# Environnement de résolution de problèmes distribué

# Distributed Problem Solving Environment

Thibault PARMENTIER Danielle ZIÉBELIN François RECHENMANN

Projet SHERPA, INRIA Rhône-Alpes

655 Avenue de l'Europe, 38330 Montbonnot, France

{Thibault.Parmentier, Danielle.Ziebelin, Francois.Rechenmann}@inrialpes.fr

## Résumé

*L'utilisation pertinente de bibliothèques spécialisées de programmes n'est pas toujours aisée : choisir, enchaîner et exécuter les modules adéquats lors de la résolution d'un problème demande souvent, de la part de l'utilisateur, une excellente maîtrise du domaine concerné et une bonne connaissance de l'organisation même des bibliothèques utilisées. Afin de décharger l'utilisateur de ces difficultés, l'intérêt principal d'un environnement de résolution de problèmes (ERP) est d'offrir un pilotage automatique de bibliothèques. Pour cela, les ERP s'appuient sur des stratégies de résolution (connaissances méthodologiques) facilement compréhensibles par l'utilisateur. Ils utilisent les connaissances méthodologiques pour identifier, spécialiser et exécuter les procédures les plus adaptées au problème posé. L'utilisateur contrôle la résolution au moyen d'interfaces lui permettant de visualiser toutes les étapes d'une stratégie au cours de son exécution, et de visualiser les résultats produits.*

*Nous proposons un environnement de résolution distribuée de problèmes (ERDP) à un ensemble d'utilisateurs d'un même ERP. Cet ERDP permet de mettre en commun les connaissances méthodologiques développées sur les différents sites. Chaque site permet l'utilisation de façon transparente des ressources (bibliothèques et connaissances méthodologiques) d'autres sites. Pour cela, nous avons adapté conceptuellement et technologiquement un modèle de tâches mono-site afin qu'il supporte des résolutions multi-sites. La mise en œuvre de résolutions distribuées nécessite :*

- *la définition de mécanismes de localisation des compétences, ce qui implique une normalisation de la description des tâches sur tous les sites,*
- *l'élaboration d'un processus d'interaction entre les tâches réparties sur les différents sites,*
- *le choix technologique de concevoir des tâches à l'aide d'un langage pouvant migrer à travers le réseau et s'exécuter localement. Par contre, le code peu portable des bibliothèques s'exécute à distance.*

## Mots-Clefs

*Environnement de résolution de problèmes, pilotage de bibliothèques, application répartie, modèle de tâches, résolution collaborative et distribuée de problèmes.*

## Abstract

*Using specialized libraries in a relevant way is not always easy: choosing, linking together and executing appropriate modules during the solving process often require a masterized control of both the related domain and a good knowledge of the whole organization of involved libraries. In order to exempt the user from these difficulties, the main goal of a problem solving environment (PSE) is to provide the user with the automatic driving of librairies. In this purpose, the PSE is based on solving strategies (methodological knowledge) meaningfull for the user. Methodological knowledge is used to identify, specialize and execute procedures best suited to the problem stated. The user controls the solving process by interface enabling him to visualize all the stages of a strategy during its execution, and to visualize the results produced.*

*We propose a distributed problem solving environment (DPSE) to a group of users of a same PSE. This DPSE makes possible to share the methodological knowledge developed on different sites. Every site is allowed to use openly the resources (libraries and methodological knowledge) of other sites. Our solution is drawn from a conceptual and technical adaptation of the so-called mono-site task model such that it gets capable of performing multi-sites problem solving. The implementation of a distributed problem solving process requires :*

- *to define the mechanisms that locate the skills using the normalization of task descriptions on all sites,*
- *to work out the process of interaction among distributed tasks on different sites,*
- *to create tasks over a movable programming language while the library binaries are allowed to be executed from afar whenever they are not easily portable. This corresponds to a technological choice we made.*

## Keywords

*Problem solving environnement, task model, distributed problem solving, distributed application.*

## 1. Introduction

L'utilisation pertinente de méthodes de résolution de problèmes scientifiques comme, par exemple, la résolution d'un systèmes d'équation aux dérivées partielles [24], l'analyse statistique d'un ensemble de données ou l'utilisation de techniques d'analyse de données, etc., nécessite d'avoir acquis une somme de connaissances importante sur ces méthodes. En effet, il n'est pas rare de voir un utilisateur, non spécialiste d'un de ces domaines scientifiques, n'utiliser qu'une seule méthode très générale pour résoudre tous problèmes qu'il aura jugés similaires et obtenir parfois des résultats erronés. Il enchaîne ainsi toujours selon la même séquence, les mêmes modules de

sa bibliothèque sans connaître vraiment leurs conditions d'application ou l'impact de cette séquence sur la solution.

