Projet MIM2

ENS Lyon - septembre 2001 - décembre 2001

Vivian Brégier & Laure Danthony

Réalisation d'un logiciel d'emploi du temps

PLAN :

- Présentation du projet et cahier des charges
- Partagons-nous le travail !
- L'interface graphique
- Le calcul d'une solution
- Conclusion.

Présentation du projet et cahier des charges

Le problème

PROJET : réaliser un logiciel Windows facile d'utilisation qui étant données des contraintes de type horaires, disponibilités des professeurs et non-collision des cours d'un élève donné, donne une solution possible au problème.

Les contraintes

Elles sont de trois types :

- Un certain nombre de créneaux disponibles : 1, 2, ...
- Disponibilité des professeurs : `yrobert` peut être présent aux créneaux 4 et 7, ...
- Pas de collisions entre les cours des élèves : si `vbregier` va aux modules "algo_cours" et "compil_cours", alors ces deux modules ne peuvent avoir lieu en même temps [a].

[a] Quoique :-)

Partageons-nous le travail !

Trois parties "indépendantes" ...

- La partie "entrée des données" sous forme intuitive qui se fera à l'aide d'une application Java sous Windows. [VIVIAN]
- La partie "calcul d'une solution". [LAURE]
- La partie "affichage d'une solution". [VIVIAN]

... qui communiquent (1)

- L'interface-utilisateur lance à l'aide d'un bouton "calcul" le programme CAML de calcul d'une solution.
- Ce programme récupère un fichier `toto.data` sous la syntaxe suivante :

```
%PROFS%
     yrobert:algo_cours,compil_cours;1,2,3,4;
     fdedinec:compil_td;2,7,9;
     ...
%MODULES%
     compil_td:fdedinec;
     ...
%ELEVES%
    ldanthon:algo_cours,algo_td,compil_cours,compil_td;
    vbregier:projet,algo_bio_cours;
    ...
```

... qui communiquent (2)

- Certaines informations sont redondantes
- Ordre de grandeur : 14 modules à 15 élèves, 7 profs et 10 plages horaires possibles par prof, 10 modules par élèves et 30 élèves : au plus 5 ko.
- Après calcul, le programme renvoie une solution sous la forme :

```
1:algo_cours
4:compil_cours
...
```

