Architecture, trace, mise au point et visualisation dans CHIP

A. Aggoun, D. Bocquet, I. Gouachi, M. Inelhaj, R. Martin (COSYTEC)
P. Deransart (INRIA)

Projet: OADYMPPAC
Document: Délivrable
Date: Mai 2004
Version: 1.0
Référence: D3.4.2

Table des matières

1. INTRODUCTION 4

2. OBJECTIFS 6

3. INDUSTRIALISATION 6

4. BREVE INTRODUCTION A CHIP 7

4.1 DOMAINES DE CALCUL 7
4.2 LE CONCEPT DOMAINE 7
4.3 LES CONTRAINTES SYNTAXIQUES 7
4.4 L'ENUMERATION 7
4.5 LES CONTRAINTES GLOBALES 8

5. DEBOGUEUR PAR DES EXEMPLES 9

5.1 ECHEC 10
5.2 MAUVAISE SOLUTION 10
5.3 AUCUNE SOLUTION APRES UNE DUREE FIXEE 10
5.4 TEMPS D'EXECUTION IMPORTANT 10

6. ARCHITECTURE 10

6.1 ARCHITECTURE OADYMPPAC 10
6.2 ARCHITECTURE DISTRIBUEE DE DEBOGAGE 11
6.2.1 Visualiseur 12
6.2.2 Gestion des évènements 12
6.3 ENVIRONNEMENT DE DEBOGAGE DE CHIP 13

7. PRINCIPE DE FONCTIONNEMENT 14

8. SYSTEME PPC CHIP : SOURCE DE LA TRACE 15

8.1 CNI (CHIP NETWORK INTERFACE) 15
8.2 CTX (CHIP TRACE XML) 15
8.3 RECEPTION / CONTROLE 16
8.4 EMISSION TRACE BRUTE 16

9. SYSTEME DE DEBOGAGE ET DE VISUALISATION : DESTINATION DE LA TRACE 17

9.1 EMISSION / RECEPTION 17
9.2 PARSEUR 18
9.2.1 Paquet 18
9.2.2 Intégration de plusieurs DTD 18
9.2.3 Délégation 18
9.3 VISUALISEUR 18
9.4 COMPOSANTS EXTERNES A CHIP 19
9.5 CONCEPTION ORIENTEE OBJET 19
9.6 LE LANGAGE DE DEVELOPPEMENT 19
9.7 LE FRAMEWORK MVC 19
9.8 VISUALISATION DE L'ARBRE DE RECHERCHE 20
9.8.1 Principe de fonctionnement 20
9.8.2 Application du modèle MVC 21
9.8.3 Vues principales d'un arbre de recherche 21
9.8.4 Vue du domaine des variables 22
9.8.5 Vue combinée 24
9.9 VUES METIER 25
9.9.1 Démarche 26

Confidential © 2004 COSYTEC SA. All Rights Reserved.

9.9.2 Vues Métiers et les concepts des contraintes globales 27
9.9.3 Architecture du composant « vues métiers » 30
9.9.4 Objets sémantiques 31
9.9.5 ChipViewTable : vue table 32
9.9.6 ChipViewChart : vue courbe 32
9.9.7 ChipViewGantt : vue affectation 33
9.9.8 Vue séquencement 36
9.9.9 Vue 3D 37
9.9.10 Vues combinées 38

10. CONCLUSION 40

11. REFERENCES 41

Confidential © 2004 COSYTEC SA. All Rights Reserved.

1. Introduction

Ces dernières années, un nombre important d'applications dans le domaine de l'aide à la décision ont été développées avec les outils de programmation par contraintes [Simonis 1996]. Les problèmes traités sont de très complexes et le besoin d'outils et de méthodologie de mise au point, dédiés PPC, se font de plus en plus ressentir. Il existe très peu de travaux dans ce domaine. Les programmes PPC sont différents des programmes conventionnels, ce sont des programmes dirigés par les données contrairement aux autres dirigés par la logique du programme. En général, un programme PPC est structuré en trois parties : déclaration des variables (de décision) domaines, pose des contraintes et enfin l'énumération. La mise au point concerne les trois parties; cependant une attention particulière est accordée à la procédure d'énumération car dans la plupart des problèmes industriels l'espace de recherche est très important. Pour comprendre et analyser l'effet de la procédure de recherche sur l'espace de recherche, il existe un besoin réel d'outils de visualisation dédiés permettant différentes visualisations des domaines (le domaine d'une variable, l'effet de la propagation), des contraintes et de l'espace de recherche. En plus de la visualisation des domaines des variables de décision, nous avons abordé dans ce projet la visualisation des objets métiers. En production, une opération est définie par un ensemble d'attribut (début dans le temps, la durée de l'opération sur une machine donnée, la machine d'affectation). Tous ces attributs sont dans beaucoup d'applications des variables de décision. Le développeur de l'application, comme l'utilisateur final, a besoin d'analyser ces attributs ensemble et ce à tout nœud de l'arbre de recherche. Dans le cadre de ce projet, nous avons développé des moyens pour exprimer des vues métier et de les visualiser. Les composants développés vont au-delà de l'aspect débogage, mise au point de programme. Ces composants offrent des moyens innovants et efficace pour faire également de l'analyse de performance.

Dans le cadre du projet RNTL OADymPPaC, COSYTEC et ses partenaires contribuent au développement d'outils de mise au point et de visualisation dédiés PPC. Ce document présente l'environnement de mise au point et de visualisation pour un système PPC et plus particulièrement CHIP.

Ce document reprend et étend les idées développées dans le rapport [ABGIMAD 03] « Environnement de mise au point et de visualisation dans CHIP. OADymPPaC, D3.1.2.2, Avril 2003 ».

Cette extension consiste particulierement à l'ajout d'un module dédié aux objets métiers. Le concepteur ou l'utilisateur d'une application réelle manipule beaucoup plus des objets métiers composés d'une ou plusieurs variables de décision. Par exemple, dans des problèmes d'ordonnancement, l'objet utilisé est une tâche. Cette tâche comprend une date de début, une durée et une variable qui contient la machine sur laquelle la tâche s'exécute. La visualisation en temps réel de l'emplacement de ces objets (par exemple sur un gantt), suivant l'avancement de la résolution, permet facilement de comprendre le comportement du solveur. Il peut permettre, également de déceler ses faiblesses ou ses incohérences. Car une position d'une tâche dans un gantt est plus parlante que la lecture de l'état des domaines de chacune des variables qui compose cette tâche. Par exemple l'utilisateur peut bien se poser la question pourquoi une tâche s'exécute sur telle machine alors qu'il est plus optimal si elle s'exécute sur une autre.

Le module de visualisation des objets métiers rend l'environnement de débogage de CHIP, plus complet, accessible aux spécialistes et aux non-spécialistes de la PPC. Notons que la majorité des utilisateurs de CHIP ont peu de connaissances sur la PPC. Le module de visualisation leurs sera d'une utilité importante pour déboguer leur solveur ou améliorer ses performances.

COSYTEC a consenti des efforts importants pour la réalisation de ce module. Les principales tâches effectuées pour réaliser le prototype sont:

- Conception d'une nouvelle DTD vue métier pour l'envoie de la trace XML des objets métiers comme des objets graphiques, donc avec leurs différentes propriétés graphiques.
- Conception et réalisation d'un langage de description des objets métiers
- Réalisation d'un nouveau générateur de trace XML de ces objets métiers.
- Création des vues macro simple d'utilisation. Dont l'objectif est de visualiser les objets proches de la modélisation utilisant les contraintes globales:
    - ChipVisualizeGantt,

Confidential © 2004 COSYTEC SA. All Rights Reserved.

  - ChipVisualizeTable,
  - ChipVisualizeChart,
- Implémentation d'un nouvel analyseur/contrôleur de la trace XML de la nouvelle DTD
- Implémentation des nouvelles vue métiers:
  - la vue gantt,
  - la vue table,
  - la vue sequencement,
  - les vues 2D et 3D.

La description de ce nouveau module est donnée dans la section 9.9

Confidential © 2004 COSYTEC SA. All Rights Reserved.

