Département Informatique
Année 2001/2002

Rapport de Synthèse

Grapheur pour une application en planification

Sylvain Leblond

Pacte Novation
2, rue du Docteur Lombard
92 441 Issy-les-Moulineaux Cedex
☎ 01.45.29.06.06

Enseignant responsable :
M^r Jean-Marie PINON

Tuteur de stage :
M^r Philippe Morignot

Résumé

Le but de ce projet de fin d'étude est la spécification,conception ainsi que la réalisation d'une interface Homme-Machine permettant de représenter simplement des exemples de plans d'actions ainsi que l'ensemble des informations permettant de comprendre ceux-ci. Ce projet fait suite à deux projets de fin d'étude résultants en la création de FastPlan 1 et 2, deux applications de résolution de plans d'action qui ne disposent pas d'IHM graphique, on s'est donc proposé de les relier à la nouvelle interface. La réalisation de l'interface a été faite sous visual C++ en utilisant les bibliothèques Ilog Views, et plus précisément l'objet grapheur.

Cette IHM a pour but de démontrer l'utilité des plans d'action aux clients de Pacte Novation qui ne sont pas censés connaître ces méthodes.

Mots clefs

IHM, C++, méthode de conception OO, Librairie, Ilog Views, Ilog CPLEX, Ilog Solver, PDDL, planification, IA, Plans d'actions, Visual studio

Abstract

The purpose of this project of end of study is the specification, the design as well as the implementation of a graphical user interface allowing to simply represent examples of action plans as well as all the information allowing to understand them. This project follows upon two resultant projects of end of study in the creation of FastPlan 1 and 2, two applications of resolution of action plans which do not iclude GUI, one thus suggested connecting them with the new interface. The implementation of the interface was done under Visual C++ by using libraries Ilog Views, and more exactly object graphics application package.

This GUI aims at demonstrating the utility of action plans to the customers of Pacte Novation who are not supposed to know these methods.

Key words

GUI, C++, Libraries, Ilog Views, Ilog CPLEX, Ilog Solver, PDDL, scheduling, AI, Action plans, Visual studio

Introduction

Le présent document constitue une synthèse du travail réalisé au sein de la société Pacte Novation au cours de mon Projet de Fin d'Etudes (PFE) 2001-2002.

L'intelligence artificielle est aujourd'hui de plus en plus utilisée en informatique pour répondre à la demande d'industriels cherchant à utiliser des systèmes autonomes ou des systèmes réalisant des tâches complexes.

Le domaine de la planification commence à être utilisé pour répondre aux exigences des industriels. Ce domaine permet d'organiser un grand nombre de tâches dans le temps afin d'arriver à un but souhaité et ainsi répondre aux besoins d'autonomie ou d'aide à la décision. Les applications sont diverses, tels que le déploiement de troupes militaires sur une zone donnée ou encore le choix d'alternatives pour une sonde spatiale.

Afin de répondre aux besoins des clients, Pacte Novation a acquis le savoir-faire nécessaire à la réalisation de logiciels de planification en développant, lors de deux projets de fin d'étude, deux logiciels de génération de plans d'action (sous-ensemble de la planification) répondant à un critère de rapidité de résolution. Bien qu'efficaces, ces logiciels ne disposent pas d'interface Homme-Machine satisfaisante.

Le but de ma mission a été de spécifier concevoir et réaliser cette interface en ayant la contrainte de créer une interface simple d'utilisation et facilement compréhensible par toute personne n'ayant que quelques connaissances sur le problème de la planification.

I Analyse de l'existant

Une des difficultés de ce projet était l'appropriation de l'existant. En effet, suite à deux projets de fin d'étude sur le sujet des plans d'action, l'existant a été fortement étoffé.

La première analyse a consisté à comprendre le fonctionnement des plans d'action et de son formalisme, la seconde a consistée en une analyse du code et des structures de données des deux applications créées au préalable.