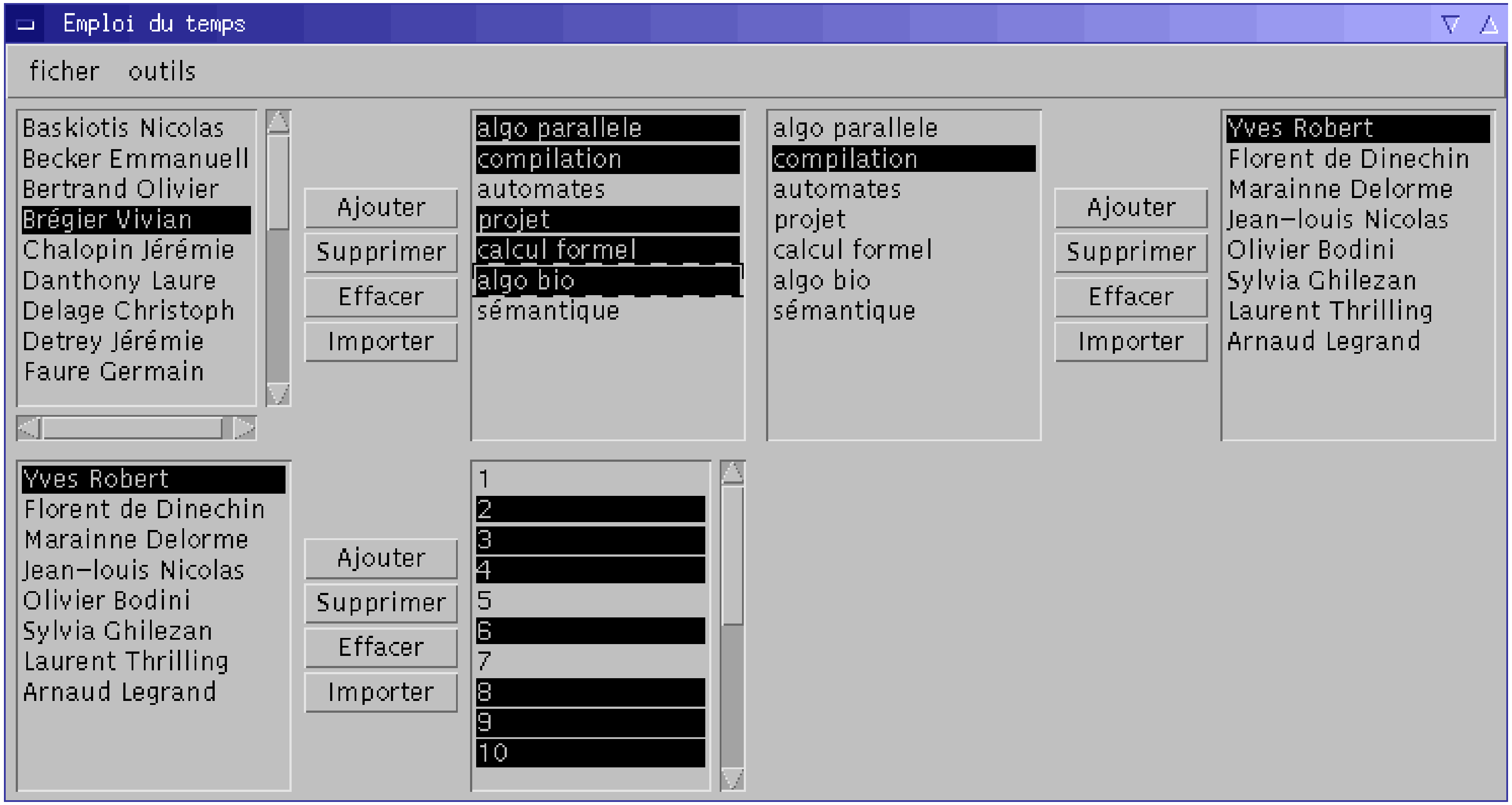
. . .dans une jolie boîte !