2. Objectifs

L'objectif fixé par COSYTEC est la conception et le développement d'un prototype industriel pour la mise au point et la visualisation pour le solveur CHIP. Nous avons travaillé sur les différents composants de l'architecture de l'environnement de mise au point et de visualisation de CHIP. Cette architecture est détaillée dans le rapport D3.1.2.2 [ABGIMAD 03]. Globalement, les principaux composants sont le solveur, le traceur, le débogueur, le visualiseur et enfin un outil de communication entre le solveur et les autres modules. L'application (solveur + application de l'utilisateur) est un processus indépendant et le débogueur en est un autre. Les deux processus peuvent tourner sur la même machine ou sur des machines hétérogènes. Ce choix nous a amené à travailler sur un protocole spécialisé de communication. Nous avons mis au point un algorithme permettant d'assurer la communication. La trace est générée au fil de l'eau et est immédiatement traitée par les autres modules : filtrage, bufferisation, transport, analyse et enfin la visualisation. La verbosité de XML nous a conduit à mener des expériences sur le découpage de la trace pour éviter de générer des messages trop longs pouvant dégrader les performances des autres modules et principalement l'analyse et la visualisation. Pour des petits problèmes combinatoires, la taille de la trace peut atteindre plusieurs mega-octets ; voire des giga-octets pour des applications réelles. Au niveau du stockage de la trace, nous avons travaillé sur des structures efficaces et sur des algorithmes rapides d'accès et de stockage de l'information pour contribuer à la réalisation d'un environnement (framework) performant. Au niveau du solveur, nous avons apporté des modifications au cœur même du solveur CHIP pour implanter la génération de la trace. Les résultats de ces travaux ont contribué à l'enrichissement de la DTD OADymPPaC pour l'ensemble du consortium du projet. Nous avons mené également des investigations pour définir une DTD XML pour le besoin des vues métiers. Les résultats sont encourageants, un prototype bien avancé de la DTD est en cours de finalisation et de test. Cette DTD appelée DTD Vues Métiers est intégrée dans l'environnement de CHIP en complément de la DTD Trace issue du solveur. Nous avons travaillé également avec l'EMN (groupe contraintes) sur l'explication et sur la réparation.

L'environnement de mise au point et de visualisation de CHIP est entièrement écrit en Java. Il utilise des composants de W3C. Dès sa conception, nous avons travaillé sur les points suivants :

1. Généricité ;
2. Facilité d'utilisation.
3. Prise en compte des spécificités des moteurs de contraintes.
4. Prise en compte des besoins utilisateurs.
5. Utilisation des composants open-source.

3. Industrialisation

Les résultats d'OADymPPaC pour COSYTEC, c'est à dire l'environnement de mise au point et de visualisation CHIP est en cours de finalisation (version alpha) en vue de l'intégrer dans la nouvelle version industrielle CHIP 5.6 prévue pour fin juin 2004. Les objectifs principaux sont atteints.

En terme d'exploitation, l'environnement de mise au point CHIP devrait contribuer à l'amélioration du temps de développement des prototypes industriels de l'ordre de 20% et d'une application de l'ordre de 10%. Ces résultats sont importants pour renforcer la position de COSYTEC dans le développement des solutions sous contraintes en France, en Europe et au niveau international.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

4. Brève introduction à CHIP

Les concepts fondamentaux de la PPC sont : la forme déclarative et relationnelle des contraintes, le niveau d'abstraction des contraintes [Simonis 98a], le non-déterminisme et enfin les domaines de calcul. Les contraintes sont des relations entre variables. Les variables sont les liens entre les contraintes. Ces liens sont la base des méthodes de propagation de contraintes.

Le non-déterminisme introduit la programmation incrémentale. Les contraintes sont ajoutées une à une. A chaque ajout la contrainte est soit entièrement satisfaite, partiellement satisfaite ou insatisfaite. Ce dernier cas est intéressant ; en effet insatisfaction signifie que les conséquences de celle-ci sont en contradiction avec l'ensemble des contraintes déjà existantes dans le système. La contrainte est automatiquement retirée ainsi que ses conséquences sur les domaines des variables. Ce type de programmation, dite programmation incrémentale, a un prix : quelles sont les informations à mémoriser à chaque ajout de contrainte pour pouvoir à tout moment revenir à l'état antérieur ? En revanche, elle libère l'utilisateur de la gestion des ajouts dynamiques des contraintes et des points de choix pendant la phase d'énumération.

4.1 Domaines de calcul

Les domaines de calcul sont inspirés des applications d'aide à la décision. Par exemple, dans CHIP les domaines les plus utilisés sont les domaines finis et les rationnels. Il existe d'autres domaines de calcul comme l'algèbre de Boole ou les intervalles. Cet article n'aborde que les domaines finis. Ceux ci permettent de modéliser d'une façon plus concise les problèmes combinatoires discrets. Un programme CHIP comprend trois parties : la déclaration des variables domaines (de décision), la pose incrémentale des contraintes et enfin l'énumération.

4.2 Le concept domaine

Une variable domaine est une variable qui prend ses valeurs dans un ensemble fini d'entiers naturels. Ces variables sont directement utilisées par les contraintes comme les contraintes arithmétiques sur des termes linéaires ou des inégalités. L'exemple suivant illustre un cas de déclaration de variables domaines. La ligne 1 indique que T_{11} et T_{12} sont des variables domaines pouvant prendre des valeurs entre 0 et 100. La ligne 2 exprime une contrainte (arithmétique) de précédence.

$[T_{11}, T_{12}]$ :: 0..100, *% Ligne 1*

$T_{12} \# \geq T_{11} + 5$, *% Ligne 2*

résultat après la pose de la contrainte et propagation :

T_{12} in {5..100}, T_{11} in {0..95}

4.3 Les contraintes syntaxiques

Parmi les contraintes syntaxiques, on peut citer les contraintes arithmétiques linéaires *($\leq$, $\geq$, >, <, =)*, la non-égalité ($X \neq Y$, X et Y ne peuvent pas prendre la même valeur), la contrainte comme *alldifferent(L)*, L est une liste de variables domaines. Elle contraint ses variables à prendre des valeurs différentes.

4.4 L'énumération

L'énumération est une phase importante dans un programme PPC. Elle permet de rechercher une solution au problème posé. Ce qui consiste à trouver une valeur pour chaque variable domaine tout en maintenant toutes les contraintes satisfaites.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

Le processus d'énumération défini une stratégie sur l'ordre de choix des variables et des valeurs. CHIP intègre des primitives de choix de variable et de valeur et offre également des moyens à l'utilisateur de programmer facilement ses propres heuristiques.

Choix des variables : L'utilisateur doit trouver un compromis entre des objectifs assez souvent contradictoires, par exemple minimiser la durée du projet (quelle tâche choisir ?) tout en maintenant une certaine équité en ce qui concerne l'utilisation des ressources. Les heuristiques utilisées proviennent des expériences en Recherche Opérationnelle et en Intelligence Artificielle.

Choix des valeurs : Le choix des valeurs dépend des objectifs fixés par le besoin : terminer le plus tôt possible, terminer juste à temps. Par exemple la primitive non-déterministe *indomain*(V_i) permet d'énumérer les valeurs de la variable V_i. Au premier appel elle fixe la variable à la plus petite valeur dans son domaine. En cas de retour arrière, elle essaie la valeur qui suit dans le domaine, etc.

La propagation : c'est un mécanisme qui provient de l'Intelligence Artificielle et principalement des CSP (Contraint Satisfaction Problems). Ce mécanisme détermine la manière dont les contraintes sont réveillées. Les méthodes utilisées pour résoudre les contraintes sont généralement incomplètes. Quand la contrainte est posée, elle est soit insatisfaite, partiellement ou entièrement résolue. Quand elle est partiellement satisfaite, elle peut de nouveau être appelée par le système si une de ses variables a été modifiée. Les contraintes peuvent être sollicitées de nombreuses fois. Les contraintes du problème forment un réseau connecté par les variables. La propagation est sollicitée de manière dynamique lors de l'énumération.

4.5 Les contraintes globales

La version CHIP V5 repose sur la deuxième génération d'outils de Programmation Par Contraintes : les contraintes globales [Aggoun 93, Beldiceanu 94]. Elles ont largement contribué à la progression de la technologie PPC. Il s'agit de primitives de 'haut niveau' destinées à modéliser des problèmes complexes utilisant des algorithmes de résolution spécifiques. Elles regroupent un ensemble de pré-traitements et de propagations appropriés. Certaines contraintes très proches des besoins exprimés dans des applications réelles apparaissent comme étant des conditions 'globales' et sont difficilement exprimables de façon compacte et concise avec des contraintes syntaxiques. Les contraintes globales (*cumulative, diffn, cycle* et *sequence*) ont permis une avancée technologique pour plusieurs raisons :

- un niveau d'abstraction permettant de modéliser les problèmes d'une façon beaucoup plus concise et proche de la réalité,
- ces contraintes sont génériques, complémentaires et multi-usage
- les performances - temps de calcul et consommation mémoire - sont très nettement améliorées,

Certains types de problèmes sont aujourd'hui bien résolus [Beldiceanu 96].

