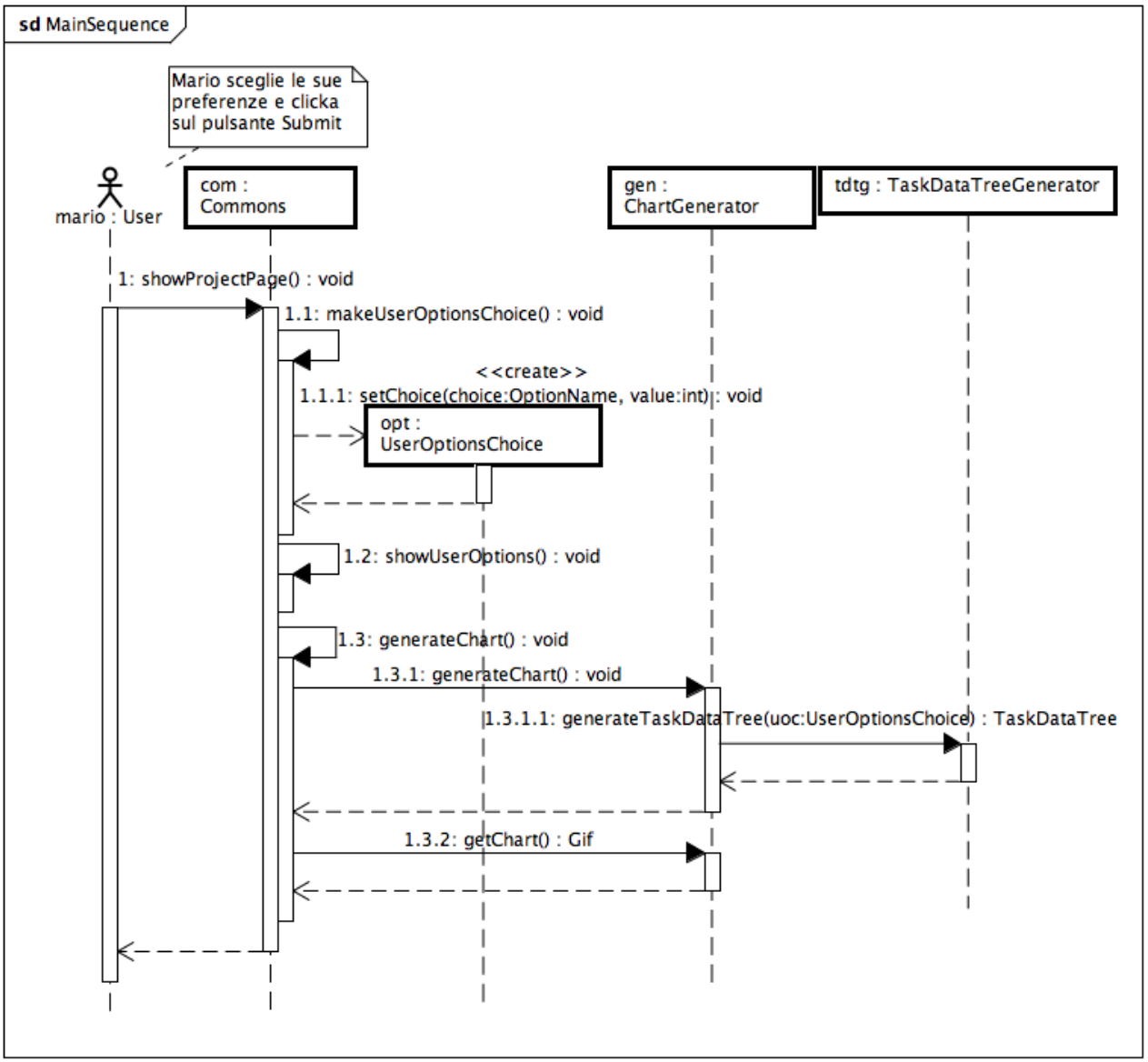
- ChartGenerator

- UserOptionsChoice
- Reports
- FPDF (esterna al modulo e già utilizzata in PMango)