Partie Graphique

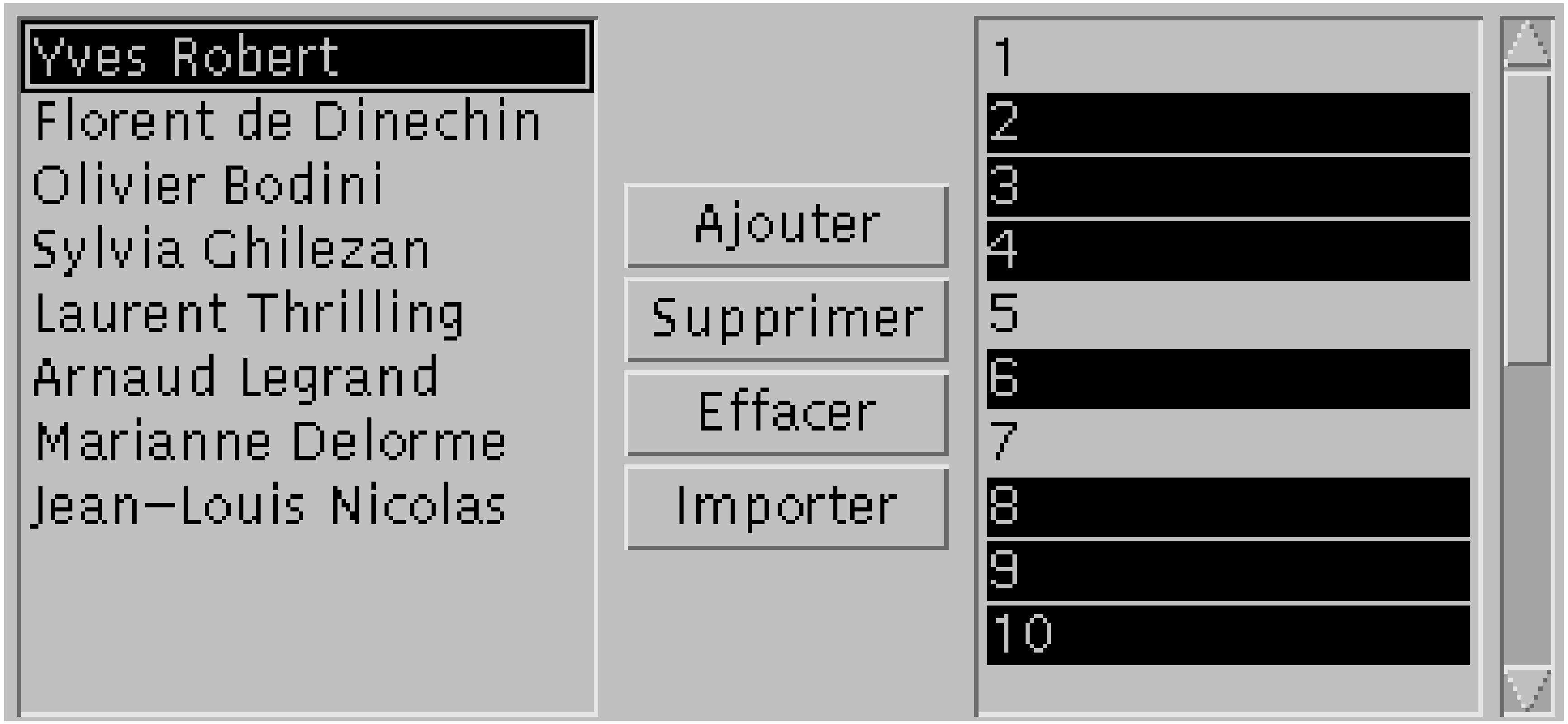
Contraintes intuitives

- Tout doit se faire à la souris, c'est-à-dire sans ligne de code.
- L'utilisation doit être ergonomique.
- Les changements doivent être dynamiques.
- On doit pouvoir sauver un ensemble de contraintes.
- Un bouton **calculer** effectuera le calcul puis l'affichera dans une jolie fenêtre.