Le premier pas pour rendre ces méthodes accessibles a été la création d'environnements intégrés comme par exemple pour les mathématiques : Macsyma, Matlab ou Mathematica [17]. Ces systèmes offrent une combinaison complète de fonctionnalités graphiques, symboliques et numériques. Les procédures des bibliothèques ne sont accessibles par l'utilisateur qu'au travers d'une interface homogène ce qui lui permet de voir toujours le même concept ou la même donnée sous la même présentation. La bibliothèque et l'environnement forment un tout insécable qui permet à l'utilisateur d'accéder d'une façon uniforme aux données qu'il manipule.

La conception d'un environnement de résolution de problèmes (ERP) a le même objectif : fournir un environnement intégré et piloter des bibliothèques de programmes [19]. Un ERP propose en plus une modélisation de la spécialité concernée au travers d'une base de connaissances représentant [20] :

- d'une part les entités du domaine (cf. Figure 1). Nous les appellerons connaissances de domaine. Elles représentent les concepts manipulés (variables, domaines de définition, nuage de points, maillage…),
- d'autre part les stratégies de résolution de problèmes qui représentent les méthodes de résolution (cf. Figure 1). Ces connaissances méthodologiques exprimées sous forme de stratégies permettent une évaluation du problème à résoudre. Elles conduisent à une sélection, un enchaînement et une exécution de la manière la plus pertinente des modules ou des programmes d'une bibliothèque [21][22][14].

Le projet Sherpa de l'INRIA Rhône-Alpes a développé avec la société Ilog, un ERP générique PowerTasks [18] dont la base de connaissances est construite à l'aide d'un modèle à objets. Les connaissances de domaine sont représentées grâce à des classes et des instances, les connaissances méthodologiques sont décrites à l'aide de classes et d'instances particulières : les tâches [4][5]. Une tâche permet une représentation compacte et structurée d'un problème et des connaissances nécessaires à sa résolution aussi bien en ce qui concerne la description de ses données et résultats que de sa stratégie de résolution (cf. section 2).

PowerTasks a été utilisé dans le cadre d'un projet rassemblant des laboratoires de biologie financé par le Groupement de Recherche et d'Etudes sur les Génomes afin de permettre une analyse de génomes. Le système obtenu, ImaGene permet l'analyse et l'annotation de génomes bactériens complets [15][16]. Il donne à l'utilisateur biologiste la possibilité d'exploiter et d'exécuter de manière transparente des stratégies d'analyse de séquences fondées sur plusieurs bibliothèques offrant des méthodes complémentaires. ImaGene recherche et identifie, sur une séquence, les objets biologiques pertinents qu'elle peut contenir comme par exemple les gènes, les signaux de régulation… ImaGene dispose, pour cela, d'une interface permettant d'afficher sur le même écran l'ensemble des résultats produits par plusieurs stratégies, pilotant plusieurs bibliothèques.

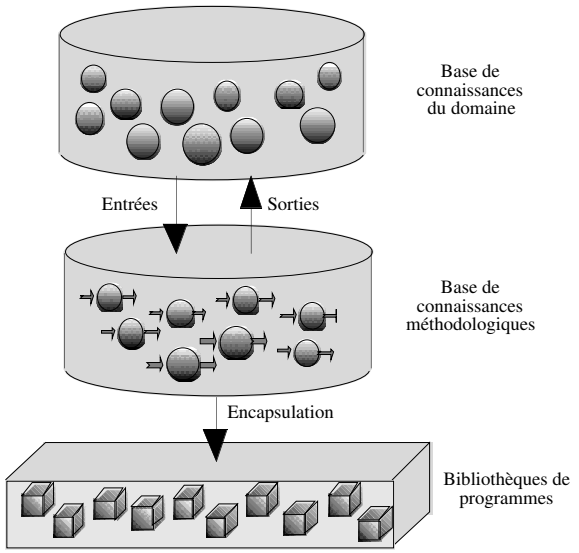



**Figure 1 : l'ERP est construit autour de deux types de connaissances qui lui servent à utiliser de façon convenable des modules et des programmes de bibliothèques : les connaissances méthodologiques qui contiennent les stratégies de résolution de problèmes, et les connaissances de domaine qui se composent des entrées dont les stratégies ont besoin et des sorties qui sont produites lors des résolutions. Les programmes des bibliothèques sont encapsulés et pilotés par les connaissances méthodologiques.**