I.1 Les plans d'action

Suivant le formalisme STRIPS (STanford Research Problem Solver) qui est une des plus populaire notation pour la représentation des problèmes de planification dans le domaine de l'Intelligence Artificielle [Aug01], un plan d'action représente un enchaînement d'actions à réaliser pour qu'un système donné passe d'un état initial à un état final différent de l'état initial.

De ce fait un plan d'action contient une succession d'actions et d'états déterminant l'évolution du système.

1.) Etats

Un état, comme son nom l'indique décrit l'état du système à un moment donné. Pour ce faire, il est composé d'un ensemble de propositions permettant de décrire de manière unitaire le système.

Un exemple d'état est donné sur la Figure 1. Celui-ci provient du monde des cubes [Cou00] et représente l'état d'un système composé de 4 cubes (A,B,C,D) et d'une table. Comme on peut le voir, cet état est composé de 4 propositions.

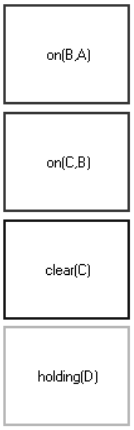

Figure 1 - Représentation de l'état du système à un instant t (tiré du monde des cubes)

2.) Actions

Une action est une procédure permettant de faire évoluer le système d'un état donné à un nouvel état pouvant être égale au premier (Actions noop). Plus précisément, une action transforme un ensemble de propositions bien définies en un nouvel ensemble de propositions en en créant ou en en niant certaines.

Un exemple de représentation d'action est disponible sur la Figure 2. L'action est représentée par une ellipse, les carrés verts à gauche de l'action sont les propositions nécessaires à la réalisation de l'action (propositions en pré condition) et les carrés à droite de l'action représentent les propositions

créées (en vert) ou niées (en rouge) par l'action (effets de l'action).

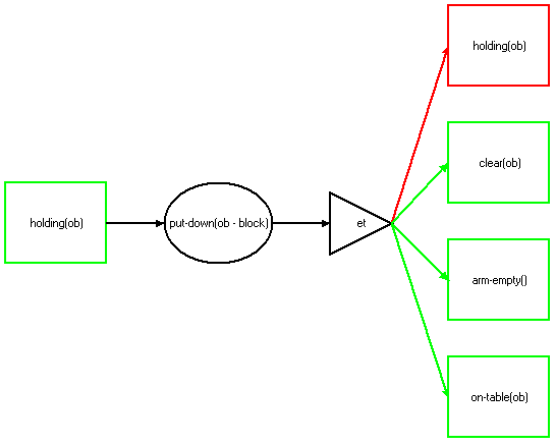

Figure 2 - Représentation d'une action (formalisme STRIPS)

I.2 Le formalisme utilisé (PDDL)

Le formalisme utilisé pour décrire un système est le formalisme PDDL (Planning Domain Definition Language).

Ce formalisme est l'un des plus aboutit pour décrire un système donné. Il a été créé afin d'uniformiser la création de logiciels de planification pour le concours AIPS (Artificial Intelligence Planning System).

Afin de décrire un système, le formalisme PDDL impose la création de deux fichiers textes : un fichier domaine et un fichier problème.

1.) Fichier Domaine

Le domaine défini les différents prédicats définissants le système, ainsi que les actions réalisables en donnant leur pré conditions et leurs effets.

Exemples :

Définitions des prédicats :

```
(:predicates (on ?x - block ?y - block)
             (on-table ?x - block)
             (clear ?x - block)
             (arm-empty)
             (holding ?x - block)
             )
```

Définitions d'une action :

```
(:action put-down
            :parameters (?ob - block)
            :precondition (holding ?ob)
            :effect
            (and (not (holding ?ob))
                 (clear ?ob)
                 (arm-empty)
                 (on-table ?ob)))
```

De plus, le domaine défini un certain nombre d'options permettant par exemple d'utiliser des types dans le problème, c'est à dire d'utiliser la possibilité d'affecter un type pour chaque objet. Dans le monde des cubes un seul type d'objet est nécessaire, le type bloc, de ce fait le fichier domaine devra donc déclarer l'option "typé", « typing » et déclarer le type bloc :

```
(:requirements :strips :typing)
(:types block)
```