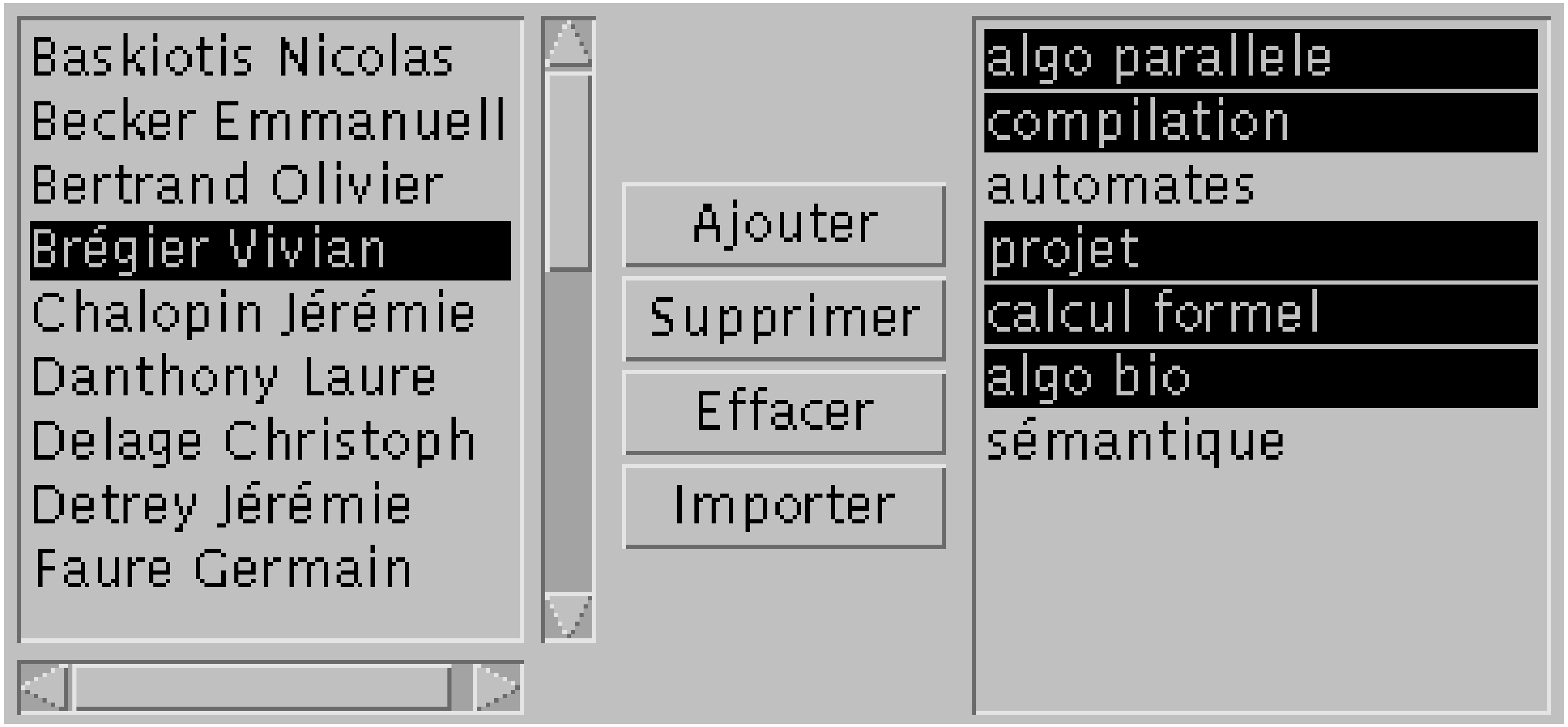
Quelques copies d'écran - Boîte entière



Quelques copies d'écran - Zoom (I)



Quelques copies d'écran - Zoom (II)



Quelques copies d'écran - Zoom (III)

Partie Calcul

Formalisation du problème

- Une solution est une affectation créneau/module qui vérifie toutes les contraintes.
- Les contraintes peuvent être mises sous forme de graphes.
- En plus : on doit pouvoir fournir une autre solution sans tout recalculer.

Implémentation (1)

- Codage des contraintes : graphe de dépendances entre modules...
- Une structure de données **situation** qui contient une liste des modules déjà affectés, et pour les autres modules les horaires possibles restant à vérifier.
- Algo : On choisit M qu'on désire placer (le plus restrictif possible). On essaie de lui allouer un créneau C (appel récursif) :

  ```
  OK      ----> alloc ok
  Echec   ----> choix d'un autre C
  ```

  Si aucun créneau n'est possible, on renvoie `Echec`. Sinon on ajoute $(creneau, module)$ à la liste des modules déjà affectés.

Implémentation (2)

- Partie Compilation réalisée dans `recup_donnees.ml`.
- Partie Calcul réalisée dans `calcul.ml`.
- Un seul exécutable dans `main.ml`.

Résultats (1)

- Test sur un fichier codant les contraintes de MIM2 de ce semestre.
- On trouve une solution en $2\mu s$ sur un Céléron 366Mhz avec 320Mo de RAM.

Résultats (2)

- Vous y avez cru ? La partie calcul n'est pas encore effectuée.
- Le fichier de tests, lui, est prêt ainsi que la partie de récupération des données.

Conclusion

Difficultés

- Problème NP-complet : plus de 40000 possibilités à tester.
- Interfaçage CAML/JAVA : communication uniquement avec des fichiers et l'exécutable `calcul fich_in fich_out`.

Ya plus ka...

- Fignoler l'interface graphique : gérer de manière intuitive les plages horaires (cases à cliquer), changer les couleurs, gérer les placement des fenêtres, mettre des onglets, améliorer la sauvegarde de contraintes, ... et aussi la partie retour du résultat sous forme d'un tableau.
- Coder la partie calcul et peut-être améliorer l'algorithme.
- Gestion des erreurs dans le fichier de communication[a]
- Réaliser un exécutable WIN !
- Evaluer les performances avec la principale intéressée.

[a] il y a des redondances !