Installer et utiliser un ERP permet effectivement de fournir une réponse satisfaisante au pilotage de bibliothèques. Il reste néanmoins les difficultés d'installation et les mises à jour de celles-ci. Le code de ces bibliothèques s'avère souvent peu portable, et difficile à installer (fractionnés en plusieurs parties, appels à d'autres bibliothèques d'utilitaires, incompatibilités de versions de compilateurs…) et entraîne souvent l'achat de matériels ou de logiciels spécialisés (nécessité de puissance de calculs, utilisations de bases de données spécifiques). Pour illustrer cette dernière difficulté, nous présentons brièvement le cas du logiciel ImaGene, qui, dans sa version actuelle, encapsule trois bibliothèques, deux d'entre elles nécessitent des ressources spécialisées.

- L'une effectue des comparaisons de séquences entre organismes en retrouvant les objets biologiques pertinents à partir de la position de ces objets sur une séquence analysée d'un organisme proche. La bibliothèque encapsulée utilise donc une base de données contenant plusieurs organismes partiellement ou complètement analysés. Lors de l'installation de cette bibliothèque, il est nécessaire d'avoir une copie de la base de données complète qu'il faut dans ce cas réactualiser régulièrement. L'autre solution consiste à avoir la possibilité d'interroger à distance la base de données initiale.
- La deuxième bibliothèque effectue une recherche statistique en utilisant les fréquences d'apparition de nucléotides dans les gènes ou dans tout autre objet biologique pertinent. L'installation de cette bibliothèque se traduit donc par l'installation à la fois

des programmes statistiques mais aussi des données relatives à des organismes totalement ou partiellement analysés, et enfin des procédures de tolérance à l'erreur pour permettre des recherches tenant compte d'erreurs de séquençage. L'installation des trois parties de cette bibliothèque, et la mise à jour de l'une d'entre elle indépendamment des deux autres ne se fait pas sans difficultés notamment pour des non spécialistes de cet outil.

Afin de supprimer ces difficultés d'installation et maintenance [13], nous proposons une extension de l'ERP PowerTasks vers un environnement de résolution distribuée de problèmes (ERDP). Grâce à cette version de PowerTasks distribué, les sites utilisant cet ERP deviennent clients des compétences d'autrui et exécutent à distance les modules et programmes de bibliothèques. Par contre, chaque site est serveur de ses propres compétences, assure la maintenance des modules et bibliothèques qui sont sous son contrôle, en permet l'accès total ou partiel et en assure l'utilisation sur son matériel spécialisé avec ses propres licences d'utilisation.

La possibilité de mettre en commun les connaissances de domaine et méthodologique des différents sites présente de nombreux avantages et apporte une solution aux problèmes évoqués dans le paragraphe précédent. En section 3, nous présenterons les choix architecturaux de PowerTasks distribué, et particulièrement les options technologiques qui nous ont guidés dans cette réalisation (sections 3.1 et 3.2). La mise en œuvre d'une telle architecture distribuée a demandé une extension du modèle de tâches (cf. section 2) de nouveaux attributs ont été introduits et différents protocoles de communications entre les tâches ont été définis (cf. section 4). Enfin, nous montrerons comment chacun des sites assure la maintenance et la diffusion de stratégies associées à une ou plusieurs bibliothèques de méthodes (cf. section 5).

## 2. Un modèle de connaissances méthodologiques : les tâches.

Afin de permettre une représentation à un haut niveau d'abstraction des connaissances et des résultats [4], d'en donner un accès aisé et d'en faciliter la maintenance, nous avons choisi de représenter la connaissance au travers d'un modèle à objets. Ce choix permet de décrire de façon similaire des connaissances de domaine et méthodologiques.