2.) Problème

Le problème est orienté vers la résolution du plan d'action, en effet il définit les différents objets utilisables et leur type si le domaine est typé, ainsi que l'état initial et final auquel on souhaite arriver. Les deux états sont définis par un ensemble de propositions :

Description des objets :

```
(:objects D B A C - block)
```

Description de l'état initial :

```
(:init
(clear C)
(clear A)
(clear B)
(clear D)
(on-table C)
(on-table A)
(on-table B)
(on-table D)
(arm-empty)
)
```

Description de l'état final :

```
(:goal (and
(on D C)
(on C B)
(on B A)
)
```

3.) la librairie PaserPDDL

Afin d'utiliser les fichiers au format PDDL dans les applications, Pacte Novation a développé lors d'un projet de fin d'études une librairie permettant de stocker les informations

provenant des fichiers domaine et problème dans différentes classes en proposant des méthodes de gestion de ces données [Aug01].

I.3 Les logiciels

Suite à deux projets de fin d'études deux logiciels de résolution de plans d'action ont étés réalisés : FastPlan 1 et FastPlan 2. Ces logiciels utilisent des méthodes de résolution différentes, utilisation de GRAPHPLAN (méthode de programmation par contrainte) pour FastPlan 1 et de SatPlan et StateChange (méthodes de programmation linéaire) pour FastPlan 2.

1.) FastPlan 1

FastPlan 1 utilise la méthode GraphPlan [Cou00] pour résoudre les problèmes des plans d'action.

GraphPlan est une méthode que l'on peut décomposer en deux phases :

1.) Expansion du graphe

A partir de l'état initial, cette phase consiste à utiliser toutes les actions réalisables en parallèle à partir de l'état précédent jusqu'à ce que l'état final soit contenu dans le résultat.

2.) Extraction de la solution

Une fois les actions « déroulées », la recherche de solution se fait en chaînage arrière, c'est à dire que l'on va rechercher s'il existe bien une solution permettant d'arriver à l'état final en respectant le fait que deux propositions, ou que deux actions puissent êtres mutex (non réalisables dans le même pas de temps).

Le principal atout de GraphPlan est que cette méthode peut être reformulée en terme de programmation par contrainte.

Pour ce faire, on choisit de définir un domaine pour chaque variable qui seront dans notre cas les propositions et le domaine sera l'ensemble des actions qui ont cette proposition comme effet.

Par exemple, si les actions a1 et a2 ont pour effet la proposition P2, le domaine de la variable P2 sera {a1,a2}. Les contraintes seront introduites par la notion de mutex, si on ajoute la proposition P1 introduite par l'action a3, et que a3 est mutex de a1, la contrainte sera : $P1 = a3 \Rightarrow P2 \neq a1$.

Cette application a été réalisée en C++ en utilisant la bibliothèque Ilog Solver, qui est une bibliothèque servant à résoudre les problèmes de programmation par contrainte.

2.) FastPlan 2

FastPlan 2 utilise la programmation linéaire pour résoudre les problèmes de plans d'actions suivant deux méthodes SatPlan [Aug01] et StateChange [Aug01].

1.) SatPlan

Cette méthode consiste à donner une valeur booléenne à chaque proposition et à traduire les actions par des propositions logiques représentants les pré conditions et les effets [Aug01].

De plus cette méthode permet de paralléliser les actions (plusieurs actions peuvent êtres exécutés en un même pas de temps) ce qui implique une définition des actions qui sont en conflit.

Pour trouver un plan de durée minimale, on choisit une durée initiale que l'on incrémente tant qu'aucune solution n'est trouvée. La solution est trouvée lorsque toutes les propositions de l'état final sont à vrai.

Afin de programmer cette méthode, l'ensemble du problème de planification est formulé en nombre entier, une variable représente une proposition et a pour valeur 1 (proposition active) ou 0 (proposition niée), les actions sont traduites par une équation de même que les actions mutex. La résolution se fait ensuite à l'aide de la librairie Ilog CPLEX.