La taille du code étant extrêmement réduite, la maintenance évolutive de l'application est grandement facilitée. En effet, une ligne d'appel à une contrainte globale remplace plusieurs centaines de lignes de modélisation avec des contraintes syntaxiques.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

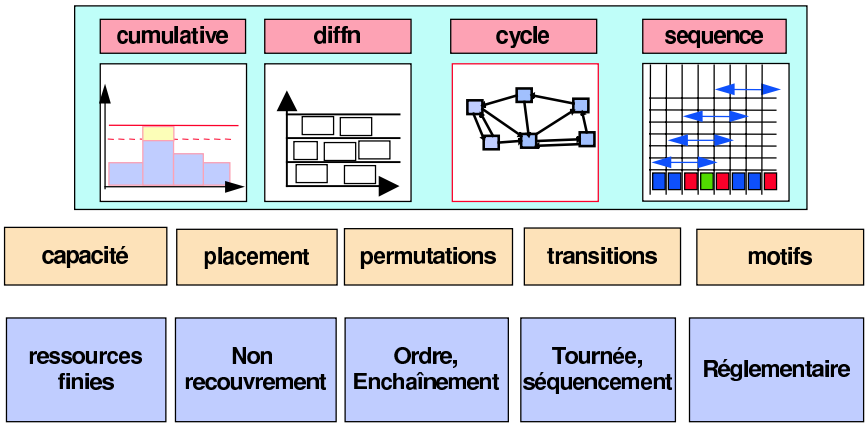

Figure 4-1 : Les contraintes globales et leurs domaines d'application

Exemple : La contrainte Cumulative

La contrainte *cumulative* [Aggoun 93] permet de résoudre des problèmes d'ordonnancement dans plusieurs secteurs d'activité tels que : l'informatique (affectation des tâches aux processeurs), la construction (gestion et suivi d'un projet), l'industrie de production (problèmes de gestion d'ateliers), l'administration (gestion des emplois du temps, rotation et affectation du personnel).

Elle exprime le fait qu'à tout instant, le total des ressources utilisées ne dépasse pas une certaine limite *Limite* fixée ou variable

Syntaxe : *cumulative(Débuts, Durées, Ressources, Limite, Fin)*

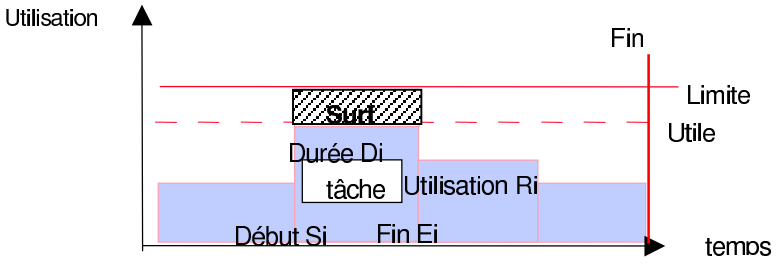

Figure 4-2 : illustration de la contrainte *cumulative*

Débuts est la liste des dates de début S_i des tâches, *Durées* la liste de leurs durées D_i, *Ressources* la liste des quantités de ressource R_i qu'elles utilisent, *Limite* le maximum de ressources disponibles, et *Fin* la date de fin de toutes les tâches.

Les autres contraintes comme *diffn* (non-recouvrement, affectation) ou *cycle* (problèmes de tournées) sont expliquées dans [Beldiceanu 1994].

5. Débogueur par des exemples

Dans les réalisations des applications réelles quelques situations sont souvent rencontrées. Cette section rappelle quelques-unes unes d'entre elles et montre comment un environnement de débogage peut être utile lors de la mise au point de ces situations.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

5.1 Echec

La phase d'énumération a échouée alors que le problème admet une solution. Le débogueur visualisera l'arbre de recherche construit et permettra l'accès à tout les nœuds. La consultation des nœuds qui ont conduit à cet échec va aider à le diagnostiquer pour comprendre son origine.

5.2 Mauvaise solution

Le programme trouve une solution mais l'utilisateur la trouve incorrecte, par exemple une variable prend une valeur qu'elle n'aurait pas du prendre. La visualisation de l'arbre de recherche et les informations sur les variables à chaque nœud vont aider l'utilisateur à mieux comprendre la source de l'erreur.

La cause peut être due soit à une modélisation incomplète d'un besoin du problème, soit à une erreur de programmation, par exemple, une contrainte qui porte sur une liste incomplète de variables.

5.3 Aucune solution après une durée fixée

La procédure de recherche explore un espace de recherche trop important sans résultat. L'existence d'une solution n'est pas garantie non plus.

Le débogueur donne le moyen à l'utilisateur d'arrêter la phase d'énumération puis de visualiser l'arbre partiel construit. La consultation de cet arbre permettra de comprendre la faiblesse de la modélisation. Cette faiblesse peut être due à une propagation très réduite des contraintes

5.4 Temps d'exécution important

Les performances du programme PPC sont, dans ce cas, restreintes. Le débogueur va visualiser un arbre important très dense.

Des changements dans le programme sont nécessaires, par exemple, par un ajout de contraintes redondantes ou une amélioration de la stratégie de recherche. Des changements dans les spécifications peuvent, également, conduire à de meilleures performances.

6. Architecture

Cette section décrit les évolutions de l'architecture de base d'un environnement de débogage. Tout d'abord, nous présentons l'architecture globale proposée dans le projet OADymPPaC. Ensuite, nous proposons l'architecture retenue pour CHIP.

6.1 Architecture OADymPPaC

L'architecture proposée s'articule autour de trois modules dont la conception est rendue aussi indépendante que possible : le traceur, le débogueur et le visualiseur.

Figure 6-1 : les modules d'un outil de débogage

- **Le traceur** : permet l'extraction de la trace du système PPC, son filtrage, sa transformation en format standard et enfin son émission.
- **Le débogueur** : analyse la trace afin d'en extraire les informations qui lui sont utiles et nécessaires. Il peut dialoguer avec le traceur en lui adressant des requêtes pour filtrer les informations de manière à améliorer son efficacité.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

- **Le visualiseur** : fournit les vues paramétrées à partir des informations transmises par l'outil de débogage.

Une description détaillée de cette architecture est donnée dans le rapport [DER02].

6.2 Architecture distribuée de débogage

COSYTEC a décidé dès le début du projet pour une architecture distribuée. L'application (solveur + génération de trace) et le débogueur sont deux processus indépendants tournant sur une même machine ou sur des machines séparées.

Dans une architecture à processus unique, les ressources d'une seule machine (éventuellement faible) sont utilisées. De plus, la défaillance de l'un des modules provoque l'arrêt complet du processus unique : il y a donc perte du débogueur. Ceci est un inconvénient majeur dans l'utilisation d'un système de mise au point.

L'architecture proposée par COSYTEC est donc une architecture distribuée illustrée par la figure ci-dessous.

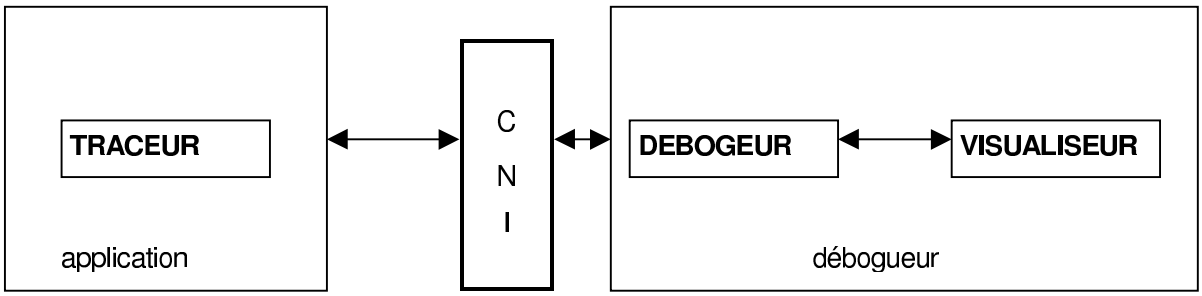

Figure 6-2 : les modules d'un outil de débogage revisité

L'application (solveur + application de l'utilisateur) est un processus indépendant et le débogueur en est un autre. Pour faire communiquer les deux processus, il est indispensable de développer un outil capable de gérer les communications. Cet outil peut être rattaché au processus application ou au processus débogueur.

Dans l'architecture proposée par COSYTEC la communication est assurée par l'outil CNI [CNI02]. Les fonctionnalités de cet outil seront développées dans les sections ci-dessous.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

6.2.1 Visualiseur

Le visualiseur est un composant dédié au traitement des vues métiers. Ci-dessous, toute une section entière est dédiée à la visualisation des vues métiers. Dans ce paragraphe, nous montrons les principaux modules de ce composant : le module« vue Table », le module « vue Gantt » et le module « vue chart » et le module « vue 3D ».