La représentation des connaissances au sein de l'environnement PowerTasks s'effectue à l'aide de trois différents types d'entités :

– la description du domaine est faite par un modèle de représentations de connaissances à objets de type classe/instance ; (cf. Figure 2)

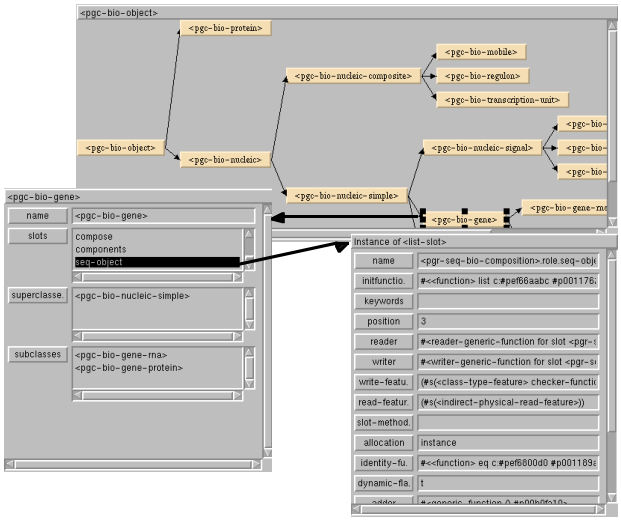



**Figure 2 : Dans l'application ImaGene, la base de connaissances de domaine représente des objets biologiques décrits sous forme d'une hiérarchie de classes. L'éditeur de gauche permet de visualiser la classe « gène » et l'éditeur de droite permet de visualiser l'attribut «séquence» représentant la séquence d'ADN associée à ce gène. Pour une instance de gène donnée, les informations contenues dans les attributs seront précisées et en particulier la séquence sera décrite grâce à un alphabet de 4 symboles (A, T, G, C).**

– les bibliothèques de programmes et de modules fournissent à l'ERP ses outils de base. L'ERP récupère ainsi toute la puissance des bibliothèques spécialisées ;
– les stratégies de résolution sont construites au-dessus des bibliothèques de programmes à l'aide de tâches afin de permettre une utilisation appropriée de celles-ci.

Comme nous l'avons défini dans la section 1, une tâche associe la description d'un problème et sa stratégie de résolution. Cette stratégie est représentée, dans notre modèle de tâches, sous forme d'un graphe de spécialisation et décomposition en sous-problèmes de complexité de plus en plus faible. Au niveau le plus bas, la description d'un problème élémentaire se réduit à une tâche et sa stratégie de résolution à l'appel d'un module ou d'un programme de bibliothèque. La base de connaissances méthodologiques de l'ERP est donc constituée de hiérarchies de tâches. Leurs constructions se font de façon déclarative à l'aide d'un éditeur graphique, avec une explicitation des liens de spécialisation et de décomposition entre les tâches et sous-tâches.

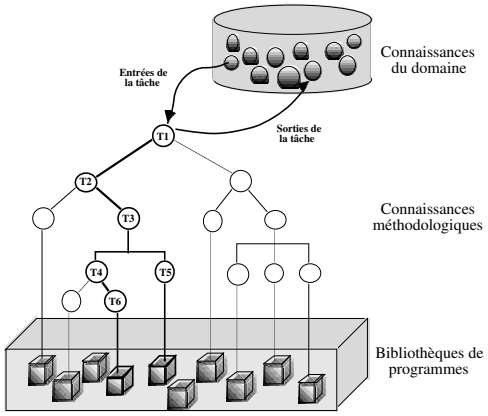



**Figure 3 : Description détaillée des liens entre les trois composantes principales de l'ERP PowerTasks (connaissances méthodologiques, connaissances de domaine, bibliothèques). Par exemple, la résolution de la tâche T1 utilise les données fournies dans la base de connaissances de domaine, T1 se spécialise en deux sous tâches, si la tâche T2 est exécutée, les données qui sont nécessaires à T2 lui sont transmises via T1, T2 se spécialise, elle-même, en deux tâches : une tâche élémentaire qui encapsule un programme de la bibliothèque et une tâche de haut niveau T3, T3 se décomposent T4 et T5 (tâche élémentaire), T4 se spécialise en T6 (tâche élémentaire).**

Nous pouvons distinguer deux catégories de tâches. Les tâches de haut niveau qui se décomposent en sous-tâches et des tâches élémentaires qui encapsulent un programme ou un module de bibliothèque. Les tâches de haut niveau permettent à l'utilisateur un apprentissage via les stratégies (ie. enchaînement des sous-tâches) qu'elles mettent en œuvre lors des résolutions de problèmes (cf. Figure 3). En revanche, l'objectif d'une tâche élémentaire est de servir de « boîte noire ». Son rôle se limite à encapsuler un programme ou un module de bibliothèque du point de vue de sa fonctionnalité. Une tâche élémentaire contient donc la version la plus performante et la plus à jour du module ou programme encapsulé.