2.) StateChange

StateChange est une méthode permettant de formuler un problème de planification en nombre entier plus efficacement que la méthode précédente. Elle permet d'obtenir des résultats plus rapides [Aug01].

Cette méthode consiste à choisir des variables qui marquent le fait d'un changement d'état pour une proposition. Par exemple, on choisira la variable x^{add} qui passera à 1 si la proposition est ajoutée par une action. L'ensemble des actions et des actions mutex sont mis en équation suivant ce principe. La résolution est aussi faite à l'aide de la librairie Ilog CPLEX.

II Spécification et conception de l'IHM

II.1 Contraintes

Les principales contraintes de l'IHM sont l'intuitivité et l'évolutivité. En effet, Le but de l'interface est de présenter le problème des plans d'action de manière simple afin qu'une personne ne connaissant pas ces méthodes puisse en comprendre le fonctionnement.

De plus, l'IHM devant être reliée à deux applications pouvant évolués dans l'avenir, elle se doit d'être évolutive afin de permettre des modifications simples et rapides dans le cas d'une mise à jour d'un des deux logiciels.

Afin de spécifier et concevoir cette IHM la méthode de conception vu en cours de 4IF a été utilisée [Pin00].

II.2 Domaines Fonctionnels

Afin de concevoir cette IHM, trois domaines fonctionnels ont étés mis en avant :

1.) Gestion de l'affichage

Ce domaine fonctionnel est le plus important, il regroupe l'ensemble des fonctions de visualisation et contient de ce fait les fonctions d'affichage des plans d'actions ainsi que de toutes les informations en relation avec ces derniers.

2.) Gestion des préférences

L'IHM étant reliée à deux applications utilisant eux même des produits Ilog (Ilog Solver et Ilog Cplex), un grand nombre de préférences sont à prendre en compte tout d'abord pour les produits Ilog, puis pour le choix de la méthode de résolution à utiliser et de ce fait le programme de résolution à utiliser.

3.) Gestion des fichiers PDDL

Le dernier domaine fonctionnel concerne les fichiers d'entrée au format PDDL. Ces fichiers d'entrées devront pouvoir être chargés, créés et modifiés.

II.3 Utilisateurs

Après une analyse des utilisateurs de l'application, nous avons retenu deux profils d'utilisateurs : l'ingénieur et le commercial interne à l'entreprise. Le graphe d'héritage des profils utilisateur est disponible en Figure 3.

L'ingénieur aura la possibilité de créer de nouveaux fichiers PDDL à l'aide de l'interface ou d'en éditer ainsi que de gérer les options de l'application.

Le commercial aura pour but de démontrer l'utilité des méthodes de planification à un client, il utilisera donc surtout les fonctions de visualisation. Ceci est détaillé dans la partie suivante.

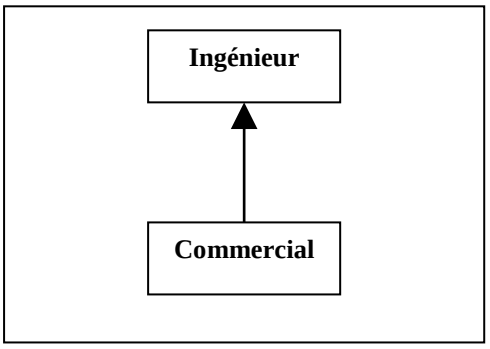

Figure 3 - Graphe d'héritage des profils utilisateurs

II.4 Description des tâches utilisateur

1.) Ingénieur

- Définir les données d'entrées (chargement/création/édition)
- Paramétrer l'application
- Visualiser les données du problème et du domaine
- Visualiser la solution s'il y en a une

2.) Commercial

- Charger les données d'entrée
- Choisir la méthode de résolution
- Visualiser les données du problème et du domaine
- Visualiser la solution s'il y en a une

II.5 Métaphore utilisée

Les utilisateurs de FastPlan connaissent à priori le fonctionnement des graphes, en effet les ingénieurs et les commerciaux en utilisent souvent sous différentes formes (Gantt, Réseaux PERT, etc.). De ce fait, on utilisera la métaphore fonctionnelle de plan d'action que l'on représentera sous forme de graphe.