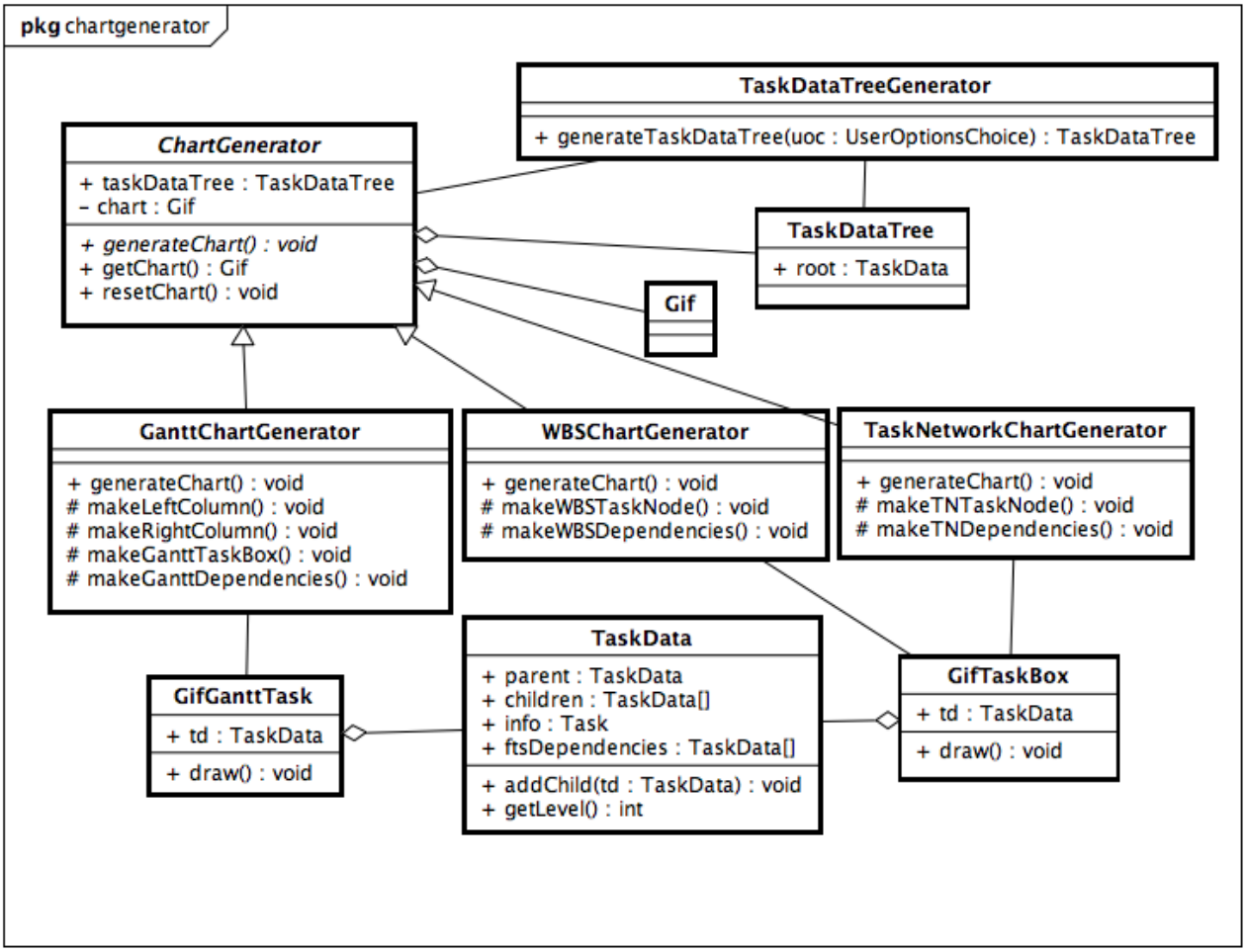
La classe è astratta a causa dei metodi astratti *generateChart*, *ShowUserOptions* e *makeUserOptionsChoice* (che dovranno essere implementati nelle successive estensioni). In generale i metodi contenuti nella classe Commons sono quelli presentati nel documento dei casi d'uso, contentente una descrizione più approfondita dei A seconda di quale tab viene caricato, può essere usata l'estensione di Commons corrispondente:

- GanttTab
- WBSTab
- TaskNetworkTab

che implementano la generazione del diagramma, la creazione della maschera delle opzioni utente e l'acquisizione delle scelte dei parametri. Ovviamente la generazione di un diagramma specifico dovrà rifarsi alla corrispondente specializzazione del generatore dell'immagine, infatti GanttTab utilizzerà GanttChartGenerator, WBSTab utilizzerà WBSChartGenerator e TaskNetworkTab utilizzerà TaskNetworkChartGenerator.
Di seguito viene riportato il diagramma di sequenza che illustra il procedimento di richiesta della pagina contenente il diagramma richiesto.


sd MainSequence
Mario sceglie le sue preferenze e clicka sul pulsante Submit
mario : User
com : Commons
gen : ChartGenerator
tdtg : TaskDataTreeGenerator
1: showProjectPage() : void
1.1: makeUserOptionsChoice() : void
<<create>>
1.1.1: setChoice(choice:OptionName, value:int) : void
opt : UserOptionsChoice
1.2: showUserOptions() : void
1.3: generateChart() : void
1.3.1: generateChart() : void
1.3.1.1: generateTaskDataTree(uoc:UserOptionsChoice) : TaskDataTree
1.3.2: getChart() : Gif

1.2 Chart Generator



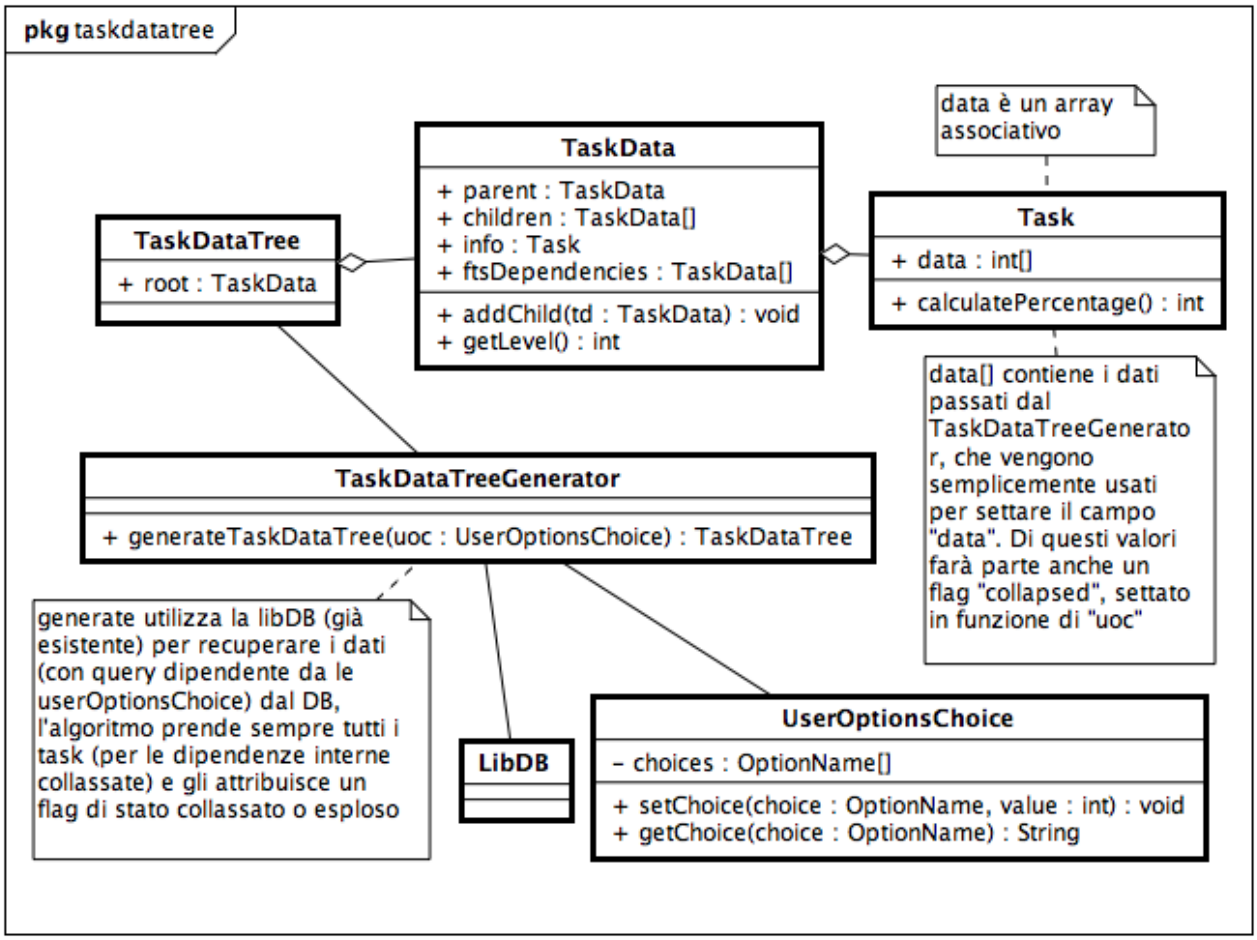
Questa componente contiene le seguenti classi:

- ChartGenerator: la classe specializzata nella generazione della gif contenente l'illustrazione del diagramma, è astratta per il metodo generateChart che verrà specializzato nelle sue specializzazioni.
- GanttChartGenerator, WBSChartGenerator, TaskNetworkChartGenerator: specializzazioni di ChartGenerator che implementano il metodo generateChart. Seguendo la traccia descritta nei casi d'uso (capitoli 7,8 e 9 del documento di analisi) il metodo si compone delle procedure di makeLeftColumn e makeRightColumn che a sua volta utilizzerà i metodi makeGanttTaskBox e makeGanttDependencies per generare il diagramma. Il WBSChartGenerator a sua volta utilizzerà makeWBSTaskNode e makeWBSDependencies per creare nodi e dipendenze, e in modo analogo si comporterà la classe TaskNetworkChartGenerator. Mentre il generatore del gantt utilizza GifGanttTask per disegnare

i nodi, i generatori di WBS e TaskNetwork usano GifTaskBox, in quanto la rappresentazione grafica è la stessa nei due casi.

- TaskDataTreeGenerator, TaskDataTree, TaskData: queste classi sono presenti nella componente TaskDataTree.
- GifGanttTask, GifTaskBox: queste classi sono presenti nella componente GifArea.

1.3 Task Data Tree

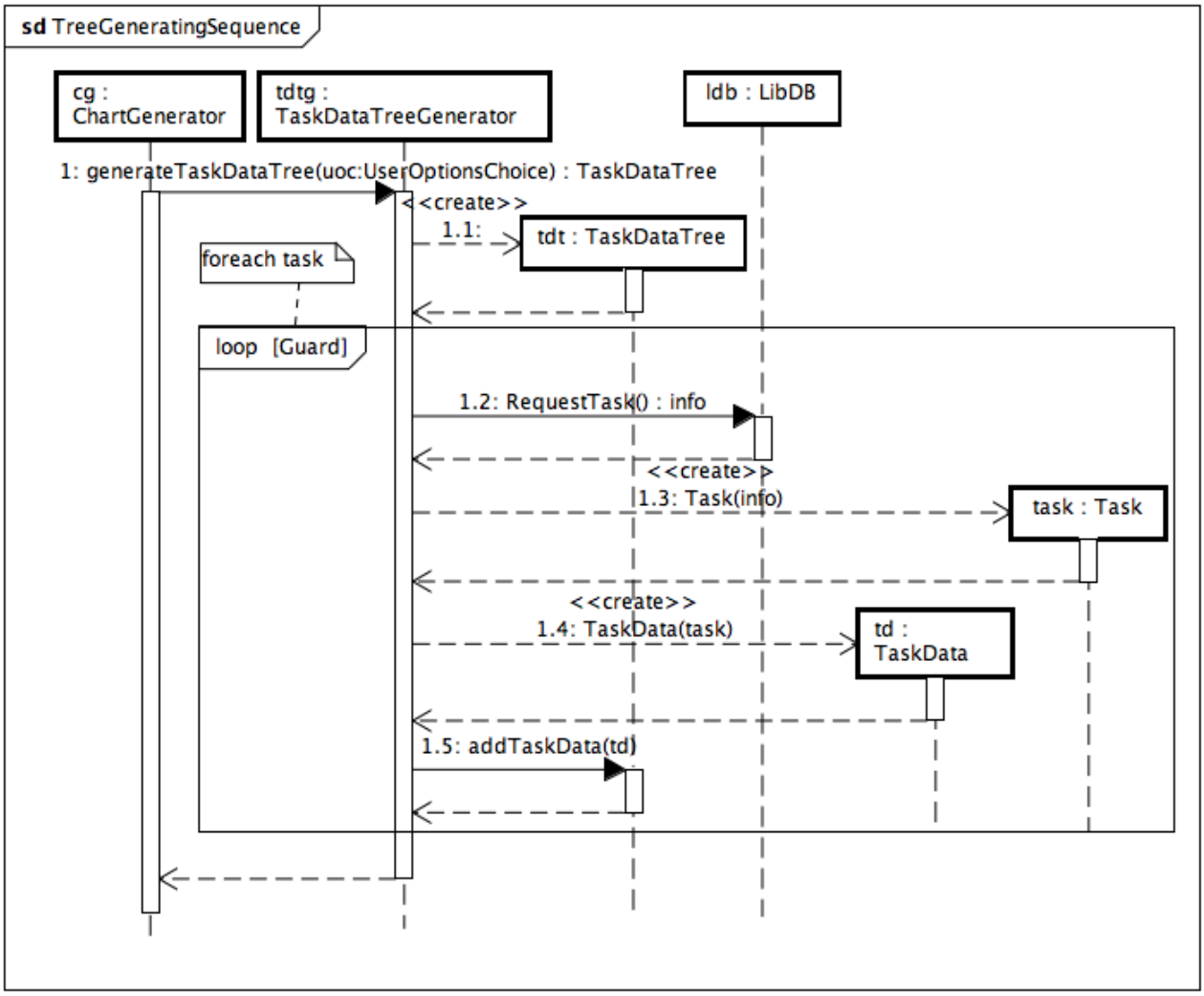


L'obiettivo di questa componente è quello di generare una struttura dati che contenga tutte le informazioni dei task che sono interessanti al fine della costruzione del diagramma specifico.

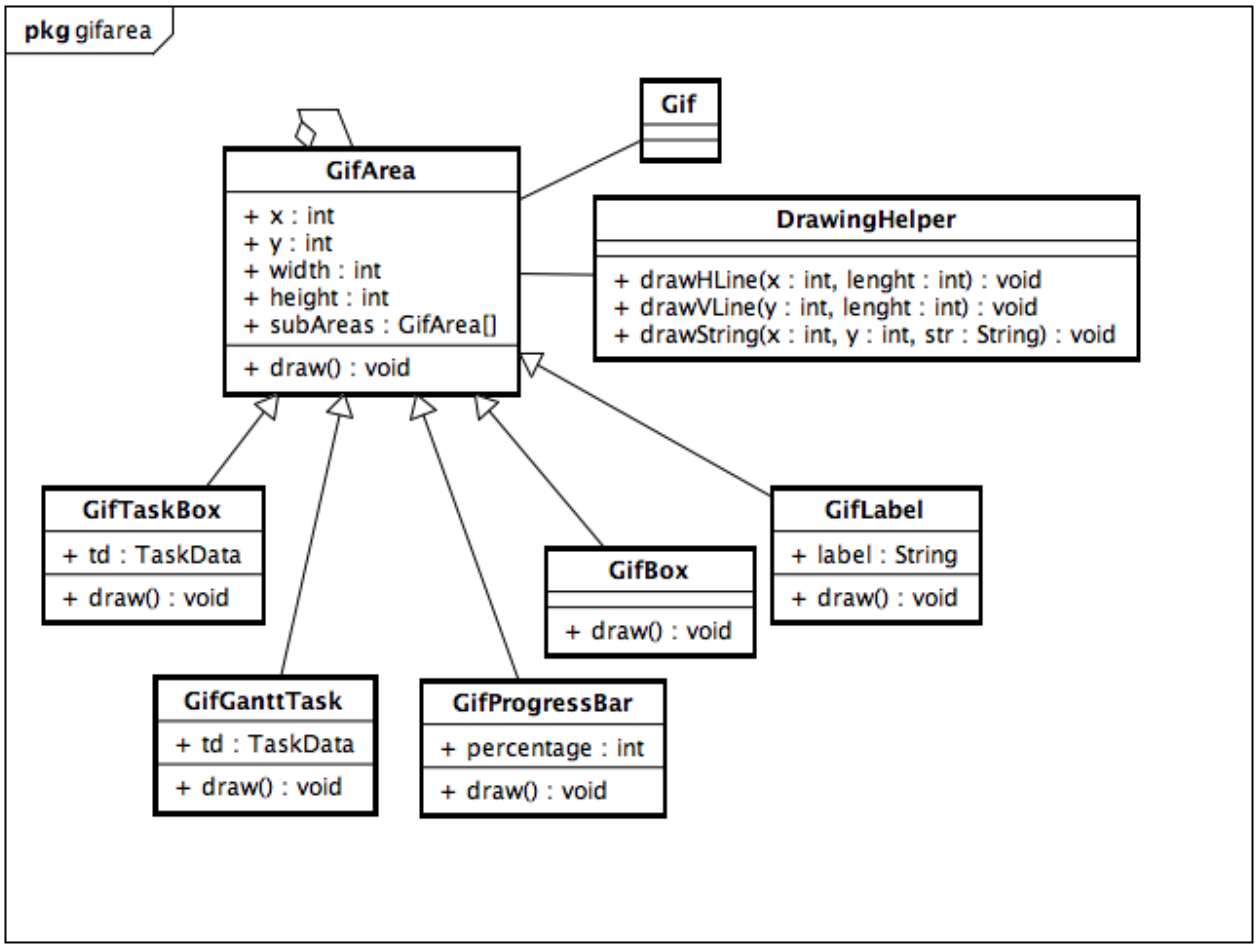
- TaskDataTreeGenerator: questa classe ha il compito di costruire l'albero dei tasks (contenente anche le dipendenze *finish-to-start*) richiedendo alla libDB i dati specificati nelle