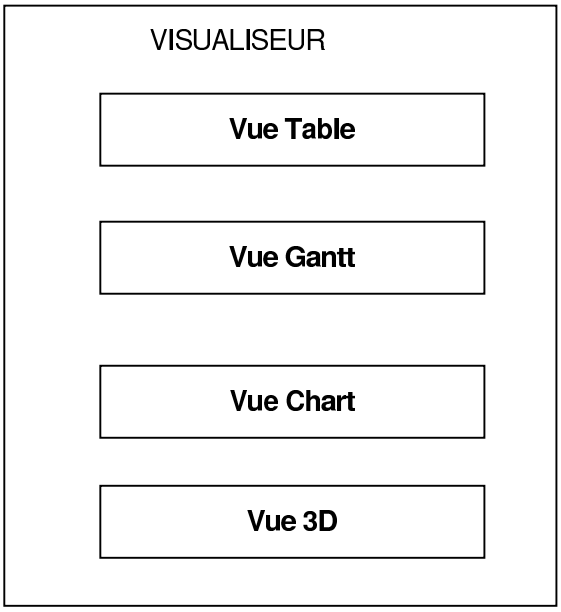

Figure 6-2b : Le composant visualiseur

6.2.2 Gestion des évènements

Le flot de trace envoyé par CHIP est analysé par un parseur qui extrait les informations nécessaires à la création et à l'affichage des vues. Ces informations sont enregistrées dans des structures de données et sont partagées entre différentes vues. L'ouverture d'une vue par l'utilisateur provoque la création ou la mise à jour de son modèle de données. La vue ainsi crée s'affiche dans le *framework*.

Les différentes vues mises en place réagissent aux événements de sélection et de mise à jour de modèle. Ces évènements sont envoyés par un contrôleur appelé: *EventDispatcher*. Ce dernier est chargé de transmettre à toutes les vues enregistrées les évènements graphiques dont il est notifié. Par exemple la sélection d'une tâche dans la vue Gantt est immédiatement notifiée au contrôleur qui se chargera de la retransmettre aux vues dont il a la connaissance. Les autres vues vont ainsi recevoir ce message de changement de sélection et vont se mettre à jour.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

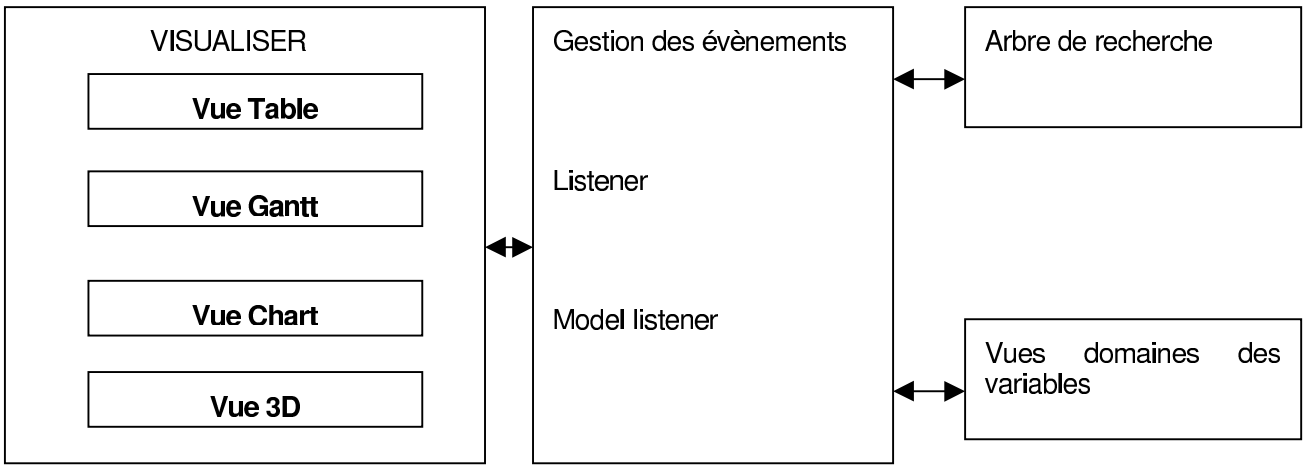

Figure 6-2c : Gestion des évènements

6.3 Environnement de débogage de CHIP

Pour clarifier l'architecture retenue pour CHIP, nous proposons de détailler l'ensemble de ses composants pour montrer les différents développements mis en œuvre dans le cadre de ce projet. Cette architecture doit fonctionner pour les différentes plate-forme de CHIP; à savoir CHIP++ (Prolog + objet), CHIP C (librairie C de CHIP) et CHIP C++ (librairie C++ de CHIP). Cette architecture facilitera l'intégration d'autres solveurs comme GNU-PROLOG. Les composants importants seront présentés dans les sections suivantes.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

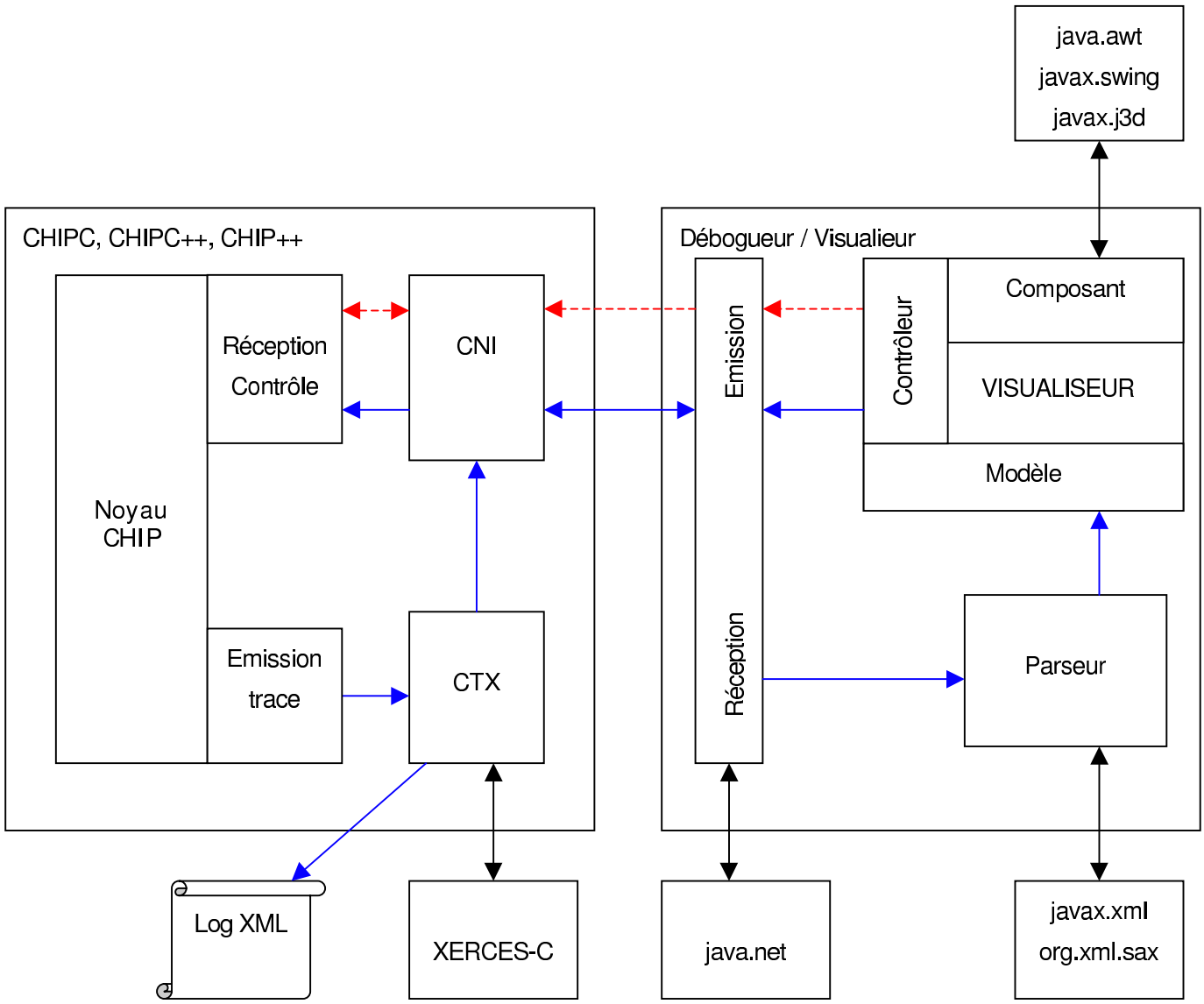

Figure 6-3 : Architecture de l'environnement de débogage de CHIP

7. Principe de fonctionnement

La gestion de la trace à partir du moteur CHIP est assurée par deux processus qui collaborent ensemble :

- Le processus principal est le moteur CHIP, appelé par un programme CHIP utilisateur,
- Le processus secondaire est le thread CNI [CNI02] qui gère la connexion entre le moteur CHIP et le debogueur. Il permet, également de se suspendre et de suspendre le processus CHIP si aucun message n'est parvenu du débogueur.