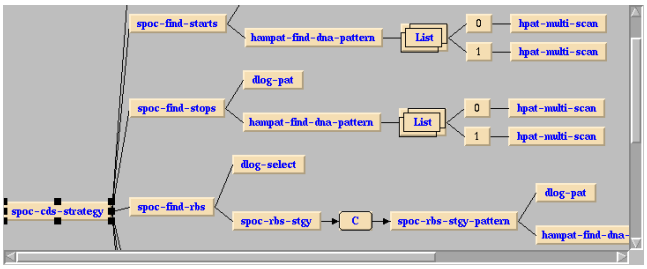



**Figure 4 : L'éditeur graphique de PowerTasks permet de visualiser une stratégie de résolution « spoc-cds-stategy »incluse dans le système ImaGene. « spoc-cds-stategy » permet d'identifier des motifs pertinents sur une séquence de génomes à partir de motifs proposés par l'utilisateur. La tâche « spoc-cds-stategy » est composée de trois tâches « spoc-find-starts, spoc-find-stop, spoc-find-rbs », et celles-ci se décompose en sous-tâches composés de boucle réalisés sur des listes d'éléments, ou de choix C .**

La représentation interne d'une tâche (cf. Figure 5) est une classe composée de deux ensembles d'attributs qui décrivent :

- le problème grâce aux *données* manipulées et *résultats* attendus. Chaque donnée ou résultat correspond à un attribut, dont le type peut être simple (réel, entier, chaîne…) ou faire référence à une classe d'objets de la base ;
- la stratégie de résolution par des *sous-tâches* (cf. Figure 4). À une tâche est associée une décomposition et/ou une spécialisation. La décomposition fait appel à une liste de sous-tâches et à des opérateurs (séquence, choix, itération) qui seront interprétés au cours de la résolution. La spécialisation consiste à préciser les entrées et les conditions d'application de la tâche et de particulariser la méthode de résolution en fonction de ces conditions.

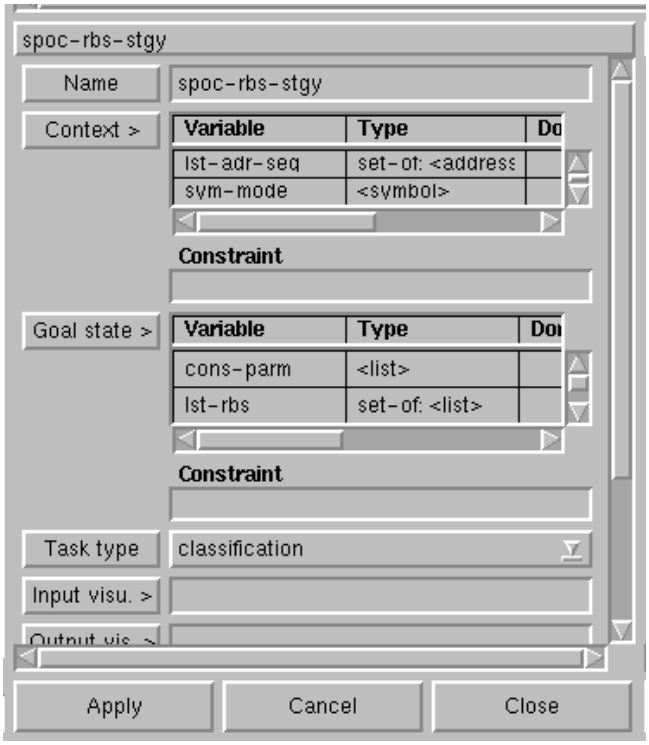



**Figure 5 : Structure interne de la tâche spoc-cds-strategy du système ImaGene. Les attributs de la tâche sont présentés à l'aide d'un éditeur, ils peuvent y être modifiés. « Context » représente l'ensemble des données de la tâche, « Goal state » les sorties. Cette tâche va être soumise lors de son exécution dans un premier temps à une spécialisation, elle est donc définie par le « Task type classification ».**

