TaskJuggler, initiation

Jean-Marc LICHTLE

16 octobre 2007

Table des matières

1 Introduction

1.1 Présentation de TaskJuggler

J'ai cherché longtemps un logiciel de planification qui soit capable de réaliser une gestion de planning mais aussi de collecter les coûts et de les concaténer. Dans ma quête il n'était, évidemment, pas question de sortir du monde libre. C'est ainsi qu'après bien des essais mon choix s'est arrêté sur taskjuggler.

1.1.1 Version, environnement

La version présentée ici est la 2.3.1 disponible dans les paquets de LINUX UBUNTU 7.04.

1.1.2 Aide, manuel

C'est presque une coutume dans le monde libre, les manuels sont souvent d'excellente qualité. De ce point de vue TaskJuggler est tout à fait dans le ton, l'aide accessible par F1 est très bien faite. Un petit regret, l'aide contextuelle via F2, pourtant décrite dans la documentation, ne fonctionne pas chez moi. Bug ou mauvaise compréhension de la langue des BEATLES ? Ah oui, évidemment rien n'est rédigé dans la langue de MOLIERE, mais ça vous vous en doutiez un peu non ?

Le présent document ne prétend pas se substituer à la documentation officielle. Tout au plus ai-je l'espoir, par quelques notes jetées sur le papier, de mettre le pied à l'étrier à une personne qui s'intéresserait à TaskJuggler.

1.1.3 Ergonomie

Il importe avant toute chose d'aborder le point de l'ergonomie. Si vous aimez allonger des tâches en tirant sur une poignée formée au bout de la barre, si vous souhaitez faire des liaisons d'une simple traînée de pointeur passez votre chemin, TaksJuggler, malgré ses qualités, n'est pas fait pour vous.

Si par contre vous êtes familier de l'utilisation de softs tels que LaTeX, si l'écriture d'un code source dans un language avec lequel vous allez devoir vous

familiariser ne vous fait pas fuire alors vous allez peut-être bientôt partager mon intérêt pour TaskJuggler.

1.2 Lancement

1.3 Composants

Le paquet TaskJuggler installe deux logiciels :

- taskjuggler (tout en minuscule) qui est la version ligne de commande
- TaskJugglerUI, en est la version graphique. L'installation via la procédure classique dans UBUNTU ajoute une entrée de menu TaskJuggler dans Application / Bureautique.

La version ligne de commande diffère de la version graphique dans sa présentation (évidemment !) et dans sa portée. En ligne de commande la recompilation remet à jour l'ensemble des documents de sortie, rapport, fichier html, exportation. En version graphique seul le document affiché est actualisé.

1.4 Points d'achoppement

Autant ne pas laisser traîner le sujet dans le corps du texte. Je souhaite regrouper ici les quelques points qui m'ont personnellement posé problème lors des premiers essais et les solutions que j'y ai trouvé.

1.4.1 Recompilation en mode graphique

Ca a l'air stupide mais pour les premiers essais j'ai utilisé alternativement la version ligne de commande pour recompiler et la version graphique pour visualiser les résultats. Je vous donne donc le truc : C'est l'icône en forme d'horloge qui lance la recompilation ! Allez savoir pourquoi une horloge ...

2 Mise en oeuvre

La mise en oeuvre passe par l'édition du fichier source. L'interface graphique n'offre que des fonctionnalités permettant de modifier l'apparence de l'affichage. On peut ainsi choisir l'échelle des temps pour le diagramme de GANTT, réduire l'affichage des sous tâches d'une tâche principale etc mais aucune modification d'un des paramètres tels que date, dépendance entre tâches etc n'est possible en mode graphique. Tout ceci se passe en mode texte. Cliquer à cet effet sur l'onglet "Editor".

FIG. 1 – L'icône représentant la petite horloge est celle qui commande la recompilation !

2.1 Création d'un nouveau projet

On peut imaginer partir ex-nihilo d'une page blanche mais TaskJuggler met à disposition des modèles de complexité croissante. Sélectionnez File / New, TaskJuggler vous propose de nommer le nouveau projet après quoi, le nommage étant fait, TJ vous propose de choisir entre trois modèles, blank, simple et large. Le modèle blank n'est pas tout à fait vide, il comporte simplement une entête de projet. Le modèle simple est, à mon avis, le plus utile ici, le modèle large n'apporte rien dans un premier temps, tout au plus pourra-t-il servir de référence au lecteur qui souhaitera diversifier ses sources d'information.

Dans la suite je vais détailler les primitives qui me semblent les plus intéressantes. Je suis parti pour faire ce travail du modèle simple que j'ai un peu francisé et complèté.

Le lecteur trouvera plus loin une explication sur la façon de franciser un modèle de façon permanente.

2.2 Entête de projet

La ligne définissant l'entête de projet est assez simple, par exemple :
project xxx 'xxxProject' '1.0' 2007-09-01 - 2008-09-01 {
}

Les accolades contiennent les éléments d'entête, nous détaillerons les plus courants dans la suite. La ligne d'introduction comporte des informations importantes notamment le nom abrégé du projet, son nom complet qui ap-

paraitra en permanence dans l'entête de la fenêtre TaskJuggler et les dates limites dans lesquelles seront contenus tous les constituants du projet.