Le principe de fonctionnement est le suivant : le débogueur envoie des commandes ou des requêtes au moteur CHIP, par le biais de CNI, puis le moteur CHIP exécute l'action reçue, si c'est une commande (envoi de trace, arrêt d'envoi de trace, quitter le programme, …) ou déclenche un traitement et répond à la requête par un envoi de trace XML à travers CNI.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

8. Système PPC CHIP : source de la trace

8.1 CNI (Chip Network Interface)

L'objectif de CNI est de faire communiquer 2 processus (CHIP ou autres) sur des machines NT/UNIX. Cet outil doit supporter les spécifications du projet OADymPPaC. Pour COSYTEC, les applications à déboguer peuvent être développées soit en CHIP++ (CHIP Prolog) ou CHIP C++ (la librairie C++). De plus, il faut prendre en compte le fait que le débogueur soit entièrement écrit en Java.

CNI doit être réalisé en tant que librairie (statique ou dynamique) ***totalement indépendante*** de CHIP++ ou de CHIP/C++.

Pour les besoins de OADymPPaC, des développements conséquents ont été réalisés, testés et en cours d'exploitation. La librairie CNI est désormais intégrée dans l'offre de COSYTEC (Bêta version) depuis le 31 mars 2003.

8.2 CTX (Chip Trace Xml)

Ce module reçoit de la trace brute et propriétaire envoyée par le solveur de CHIP. Il construit un arbre DOM (Document Object Model). Cette structure arborescente offre plusieurs possibilités pour appliquer une variété de transformations pour filtrer la trace en fonction du degré de finesse attendue. Une fois l'arbre final construit en mémoire, nous utilisons le formateur APACHE XERCES pour générer la trace. La trace est soit envoyée dans un fichier pour une analyse off line; soit envoyée au module CNI qui la transmettra au débogueur.

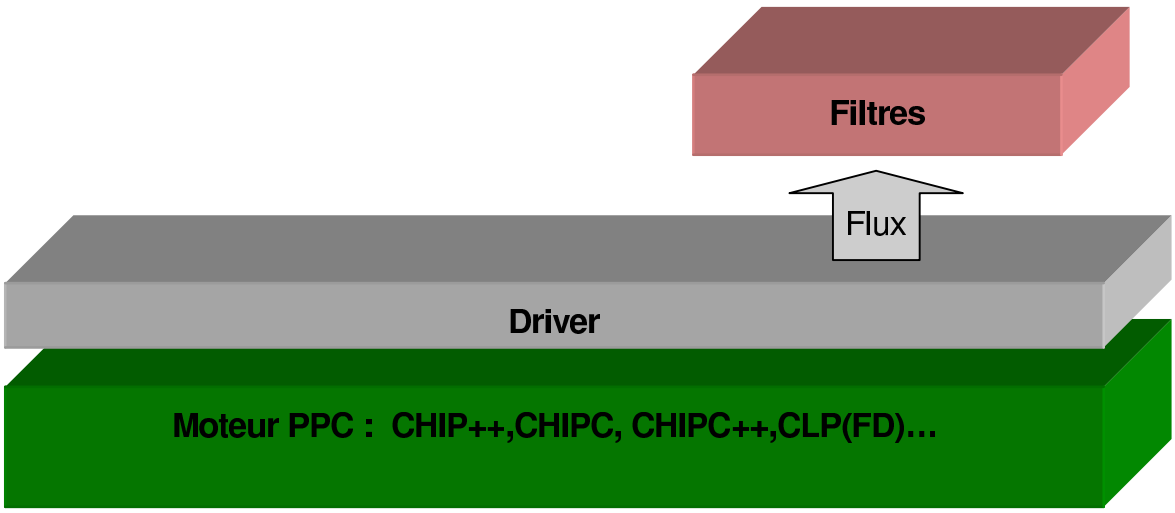

Figure 8-1 : les filtres

Ce module est connu sous le nom de "filtres", les fonctionnalités possibles qu'il peut offrir sont :

- Génération d'un format de trace XML conforme à une DTD;
- Réglage de la profondeur de la trace;
- Définir des modèles de paramétrage type :
  - technique : conséquences (les variables impactées) d'une affectation d'une valeur,
  - sémantique : conséquence de l'affectation d'une tâche à une machine.

Ce composant se présente comme une librairie de fonctions paramétrables.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

8.3 Réception / Contrôle

Pour gérer la communication entre le débogueur et le moteur CHIP par l'intermédiaire de CNI, un protocole est défini. La communication entre ces deux outils se fait par envoi de messages. Ce protocole consiste à encoder, au niveau du débogueur, les commandes et les requêtes pour les transmettre à l'application CHIP.

Les spécifications du protocole sont terminées. L'implantation des spécifications est aussi terminée. Elle est en cours de test. Il est à noter que le protocole peut être étendu.

Ce composant joue un rôle clé dans la gestion de la communication entre le moteur PPC et le débogueur. Il est constamment à l'écoute des commandes de CNI. Il s'assure que les conditions sont réunies pour pouvoir commencer les envois. En particulier il teste que le débogueur n'a pas envoyé de commandes comme l'arrêt du débogueur ou carrément le débogueur s'est déconnecté. Si CNI a déjà reçu une commande d'arrêt de génération de trace, cette information est enregistrée et communiquée au moteur pour prendre acte; par exemple le solveur continue son exécution sans générer de trace.

8.4 Emission trace brute

La génération de la trace nous a amené à effectuer des modifications au niveau du noyau du solveur de CHIP. Cette tâche est maintenant terminée. Quelques modifications ont été apportées à la DTD OADymPPaC pour répondre aux besoins de l'architecture proposée ci-dessus. La trace générée traite principalement la création des variables et la phase d'énumération. La génération de la trace détaillée pour les contraintes est en cours de développement.

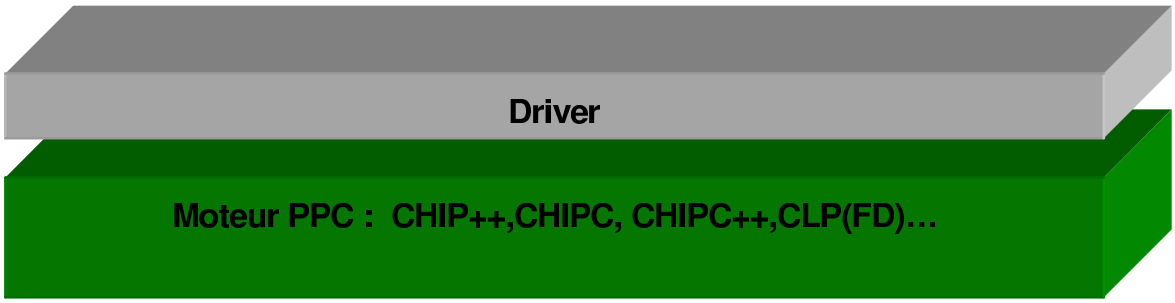

Figure 8-2 : Drivers

Nous rappelons que la trace envoyée par le solveur peut être une trace brute. Aucun filtrage n'a été appliqué (degré de finesse de la trace). La trace est créée grâce à l'adjonction d'un driver définissant un certain nombre de fonctions pour récupérer les évènements produits par le solveur de contraintes.

L'ensemble de ces informations est ensuite filtré. Le niveau de filtrage est réglable, ce qui permet une définition plus ou moins détaillée du flux de trace.

Les différents filtres sont expliqués dans [MI02, RM02].

Ce module offre les fonctionnalités suivantes :

- Paramétrage de la partie spécifique à la plate-forme;
- Paramétrage du degré de finesse de génération de la trace:
  - Utilisateur : mise au point d'une application;
  - Didactique : enseignement et diffusion de la PPC;
  - Ingénieur CHIP : mise au point d'une nouvelle contrainte.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

Ce module se présente comme une librairie de fonctions paramétrables appelées par le moteur de PPC.

9. Système de Débogage et de visualisation : destination de la trace

Cette section décrit l'ensemble des composants du processus de débogage d'un système de PPC. Ce processus est complètement indépendant. Comme nous l'avons expliqué dans les sections ci-dessus, il dialogue avec l'application PPC à travers le composant CNI.

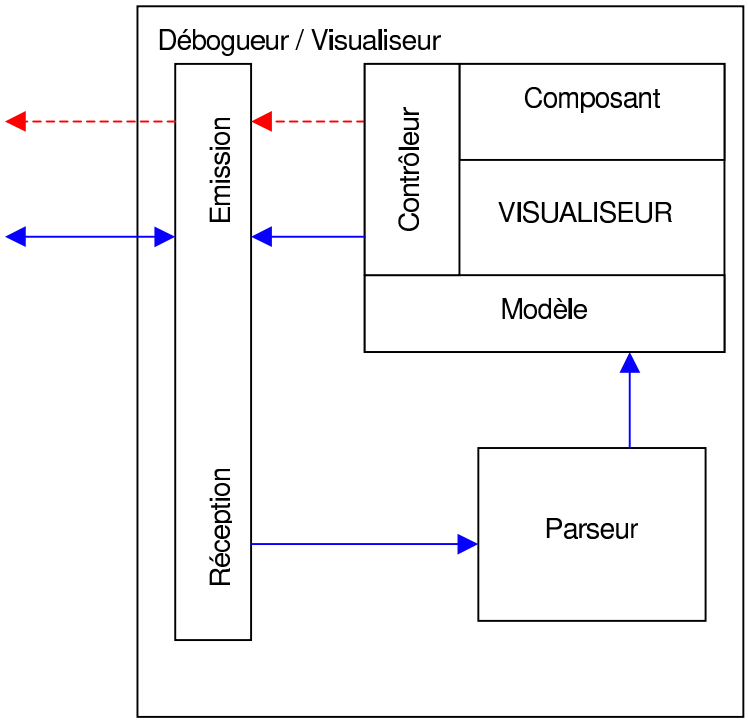

Figure 9-1: le processus débogueur et visualisation.