Lors d'une résolution de problème, le moteur d'inférences de PowerTasks choisit la stratégie de résolution la plus appropriée en fonction du contexte. Pour cela, la résolution d'une tâche passe par l'exécution en alternance de deux phases essentielles :

- la phase de spécialisation. Elle utilise un mécanisme qui classifie la tâche en fonction de ses données et des conditions décrivant le contexte. Cette classification permet de sélectionner la sous-tâche, qui répond aux caractéristiques du problème posé, la plus spécifique ;
- la phase de décomposition utilise des opérateurs de choix, séquence et itération. Elle permet de décomposer une tâche en sous-tâches plus simples.

L'alternance de ces deux phases (caractérisation et décomposition) engendre une résolution opportuniste puisque l'enchaînement des tâches n'est choisi qu'en cours d'exécution [14] en fonction des données du problème. Pour chaque résolution, le système sélectionne la stratégie la plus appropriée. (cf. Figure 6)

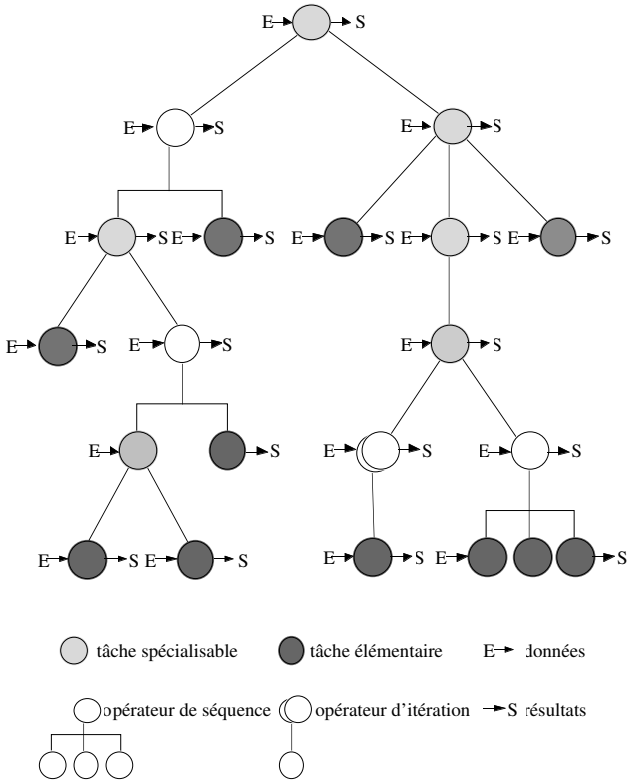



**Figure 6 : la stratégie de résolution de problème est construite au-dessus d'une spécialisation. Le moteur d'inférences va donc commencer par spécialiser le problème à résoudre en fonction des données. Puis il enchaînera successivement les phases de spécialisation et décomposition jusqu'à l'exécution des tâches élémentaires appropriées et la fin de la résolution.**

Le modèle de tâche de PowerTasks que nous avons présenté succinctement a l'avantage de permettre une expression déclarative des stratégies de résolution de problème [10]. Il se démarque en cela des approches adoptées dans les domaines de l'acquisition des connaissances et de la modélisation [5][8], et rejoint l'utilisation des tâches dans les ERP comme outil de pilotage de modules algorithmiques [7][9][23][14]. Cette représentation des connaissances d'un ERP par objets et tâches fait suite à plusieurs modèles de tâches et ERP [1][3][20][25] conçus au sein du projet Sherpa au cours de ces cinq dernières années et qui ont donné lieu à de nombreuses collaborations pour le pilotage de bibliothèques dans des domaines aussi variés que l'analyse de données [2], le traitement du signal, et l'analyse de séquences génomiques [15].

## 3. Environnement de résolution distribuée de problèmes

Les ERP permettent d'utiliser des bibliothèques spécialisées de façon optimale. Par contre l'installation de ces bibliothèques reste un lourd problème en temps et en compétence et nécessite en plus des ressources qui ne sont pas toujours disponibles. Par ailleurs, les sites qui utilisent PowerTasks, créent des bases de connaissances méthodologiques et développent, de ce fait, une compétence propre qu'il est intéressant de mettre à disposition de toute la communauté utilisatrice. Nous avons donc construit un environnement permettant une utilisation distribuée des bases de connaissances méthodologiques et des bibliothèques pilotées. Ce choix nous permet de réduire le coût d'installation et de maintenance des bibliothèques à un minimum d'un seul site serveur. Nous fournissons ainsi l'accès à ces bibliothèques pour des sites ayant des configurations matérielles ou logicielles insuffisantes et nous proposons aux sites clients d'utiliser des bases de connaissances distantes.