Pour les objets graphiques de type icônes on utilisera alors la métaphore du plan d'action appliqué au monde des cubes, ainsi un état par exemple sera représenté par une image montrant un ensemble de cubes empilés.

II.6 Description des fonctions de l'interface

Nous allons ici décrire les différentes fonctions de l'IHM déduite de l'analyse des tâches utilisateur, que nous organiserons par domaine fonctionnel.

1.) Gestion des fichiers PDDL

- Chargement/Création/Edition du domaine
- Chargement/Edition/Création du problème

2.) Gestion des préférences

- Choisir la méthode de résolution (GraphPlan, SatPlan, StateChange)
- Paramétrer les produits Ilog

3.) Gestion de l'affichage

- Afficher les propriétés d'un prédicat
- Afficher les propriétés d'une action
- Afficher le domaine
- Afficher le problème
- Afficher la solution
- Afficher les informations relatives à la résolution (temps de résolution et nombre d'étapes pour arriver à l'état final)

II.7 Diagramme d'enchaînement des fenêtres

Le diagramme d'enchaînement des fenêtres (Figure 4) montre que l'ensemble des fonctionnalités sont accessibles depuis la fenêtre principale, bien que le point d'entrée soit sur la fenêtre d'entrée (F/Entrées), fenêtre permettant de choisir le domaine et le problème ainsi que la méthode de résolution.

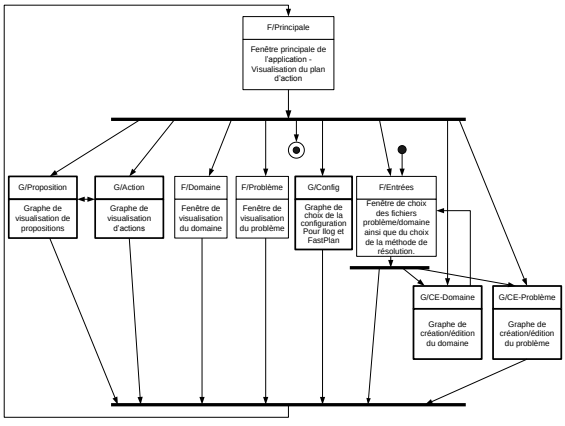

Figure 4 - Diagramme d'enchaînement des fenêtres

II.8 Fenêtre principale

1.) Représentation de la fenêtre principale

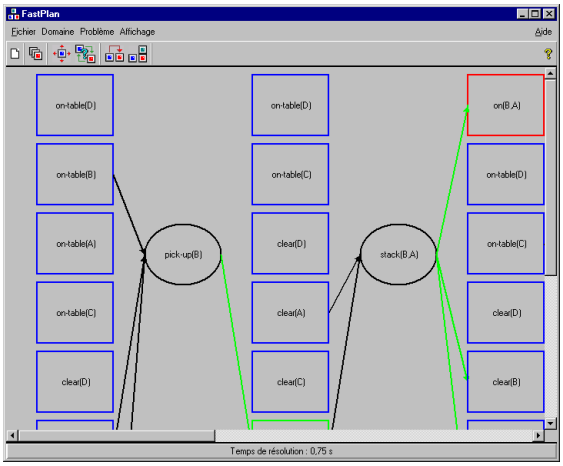

Figure 5 - Fenêtre principale de l'application

2.) Description succincte de cette fenêtre

La fenêtre principale de l'application est composée d'un menu, d'une barre de tâches, d'un espace de visualisation pour la solution ainsi que d'une barre de texte permettant d'afficher les informations sur la résolution.

Le menu, contrairement à la barre des tâches permet d'accéder à l'ensemble des fonctionnalités de l'application (Configuration, gestion de fichier PDDL et Visualisation). La barre des tâches quant à elle regroupe seulement les fonctions de visualisation afin de permettre un accès simple et rapide à ces fonctionnalités.