2.2.1 timingresolution

Tout en début d'entête, avant toute autre primitive travaillant sur une échelle de temps je recommande de placer la primitive timingresolution qui précise avec quelle résolution de temps TJ va calculer sa planification. Deux choses importantes :

- Ne pas choisir une valeur trop faible (mini 5 min) de telle sorte à ne pas surcharger la mémoire lors des recalcus faits par TJ
- La valeur par défaut est 1h, elle peut convenir sauf si, par exmple, les horaires de travail comporte des demi-heures, par exemple sortie le soir à 17h30. Dans ce cas il faut fixer la résolution à 1/2 heure faute de quoi on verrait apparaître un message d'erreur.

2.2.2 timeformat

On devinera aisément que cette primitive permet de fixer le format de la date. Ce format sera en fait surtout utilisé pour les sorties graphiques. La syntaxe :

timeformat "%d-%m-%Y"

qui est la valeur par défaut va afficher les dates sous la forme 16-07-1954. Pour un diagramme de GANTT je pense qu'il est plus efficace d'afficher la date sous la forme ven 16-07-54 qui précise le jour de la semaine et raccourci l'indication de date pour gagner un peu de place. Cet affichage est obtenu avec la syntaxe :

timeformat "%a %d-%m-%y"

ATTENTION ! Par défaut les modèles précisent, dans les sections définissant les sorties comme le diagramme de GANTT, les formats de date à utiliser. Si vous conservez ces primitives et réglez le format à cet endroit vous risquez de ne jamais comprendre à quoi sert le timeformat situé dans l'entête et qui est applicable à l'ensemble des sorties.

2.2.3 currency

Fixe la monnaie à utiliser. Par défaut :
currency "EUR"
On peut remplacer par
currency "€",
ça semble fonctionner correctement.

2.2.4 scenario

Initialisation des scénarios :

La syntaxe est assez simple et montre bien la subordination entre les scénarios :

scenario plan 'Plan' {
scenario real 'Réalisé'
}

Ces lignes font partie de l'entête du projet.

La syntaxe montre que le scénario Réalisé est une variante du scénario Plan. Il ne peut en fait y avoir qu'une seule définition de scénario principal, par contre vous pouvez définir plusieurs scénarios fils, voir même décliner les fils en d'autres scénarios. Après tout si vous arrivez à vous y retrouver tant mieux !

Mise en oeuvre de la notion de scénario :

La mise en oeuvre est simple. Supposons que dans l'une des tâches vous ayez une affectation d'effort comme suit :

effort 5d

Cette indication est relative au scénario de base, le scénario principal.

Dans un scénario secondaire, qui peut être le suivi du déroulement réel du projet vous constatez que la tâche met en fait six jours. Ajoutez donc :

real :effort 6d

sous la ligne précédente et le tour est joué. 5d sera l'hypothèse de base, 6d celle du scénario 'real' que vous aurez déclaré dans l'entête du projet.

Exploitation des scénarios :

Certains rapports, resourcereport et taskreport, ne peuvent prendre en compte qu'un seul scénario à la fois. Une ligne :

scenario real

dans la définition du rapport permet d'affecter un scénario.

D'autres, notamment htmlaccountreport sont très à l'aise dans la présentation comparée des deux ou plusieurs scénarios. Dans ce cas la ligne devient :

scenarios plan, real

Notez le 's' !

2.2.5 minslackrate

Mwouais ! La ça se complique. Valeur par défaut 15, ce qui signifie 15%. Chaque tâche est examinée en fonction des éléments de sa planification et des possibilités qu'elle a de glisser sans compromettre l'ensemble de la construction du projet. Si cette possibilité de glisser mesurée par rapport à la lonueur totale de la branche dans laquelle la tâche est incluse est supérieure à la

valeur du paramètre slackrate alors la tâche n'est pas considérée comme critique. Dans le cas contraire elle apparait comme critique, en rouge dans le diagramme de GANTT. En fixant minslakrate à 100 (100%) toutes les tâches deviennent critiques, en fixant minslackrate à 0 la notion de criticité disparait.

2.3 Corps du projet, paramètres

Avant d'attaquer le contenu, à savoir les successions de tâches, il va être nécessaire de préciser les paramètres du projet, notamment le calendrier (vacaces, jours fériés), les horaires de travail, mais aussi les comptes de recettes et frais, la ressource humaine etc.

2.3.1 now

Utilisé au coup par coup now permet de fixer la date du jour afin de simuler l'état du projet à une date différente de la date de l'horloge du PC. Syntaxe :

now 2007-10-15

2.3.2 extend

Applicable entre autres aux tâches.

2.3.3 dailyworkinghours

Ce paramètre est important pour effectuer la conversion d'heures en jours de travail et réciproquement. Syntaxe :

dailyworkinghours 7.5

Là aussi, pas question que la demi heure soit acceptée si le paramètre timingresolution n'a pas été fixé à 30 min. Voir plus haut.