### 3.1 Choix d'une architecture

Deux choix d'architecture permettent l'utilisation distribuée de stratégies de résolution :

- un réseau en étoile composé d'une part d'un serveur centralisant bases de connaissances et bibliothèques spécialisées, et d'autre part d'un ensemble de clients disposant de leurs données ;
- un réseau composé de plusieurs sites serveurs avec des compétences diverses. Comme pour l'architecture précédente, l'ensemble des clients disposent de leurs données.

Le choix d'un serveur centralisé engendre des problèmes aussi bien d'installation que de gestion :

- le code des procédures doit être porté sur l'environnement serveur. Ce qui n'est pas vraiment différent de la configuration où chaque site supporte un environnement non distribué. Lors de l'installation du site serveur, on retrouve des problèmes de compatibilité matérielle et logicielle qui peuvent demander une réécriture partielle du code existant ;
- le serveur doit disposer de chacun des logiciels spécifiques, ainsi que des bases de données appropriées à chaque stratégie ;
- le serveur doit être capable de gérer un nombre suffisant de connections et d'apporter à chaque requête une réponse en un temps acceptable ;
- lorsqu'une procédure présente un défaut, il est difficile d'identifier son origine. Il peut provenir d'une erreur de la procédure, de la compatibilité entre l'environnement natif et celui du serveur, ou encore d'un manque de données sur le serveur ;
- la gestion des bases de connaissances demande des mises à jour fréquentes, ce qui représente un lourd travail pour le gestionnaire du site ;
- le gestionnaire du site doit posséder une compétence très étendue sur la totalité des logiciels et stratégies disponibles.

La solution adoptée par PowerTasks est la création d'un système distribué sur plusieurs sites serveurs avec une solution mixte quant à la mobilité du code (cf. section suivante). Chaque site maintient les bibliothèques qui lui sont locales, et les stratégies de résolution qu'il a créées. L'ERDP permet au code de chaque module de rester dans son environnement natif, accompagné des logiciels spécialisés et des données qui lui sont nécessaires. Cela évite la réécriture de code et les problèmes de compatibilité matérielle. Ce choix d'architecture distribué autorise ainsi la coexistence de configuration différente, tant matérielle que logicielle.

Il permet une mise à disposition des différentes stratégies de résolution quelle que soit leur localisation. Chaque site serveur contribue donc, en accord avec son domaine de compétence, à la conception d'une unique base de connaissances méthodologique, multilocalisée. Pour cela, il développe des stratégies mono ou multi localisée et les met à la disposition des autres en les rendant publiques. De même que le WWW permet de voir une multitude de documents distants comme un même hyper-document, le but ici est de voir une multitude de stratégies développées et maintenues sur différents sites serveurs comme une même « hyper-base méthodologique ».

## 3.2 Solutions techniques choisies pour la conception de PowerTasks distribué

PowerTasks distribué est basé sur PowerTasks mono-site (cf. section 2) auquel ont été rajoutées la gestion de la multilocalisation des bases de connaissances, l'utilisation du réseau pour accéder aux bibliothèques distantes.

Les architectures distribuées basées sur plusieurs serveurs, peuvent se comporter, vis-à-vis du code des programmes de bibliothèques, de trois façons différentes :

- soit le code est intransportable pour des raisons d'incompatibilité d'environnements matériels ou logiciels. Les exécutions sont lancées sur le serveur et les résultats rapatriés sur l'environnement client comme dans le cas de RPC (Remote Procedure Call) ;
- soit le code peut migrer à travers le réseau et être utilisé sur une machine distante (mobile). Dans ce cas, le client possède des outils permettant de rapatrier puis l'exécuter en local le code sélectionné. Cette solution permet un nombre important d'accès clients simultanés, sans pour autant être ralentit par l'accès au réseau. C'est le cas d'un langage comme JAVA ;
- soit le code existe en plusieurs versions. Le code peut être utilisé sur différentes plates-formes et s'adapte à différents environnements clients ; les exécutions sont alors, elles aussi, locales. Ces possibilités sont offertes, par exemple, par ActiveX.