Les principaux composants sont :

- Emission / Réception des messages
- Contrôleur
- Visual Search Tree
- Visualise

9.1 Emission / Réception

Ce composant traite de l'envoi et de la réception des messages. Nous distinguons principalement deux types de message : les commandes de contrôle et les requêtes du débogueur.

Les commandes de contrôle permettent de dialoguer avec le solveur. Parmi ces commandes, on peut citer la demande de connexion, l'arrêt de la trace, etc. Ces commandes sont en pointillé rouge dans la figure ci-dessus.

Les requêtes du débogueur sont des commandes envoyées au solveur et elles représentent les requêtes de l'utilisateur. Parmi ces requêtes, on peut citer les conséquences de la propagation des contraintes d'une suite d'affectations de variables, un chemin partiel dans un arbre de recherche.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

9.2 Parseur

Les flots de données, transmis par le solveur sous forme de trace XML, suivent les spécifications d'une DTD. Lors de la réception de cette trace, l'outil de visualisation analyse et enregistre les informations nécessaires à la conception des différentes vues (arbre de recherche, visualisation du domaine des variables). Cette analyse est réalisée par un parseur de document XML.

Tous les messages sont en XML, y compris pour les commandes de contrôle. Ce choix facilitera l'évolution du débogueur pour l'enrichir avec de nouvelles fonctionnalités tout en gardant la cohérence des formats de données échangées grâce à la validation selon une DTD.

Lors de la lecture des messages XML, des événements correspondants aux différents tags XML sont générés. Ces évènements provoquent l'appel des méthodes correspondantes. Les méthodes déclenchées sont définies via l'implémentation d'interfaces prédéterminées (SAX, voir sections suivantes).

9.2.1 Paquet

COSYTEC a choisi le mode interactif de mise au point des programmes, c'est à dire l'analyse au fil de l'eau (online) de la trace. La verbosité de XML nous a conduit à mener des expériences sur le découpage de la trace pour éviter de générer des messages trop longs pouvant dégrader les performances des autres modules et principalement l'analyse et la visualisation. Nous avons proposé de nouveaux tags facilitant le découpage et la structuration des messages.

9.2.2 Intégration de plusieurs DTD

Dans l'environnement COSYTEC, plusieurs DTD (trace, vues métiers, protocole de communication) cohabitent ensemble pour le besoin d'une application. Nous avons également proposé des solutions qui ont permis d'enrichir la DTD OADymPPaC.

9.2.3 Délégation

L'environnement de mise au point et de visualiser est amené à évoluer. Pour faciliter l'intégration de nouveaux modules, les mécanismes de délégation sont essentiels pour la conception, le développement et l'intégration.

9.3 Visualiseur

Cette section traite principalement de la visualisation de la trace XML. Le visualiseur comprend deux composants :

Visual Search Tree : ce composant permet la visualisation et la consultation d'un arbre de recherche.

Visualise : ce composant traite de visualisations agrégées (contraintes globales, vues métier).

A l'autre bout de la chaîne, les outils de visualisations et de mise au point réceptionnent et utilisent les informations transmises par le solveur. Chaque outil -des différents partenaires- réalise lui-même son propre filtrage -en plus du filtrage effectué au niveau du solveur-, ce qui permettra une conception des outils de débogage indépendante du solveur et la possibilité de faire fonctionner plusieurs outils simultanément sur une plate-forme donnée.

Une fois le flot de données décodé et les informations utiles enregistrées, la vue spécifique de l'arbre de recherche affiche graphiquement les nœuds explorés par la procédure de recherche. L'utilisateur peut alors interagir avec le système.

Lorsqu'un nœud de l'arbre est sélectionné, le système va réinstaller, par un mécanisme que nous détaillerons un peu plus loin, l'état de la procédure de recherche à ce nœud précis de l'arbre; en d'autres termes redonner aux variables et contraintes la configuration qu'elles avaient à ce point précis de l'exécution du programme et remettre en place les décisions prises sur ce chemin à l'état original.

Un mode opératoire on-line sera ultérieurement mis en place. Ce dernier consistera en fait, en une coopération accrue entre les outils de débogage et le solveur de contraintes symbolisée par un échange

Confidential © 2004 COSYTEC SA. All Rights Reserved.

d'informations (requêtes des outils de visualisations sur les données à afficher selon l'action effectuée par l'utilisateur et réponse du solveur après calcul de ces données).

9.4 Composants externes à CHIP

Pour la génération et la lecture (parseur) du code XML, le choix s'est porté sur les recommandations de W3C - http://www.w3c.org. Les standards utilisés sont DOM et SAX. Dans cette section nous donnons un aperçu de ces API.

API DOM (Document Object Model): Cette interface de programmation utilise une modélisation objet des documents XML, dans notre cas la trace PPC. Ces objets possèdent des interfaces facilitant leur manipulation par programme. Pour les besoins de notre application, nous nous sommes appuyés sur les spécifications de niveau 1 - partie 1; c'est à dire DOM core permettant de manipuler les objets XML.

API SAX (Simple API for XML): Cette interface de programmation permet d'analyser un document XML sur évènement. Nous avons implanté une librairie de fonctions de traitement d'évènements pour les différentes balises des éléments de la DTD OADymPPaC (plus extensions).

Les API DOM et SAX utilisées pour les besoins de notre architecture s'appuient sur les implémentations du groupe de travail XML Apache. Ce dernier maintient et assure l'évolution de Xerces-C++. Ce parseur XML est également une référence dans le monde Java. Pour plus d'information, nous recommandons au lecteur de se référer aux adresses suivantes : xml.apache.org/xerces-j et http://www.w3c.org.

9.5 Conception orientée objet

La conception Orientée Objet est sans nul doute l'une des méthodes la plus complète et la plus adaptée pour la définition des outils complexes et génériques que ce projet souhaite mettre en place. La conception et le développement des différents composants de visualisations s'est appuyée sur l'utilisation de Design Patterns [GHJV] tel que le mémento, l'observateur, le décorateur ou encore le composite.

Ce choix facilitera la structuration des composants de visualisation pour faciliter l'implémentation, simplifier la maintenance et permettre la réutilisation des objets développés.

9.6 Le langage de développement

Java est à l'heure actuelle le meilleur outil permettant une portabilité et une intégration très complète d'un système. En plus de sa portabilité, ce langage possède des atouts suivants :

- Orienté Objets
- Très bonnes bibliothèques graphiques (JTree, JTable, Java 2D, Java 3D, …) [WC99]
- Permet des développements complexes
- Robustesse
- Grand nombre de bibliothèques diverses et variées (communication, graphique, …)

L'ensemble des composants du processus de débogage sont développés en Java.

9.7 Le framework MVC

Le modèle MVC (Model - View - Controller) [GR97] est souvent utilisé pour la création des interfaces graphiques utilisateur (Graphical User Interface) d'une application. Les GUIs sont la représentation d'un arrangement de composants (text fields, scrolling lists, etc.). Cet arrangement représente la vue (*View*) d'un certain modèle (*Model*). Ce modèle encapsule des objets représentant la logique métier. Vue et modèle sont totalement distincts. Le contrôleur (controller) associe l'interaction de l'utilisateur avec une mise à jour éventuelle de la vue.

Les apports du modèle MVC sont non négligeables :

- Possibilité de définir plusieurs vues, dynamiques, partageant les mêmes données.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

- Flexibilité par la possibilité d'ajouter ou de retirer des composants, ajout d'une nouvelle variable dans la table par exemple.
- L'implémentation est la combinaison des patterns observateur, stratégie et composite.

La séparation du modèle de données et des vues permet de définir plusieurs représentations graphiques différentes pour un même modèle.