2.3.4 workinghours

Par défaut TJ suppose que nous travaillons 3 heures le matin et 5 l'après midi. On peut préciser grâce à workinghours un horaire plus conforme à nos 35 heures du genre :

workinghours mon-thu 8 :00-12 :00, 13 :00-16 :30
workinghours fri 8 :00-12 :00, 13 :00-15 :00
workinghours sat-sun off

L'association de ces trois lignes défini un horaire assez standard.

2.3.5 flags

Flag signifie drapeau. Cette primitive défini un marqueur qui permettra, en cours de traitement du fichier, d'isoler, séparer, ignorer certaines parties qui seraient marquées de ce drapeau. Nous verrons en détail l'utilité de ce paramètre lors de la construction des rapports. Retenez dans un premier temps que la notion de flags apparaît à trois endroits, leur déclaration, le marquage de tâches et l'exploitation sélective des résultats.

2.3.6 account

Le suivi budgétaire est sans doute l'un des aspects qui a le plus attiré mon attention dans l'étude de TaskJuggler.

Initialisation des comptes :

Trois mots clefs sont suffisants pour créer toute votre structure de comptes, account, cost et revenue. Account désigne un compte, cost en fait un compte de frais ou de dépenses, revenue un compte de recettes.

Vous pouvez souhaiter distinguer dans les dépenses celles qui concernent de factures payées à tiers et celles qui concernent des prestations internes. C'est très simple avec la syntaxe suivante :

account dep 'Dépenses non affectées' cost
account ct 'Commandes à tiers' cost{
account fp 'Frais de personnel' cost
account rec 'Recettes' revenue
}

Compte par défaut :

Une ligne **account dep** placée dans l'entête de la tâche principale permet d'affecter par défaut tous les frais dont vous auriez oublié de définir l'imputation à un compte spécifique. Ce compte apparaîtra dans le rapport de situation budgétaire. Tant que son montant ne sera pas nul vous saurez qu'il y a un problème dans l'affectation de vos recettes et dépenses.

Affectation aux comptes :

Pour les frais de personnel qui sont des frais répartis sur toute la durée de la tâche il suffit ensuite d'affecter méticuleusement chacune des recettes ou dépenses aux différents comptes. Cette affectation se fait au niveau des tâches. Personnellement je le fais sitôt l'affection du personnel faite. Exemple :

allocate jcs
account fp

Affectation dépenses ponctuelles :

Les réglements des clients ou les factures des fournisseurs sont des évênements ponctuels le plus souvent liés à des étapes du projet, début du projet, remise

des plans, réception provisoire, réception finale etc.

Les mots clefs **startcredit** et **endcredit** permettent de positionner ces évènements sur une tâche de durée non nulle. Sur une borne ces deux commandes sont équivalentes.

Exemple :

task e 'Tâche E'{
startcredit 12000
account rec
depends !d
milestone
}

Autres possibilités :

L'emploi du mot clef **credit** permet d'affecter directement des sommes aux différents comptes à une date donnée sans être contraint de le faire via une commande **startcredit** ou **endcredit** dans une tâche ou une borne. Avantage, on regroupe toutes les recettes dans la définition du compte recettes, inconvénient si une tâche conditionnant le paiement d'une facture glisse vous ne penserez pas obligatoirement à décaler l'échéance du règlement.

Autre inconvénient : Les dépenses et recettes déclarées lors de la mise en place des comptes n'apparaissent pas sur les éventuelles colonnes du diagramme de GANTT alors qu'elles sont correctement prises en compte dans le bilan d'affaire.

Pour affecter une recette (par exemple) au compte recette :

account rec 'Recettes' revenue{
credit 2007-06-08 'Réglement avance sur travaux' 5000.0
}

Vous avez également la possibilité de créer toute une arborescence de comptes. Exemple :

account dep 'Dépenses' cost
account ct 'Commandes à tiers' cost{
account un 'Sté Untel'
account so 'Boucherie Sansos'
account xx 'Autre fournisseur'
}
account fp 'Frais de personnel' cost
account rec 'Recettes' revenue

Les comptes un, so et xx sont des sous-comptes de ct. Ils apparaissent en particulier dans htlmaccountreport sauf si rollupaccount a été validé (rollupaccount 1).

2.4 Corps du projet, les tâches

2.5 Exploitation des données

3 Personnalisation de TaskJuggler

3.1 Francisation modèles

Les fichiers modèle se trouvent dans :
/usr/share/apps/taskjuggler/templates/en_US/

Il suffit de copier les fichiers modèles en leur donnant de nouveaux noms et vous obtenez une bonne base de travail pour effectuer la francisation des modèles. J'avoue que la francisation n'est pas parfaite. Les rapports html en particulier montrent des résultats très différenciés. Le tableau de dépenses est recettes est assez correct, avec toutefois un 'Revenues' assez maladroit. Le rapport de situation quand à lui est inchangé par rapport à la version d'origine.

Exemple de fichier modèle `/home/jml/Projet_simple.txt`