opzioni utente. L'oggetto generato sarà un TaskDataTree.

- TaskDataTree: classe albero ad arietà non fissata, contiene solo il nodo radice. I nodi in questione sono TaskData.
- TaskData: sono i nodi del TaskDataTree, hanno un riferimento al padre e al vettore (dinamico) dei figli. È presente anche un vettore di nodi TaskData che indica le dipendenze finish-to-start. L'informazione contenuta nel nodo è di tipo Task.
- Task: contiene le informazioni di base del task, memorizzate nell'array associativo *data*

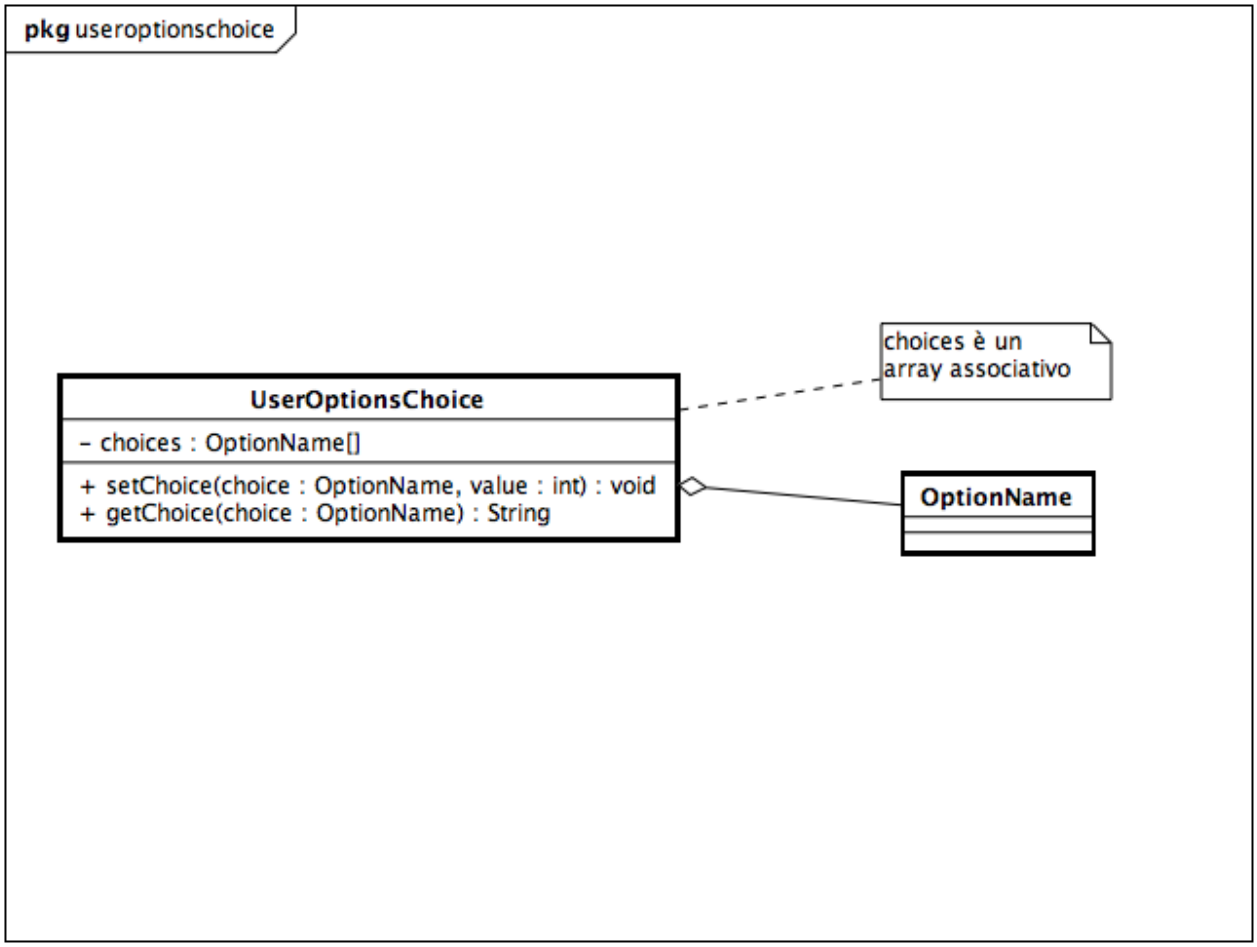


1.4 Gif Area



La GifArea e i suo derivati realizzano il pattern *composite* che si presta a portare ad un livello più alto la gestione della grafica, infatti qualunque cosa che sia una "GifArea" avrà la possibilità di essere stampata (tramite la funzione *draw*) su una gif in un determinato punto (x, y) occupando una certa area di dimensioni "width" e "height". I componenti più base per maneggiare la grafica in questo contesto sono la "GifLabel" che è la rappresentazione grafica di un testo, la "GifBox" che è una rappresentazione grafica di un rettangolo, la "GifProgressBar" che che rappresenta una barra di completamento. Anche i GifTaskBox sono una GifArea e sono composti da dei GifBox con al suo interno delle GifLabel, e da una GifProgressBar quando richiesta, e il draw del GifTaskBox farà il draw di tutti questi suoi componentini interni. Una volta costruito l'albero dei taskdata sarà molto facile disegnare un GifTask da un taskdata. Così si separano la visualizzazione dei taskbox dal problema del posizionamento e l'unione con le linee.