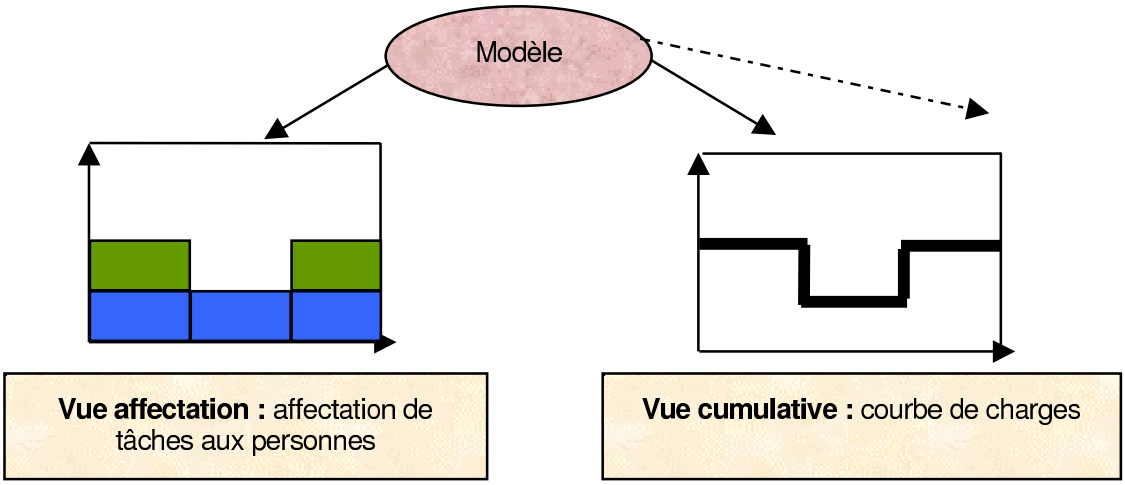

Figure 9-2 : Modèle MVC

9.8 Visualisation de l'arbre de recherche

Cette section décrit les grandes lignes de l'outil de visualisation de l'arbre de recherche [SA99].

9.8.1 Principe de fonctionnement

L'outil fonctionne en étroite collaboration avec le moteur de contraintes. La procédure d'énumération est augmentée de quelques primitives de l'outil pour construire la structure de l'arbre de recherche des nœuds visités. Chaque nœud de l'arbre de recherche est un objet contenant le nœud père, le nœud courant, la décision (choix de variable et de valeur). Le paramétrage de l'outil arrête le moteur des contraintes quand un des critères d'arrêt suivants est atteint : (1) le nombre de solutions recherchées, (2) le nombre d'échecs autorisés, (3) le nombre de nœuds visités ou enfin (4) le temps limite. L'outil crée l'arbre de recherche et l'affiche. L'utilisateur peut naviguer pour explorer les différents chemins et les nœuds qui le composent. Le choix d'un nœud déclenche l'affectation des valeurs aux variables et le système de propagation des contraintes en parcourant le chemin jusqu'au nœud choisi. Ainsi, l'outil crée l'état de résolution correspondant à l'état premier du moteur de contraintes. L'outil prend en compte l'ensemble des fonctionnalités de l'environnement CHIP : aucune restriction sur les domaines, les contraintes syntaxiques, les contraintes globales, les méta-heuristiques, les primitives d'optimisation (branch and bound).

Les types de vues : L'outil permet de visualiser un ensemble de types de vues, correspondants à des concepts, de la procédure de recherche en se basant sur les quatre concepts suivants :

Etat : affiche une vue de l'état particulier de l'arbre de recherche comme, par exemple, les domaines de toutes les variables d'un nœud sélectionné de l'arbre de recherche. La mise à jour des informations sur les domaines et les contraintes lors d'un passage d'un nœud à un autre est automatique.

Chemin : affiche le chemin d'exploration de l'arbre de recherche depuis le nœud racine jusqu'au nœud sélectionné. Cette vue permet de montrer les changements des domaines des variables et des contraintes à travers les choix successifs effectués et qui ont mené jusqu'au nœud sélectionné.

Arbre : cette vue permet d'agréger les informations des sous arbres de recherche.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

Evolution : affiche les informations sur tous les nœuds explorés précédent le nœud sélectionné. Ce concept est intéressant pour analyser les statistiques sur les échecs.

Visualisation de l'arbre de recherche : Chaque nœud de l'arbre de recherche peut être colorié avec son nom, son niveau (profondeur dans l'arbre), son numéro dans la liste des variables (argument de la procédure de recherche) ou sa valeur. Le nœud racine est particulier et correspond à l'état du programme PPC après la pose des contraintes. Les nœuds échecs sont coloriés différemment des nœuds feuilles (des solutions). Il est possible de réduire la taille du graphe en regroupant les sous-arbres échecs en un seul nœud spécial de type échec. Il est à noter que les nœuds des derniers niveaux sont assez souvent fixés par propagation.

Visualisation des variables : Ces vues permettent principalement d'aider l'utilisateur à comprendre l'impact d'une décision (variable/valeur + propagation des contraintes) sur l'ensemble des autres variables domaines.

Visualisation des contraintes : L'outil permet de visualiser l'ensemble des contraintes du programme PPC ou l'évolution d'une contrainte particulière. Ces vues permettent de comprendre, d'une part, la structure du problème et d'autre part, le raisonnement global du réseau de contraintes du modèle. L'outil permet de visualiser également l'impact des contraintes, contrainte par contrainte et pour chacun des nœuds du chemin, sur un chemin (échec, solution ou partiel) de l'arbre. Ces vues seront détaillées lors de la présentation du composant Visualise.

Visualisation de la propagation : L'affectation d'une valeur à une variable peut déclencher une propagation en chaîne de plusieurs contraintes. Ce type de vue permet de visualiser l'impact d'une décision variable/valeur et l'apport de chacune des contraintes réveillées.

9.8.2 Application du modèle MVC

La représentation arborescente verticale permet de très rapidement se rendre compte de l'efficacité d'une procédure de recherche. En appliquant les principes de MVC, nous disposons d'un modèle qui est la structure hiérarchique des nœuds de l'arbre de recherche. Chacun de ces nœuds sera rattaché aux informations (évènement de propagations, états des variables et des contraintes …) issues directement de la trace. Suivant le principe de MVC, ce composant sera divisé en trois parties :

Modèle :

Le modèle est une structure arborescente décrivant le cheminement de l'heuristique. Cet arbre est construit directement à partir des événements enregistrés dans la trace.

Vue :

La vue généralement issue de ce modèle est une représentation graphique directe de cet arbre de recherche.

Contrôle:

Le contrôleur aura principalement deux fonctions :

- Assurer le synchronisme de la représentation arborescente par rapport au reste du système d'analyse et de visualisation.
- Répondre aux interactions de l'utilisateur (gestion des événements).

A partir de ce système, on peut associer des styles à chaque nœud en ayant recours au principe de décoration. Une vue ainsi enrichie pourra alors mettre en évidence des particularités au niveau des nœuds.

9.8.3 Vues principales d'un arbre de recherche

La première vue consiste à afficher uniquement l'arbre de recherche. Cette vue permet à l'utilisateur :

- de localiser les chemins ayant conduit à des solutions (différentes solutions avec différents coûts);
- de localiser les nœuds échecs, plus ils sont profonds moins est bonne la stratégie appliquée;
- d'apprécier la combinatoire, largeur de l'arbre de recherche, profondeurs des sous arbres.

L'analyse d'un arbre de recherche permet d'analyser plusieurs cas. Nous citerons que deux cas pour illustrer l'intérêt de cette vue.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

Cas 1 - il n'y a pas de solution : pour un expert, cette vue permettra d'améliorer la stratégie de choix de variable et de valeur afin d'explorer différents espaces pouvant conduire à au moins une solution. L'utilisation de la recherche partielle est un moyen efficace pour traiter ce type de problème.

Cas 2 - il existe au moins une solution : pour un expert, cette vue permettra d'analyse les échecs voisins su sous arbre ayant conduit à cette solution; le but recherché consiste à améliorer le choix de certaines variables et de certaines valeurs, préférences, pour converger rapidement vers les solutions. Ces préférences sont assez souvent nécessaires dans beaucoup d'applications. Les traitements associées de ces préférences sont effectués avant la phase d'énumération.