III Réalisation

III.1 Choix d'un outil de réalisation

Afin de réaliser l'interface il a fallu faire un choix entre deux design pattern de réalisation d'IHM : les MFC (Microsoft Foundation Class Libraries) [Kru01] et Ilog Views [Ilv99].

Les MFC ne disposant pas d'outils performants pour réaliser des graphes (très utilisés dans l'interface) nous avons opté pour Ilog Views qui dispose d'une suite d'outils très complets pour réaliser ces derniers [Ilv99]. De plus il permet de créer des interfaces fiables et portables sur différents systèmes d'exploitations [Ilv99].

III.2 Ilog Views

Comme nous l'avons indiqué ci-dessus, Ilog Views est un design pattern permettant de créer des interfaces graphiques.

L'utilisation de ces bibliothèques nécessite la création de classes utilisateurs qui héritent des classes de base d'ilog Views. En effet Ilog Views est essentiellement basé sur l'héritage. Par exemple, afin de créer une nouvelle fenêtre de type dialogue, il faudra créer une classe héritant de IlvDialog et redéfinir différentes méthodes virtuelles suivant l'utilisation.

Afin de faciliter la création d'interface, Ilog Views dispose d'un « studio » permettant de créer les interfaces graphiquement de façon intuitive.

III.3 Réalisation de l'API

La réalisation de l'application s'est faite en utilisant 5 classes d'interfaçage avec les programmes FastPlan ou le PaserPDDL. Chacune de ces classes représente un objet du plan d'action (une classe d'interfaçage pour la solution : CSolution, une autre pour le domaine : CDomaine ou encore pour les actions : CAction…).

Chacune de ces classes contient un certain nombre de méthodes permettant de gérer les informations de chaque objet et de les afficher plus facilement, permettant de ce fait d'alléger le code des fenêtres de l'application.

IV. Démarche de Gestion de Projet

IV.1 Phasage

Le projet c'est déroulé en trois phases :

- Etude de l'existant & prise d'informations sur le projet
- Spécification & Conception
- Réalisation & test

IV.2 Méthodes & outils utilisés

Pour la phase de Spécification et de Conception la méthode de conception d'interface Homme-Machine orientée objet vu en 4IF a été utilisé [Pin00].

L'application a été implémentée avec l'outil Ilog Views pour la partie interface graphique.

IV.3 Livrables

Les livrables pour ce projet sont au nombre de 8 :

- Dossier d'initialisation
- Etude de l'existant
- Dossier de spécification/conception de l'IHM
- Maquette intermédiaire
- Manuel d'utilisation
- Application finale
- Rapport de synthèse
- Rapport de PFE

Conclusions

Ce projet m'a permis de réaliser l'interface d'une application de l'étude des besoins jusqu'à la réalisation tout en réalisant les tâches de gestion de projet. Utilisant des méthodes d'intelligence artificielle, le projet a permis de créer une application très intéressante au niveau de ses fonctionnalités.

Mes connaissances de l'UML et du C++ m'ont permis d'appréhender très rapidement les applications existantes ainsi que leur modèle, et de m'approprié facilement un nouvel outil de réalisation d'IHM : Ilog Views.

L'application servira à convaincre les clients de l'utilité d'une telle technologie pour leurs projets futurs. Il n'est pas dénué de sens de penser que d'autres projets de planification pourrons bientôt voir le jour mais cette fois-ci appliqués à un domaine donné par le client.

Références bibliographiques

[Ilv99] Ilog
" Ilog Views 3.1 User's Manual ", Avril 1999

[Aug01] S. Augusto
"FastPlan2 Utilisation de la programmation linéaire en nombre entier dans le domaine de la planification », Rapport de stage de fin d'étude Pacte Novation 2001

[Cou00] J. Couvreur
« Génération de plans d'actions », Rapport de stage de fin d'étude Pacte Novation 2000

[Pin00] J.M Pinon
"Interfaces homme-machine », Cours 4IF Insa Lyon 2000

[Kru01] D.J Kruglinski, G. Sherherd, S. Wingo
« Atelier Microsoft Visual C++ 6.0 », Microsoft Press 2001