1.5 User Options



- UserOptionsChoice: la classe che racchiude le scelte fatte dall'utente in un array associativo controllato dai metodi get e set.
- OptionName: un'enumerate che vincola l'accesso all'array associativo ai soli campi specificati. Questa contiene le seguenti voci (associate ai possibili valori che possono assumere):
  - TaskNameOption: boolean
  - PlannedTimeFrameOption: boolean
  - ActualTimeFrameOption: boolean
  - PlannedDataOption: boolean
  - ActualDataOption: boolean

- CompletitionBarOption: boolean
- AlertMarkUserOption: boolean
- WBSExplosionLevelUserOption: numeric
- WBSUserSpecificationUserLevel: boolean
- ReplicateArrowUserOption: boolean
- FindCriticalPathUserOption: boolean
- ResourcesDetailsOption: boolean
- CompleteDiagramUserOptions: boolean
- WBSTreeSpecification: LevelSpecification, UserCustomSpecification
- TimeGrain: HourlyGrain, DailyGrain, WeaklyGrain, MonthlyGrain, AnnualyGrain
- ImageDimensionUserOption: CustomDim, FitInWindowDim, OptionalDim, DefaultDim
- TimeRangeUserOption: CustomRange, WholeProjectRange, FromStartRange, ToEndRange

La semantica di questi parametri è specificata nel domain model. Tutti i parametri boolean sono intesi $false$ se non presenti, è quindi necessario un doppio controllo per capire se è impostato il valore $true$: il controllo sulla presenza del campo ed eventualmente sul suo valore.

1.6 Reports