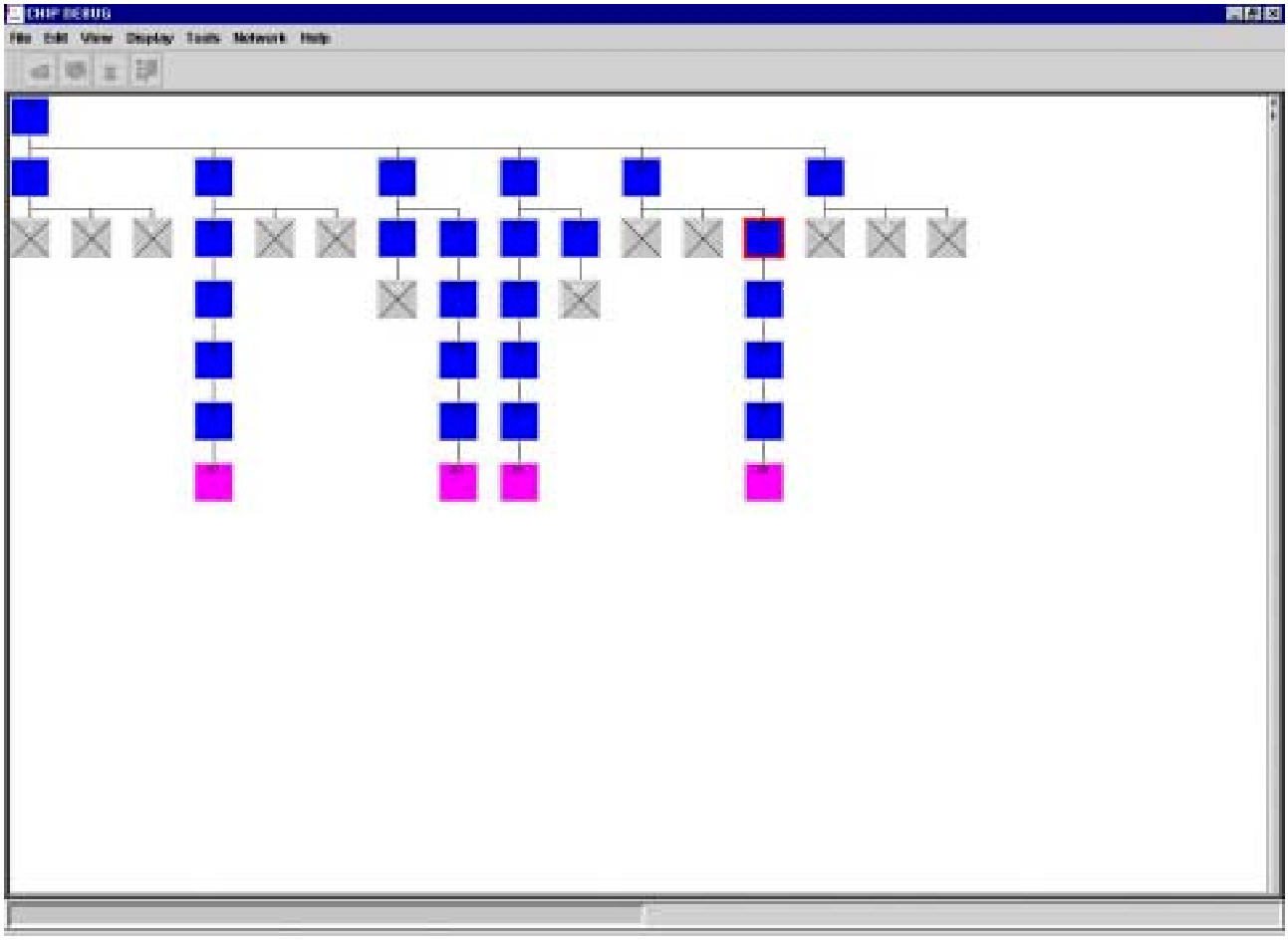

Figure 9-3 : Vue simple d'un arbre de recherche.

9.8.4 Vue du domaine des variables

La figure ci-dessous représente une des deux vues permettant de visualiser le domaine des variables. Nous rappelons que ces domaines sont fournis par le solveur conformément à une DTD XML. Les données XML sont analysées et sont attachées à l'arbre de recherche sous forme de mémentos. Une description détaillée est donnée dans [RM02].

Confidential © 2004 COSYTEC SA. All Rights Reserved.

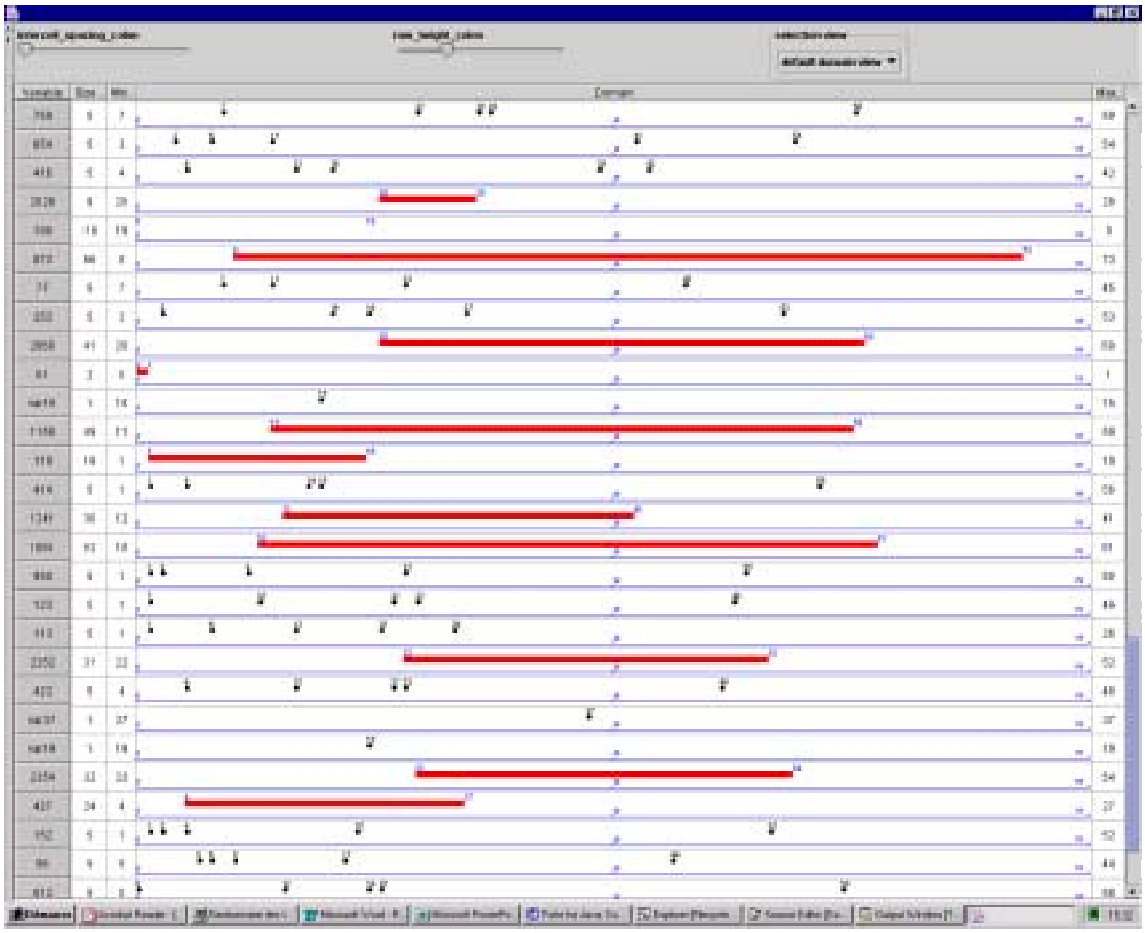

Figure 9-4 : Vue des domaines des variables.

Cette vue représente une table constituée de plusieurs lignes et colonnes. Chaque ligne symbolise la vue d'une variable. Une ligne est constituée de cinq cellules illustrant des informations différentes de la variable. Ainsi, de gauche à droite :

- La première cellule donne le nom de la variable.
- La deuxième, indique la taille du domaine.
- La troisième, spécifie le minimum du domaine
- La quatrième, fournit un affichage graphique du domaine de la variable.
- La cinquième et dernière cellule indique le maximum du domaine.

Cette vue indique de façon précise le domaine. Elle fournit également les valeurs des différents sous-domaines. L'échelle varie en fonction du minimum et du maximum des domaines de l'ensemble des variables; elle est ainsi remise à jour lorsque le minimum ou le maximum globale de la table change par l'installation d'un nouveau mémento. Cette mise à jour permanente de l'échelle permet à la fois d'avoir la possibilité de comparer les domaines des variables, mais aussi d'optimiser l'espace pour l'affichage -afin d'avoir des vues plus claires.

Confidential © 2004 COSYTEC SA. All Rights Reserved.

Il existe d'autres vues pour représenter les domaines. La vue ci-dessous montre une vue matricielle variable / valeur. Chaque ligne représente une variable. Chaque colonne représente une valeur. Chaque cellule de la table indique la présence ou non de cette valeur dans le domaine d'une variable.

Figure 9-5 : Vue des domaines des variables.

9.8.5 Vue combinée

La vue combinée consiste à présenter dans deux fenêtres la vue de l'arbre de recherche et la vue des domaines des variables. Cette vue combinée permet à l'utilisateur de faire des opérations d'inspection et d'analyse, comme par exemple connaître les états des domaines des variables à un nœud particulier de l'arbre de recherche. En cliquant sur un nœud de l'arbre de recherche, l'utilisateur déclenche une requête

Confidential © 2004 COSYTEC SA. All Rights Reserved.