Gestion d'agenda pour un groupe d'utilisateurs

François Sillion

Francois.Sillion@imag.fr

1 Introduction

On souhaite proposer à un groupe d'utilisateurs un logiciel de gestion d'agenda et de rendez-vous, permettant une gestion centralisée des horaires. Chaque utilisateur pourra ainsi planifier son temps, mais aussi prévoir des réunions avec d'autres membres du groupe, en fonction de leurs disponibilités.

Les fonctionalités proposées par l'agenda peuvent être par exemple

- extraire l'agenda du participant X pour la journée (semaine, mois..) Y
- planifier une réunion selon une liste de critères
  - avant (après) la date D (ou le plus vite possible, ou un lundi...)
  - avec les participants X,Y,Z...
  - ...

On définira un modèle simple d'agenda. Par exemple si l'on considère la donnée de base comme une *réunion*, consistant en un créneau horaire et une liste de participants, une version simpliste de l'agenda consisterait à conserver une liste de réunions prévues, et à parcourir cette liste à chaque fois qu'une requête est faite. On souhaite utiliser une structure plus efficace, et permettant des requêtes plus élaborées.

2 Structures de données

On vous propose de manipuler une structure générale (appelons-là *planning*), représentant un intervalle de temps donné comme un ensemble d'intervalles *libres* ou *occupés*. Il est alors possible de représenter l'emploi du temps de chaque participant par un tel planning. Pour éviter des recherches linéaires dans une liste et accélérer les traitements, on préferera une représentation hiérarchique des plannings, sous forme d'arbres. Les feuilles de l'arbre sont alors des blocs (libres ou occupés), et les noeuds intermédiaires représentent des intervalles de temps de durée variable (Figure1).

L'intérêt de la hiérarchie est multiple: on peut faire remonter des informations dans les noeuds élevés de l'arbre (représentant par exemple des journées ou des semaines), comme la taille du plus grand bloc libre... on peut aussi traiter tout sous-arbre comme un planning à part entière, ce qui permet l'écriture de procédures générales, travaillant sur un planning quelconque et fonctionnant de manière récursive. On souhaitera certaineement contraindre le découpage temporel de l'arbre pour que les semaines et les journées soient des noeuds.

Les algorithmes mis en oeuvre seront alors du type intersection de plannings (pour trouver un créneau commun) ou union de plannings (pour s'assurer qu'il y a toujours au moins une personne de libre pour répondre au téléphone...

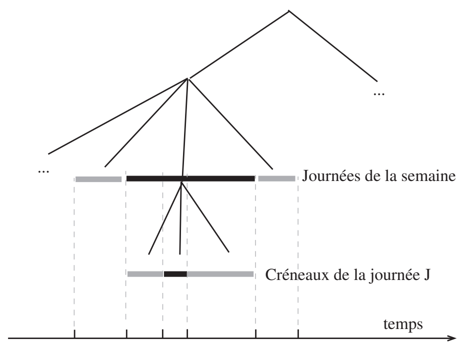

Figure 1: Arbre de représentation hiérarchique des intervalles de temps. Par souci de simplicité, on pourra limiter le nombre de fils d'un noeud à trois.

3 Travail demandé

L'agenda réalisé devra permettre de travailler avec plusieurs participants (au moins 3), sur une durée significative (au moins un mois). Il devra permettre de répondre aux requêtes simples énoncées en introduction, selon une série de contraintes ou de souhaits exprimés (et dont la liste exacte est laissée à l'appréciation de chacun).

On ne demande pas une interface complexe ! il faut d'une part pouvoir entrer rapidement une demande de réunion avec les contraintes/souhaits associés, d'autre part pouvoir extraire des plannings individuels ou collectifs (sous forme de texte). On souhaitera probablement pouvoir sauver dans un fichier et re-charger un planning, pour éviter les saisies fastidieuses.

4 Extensions possibles

4.1 re-planification dynamique d'une réunion

On pourra chercher à conserver en mémoire les contraintes non satisfaites lors de la planification d'une réunion (par exemple, réunion trop lointaine parce qu'un participant n'est pas disponible avant la semaine prochaine), de façon à réagir lorsqu'une réunion est annulée, en essayant de reprogrammer celle qui n'était pas satisfaisante.

Pour cela, il faut d'une part permettre l'annulation de réunions, d'autre part conserver la mémoire des contraintes liées à certaines réunions.