La mobilité du code dans l'ERDP PowerTasks dépend de sa nature (tâche/module) :

- le code des tâches est mobile. Le langage des tâches est interprété. Un interpréte est inclus dans PowerTasks et celui-ci pilote la résolution. Les tâches sont rapatriées pour être exécutées sur place, en local ;
- par contre, le code des programmes de bibliothèques encapsulés dans les tâches élémentaires est généralement exécuté sur son site d'origine (cf. § précédent) ;
- les données utilisées par les programmes ou modules des bibliothèques sont déplacées.

Le rapatriement des tâches sur le site utilisateur permet de réaliser localement la classification ou la décomposition du problème et offre ainsi à l'utilisateur la possibilité de conserver les fonctionnalités d'interaction présentes dans le modèle mono-site sans en modifier le temps de réponse. L'utilisateur peut donc intervenir interactivement dans le déroulement d'une résolution (relancer ou archiver une résolution). Ce système donne l'impression aux utilisateurs que les bibliothèques sont bien présentes sur leurs machines mais ce code n'est que virtuel puisque les modules encapsulés dans les tâches élémentaires sont restés sur leur site serveur.

Les derniers choix réalisés concernent la modification de l'environnement par l'ajout de fonctionnalités afin de :

- définir la localisation des compétences accessibles au travers du réseau,
- permettre la communication entre les tâches réparties,
- donner l'accès à toutes les tâches de manière identique, qu'elles soient locales ou distantes,
- élaborer un processus de synchronisation permettant l'interaction entre les tâches réparties sur les différents sites,
- vérifier la cohérence entre les bases de connaissances du domaine et méthodologiques des divers sites.

Ces différents points entraînent une modification du modèle de tâches due à l'adjonction de tâches et de sites externes ainsi qu'à la définition de mécanismes de synchronisation élémentaire qui leur sont nécessaires.

# 4. Modèle de tâches réparties

Le modèle de tâches décrit dans la deuxième partie supporte bien la multi-localisation des tâches (cf. Figure 7). Celle-ci n'a qu'un effet mineur sur le système, puisque les tâches sont rapatriées. Une fois le rapatriement effectué, les résolutions se déroulent localement jusqu'à l'appel des procédures de bibliothèque. Néanmoins, l'évolution vers un langage distribué de tâches nous a conduits, d'une part, à ajouter dans la description des tâches un attribut *statut* afin de les caractériser quant à leur visibilité au travers du réseau, et, d'autre part, à concevoir des protocoles de mise à disposition de tâches.

## 4.1 Statut des tâches

Lorsqu'un site a construit une nouvelle stratégie ou une tâche élémentaire et qu'il souhaite la rendre accessible à d'autres sites, il lui donne le statut *public*. Par contre, il est nécessaire de créer d'autres statuts restreignant les accès aux tâches des bases de connaissances. Un site peut, en effet, désirer développer des tâches qui lui soient propres. Ces tâches ont un statut *privé* et demeureront locales et invisibles. Une alternative existe lorsqu'un site a publié une tâche et souhaite que la stratégie de résolution sous-jacente demeure cachée : il lui donne, alors, le statut *masqué*. Dans ce cas, les utilisateurs bénéficient de la fonctionnalité de la tâche mais pas de la visualisation de la stratégie sous-jacente. Seule la tâche masquée est rapatriée, ses sous-tâches sont exécutées à distance.

## 4.2 Mise à disposition des tâches

Lorsqu'un site donne à une tâche le statut public, il doit se livrer à un processus de publication auprès d'un annuaire

qui centralise chacune des tâches publiques. Il existe aussi des protocoles de modification lui permettant de mettre à jour les attributs d'une tâche déjà publiée.

N'importe quel site peut publier une tâche pour la rendre accessible aux autres. Ce faisant, il devient serveur de cette tâche et de la stratégie sous-jacente vis-à-vis des autres. Il reste client pour les tâches externes qu'il utilisait déjà. Chaque site est donc potentiellement client et serveur. Un site peut même être serveur d'une stratégie dont les tâches élémentaires sont externes, il est alors uniquement serveur de la hiérarchie de tâches.

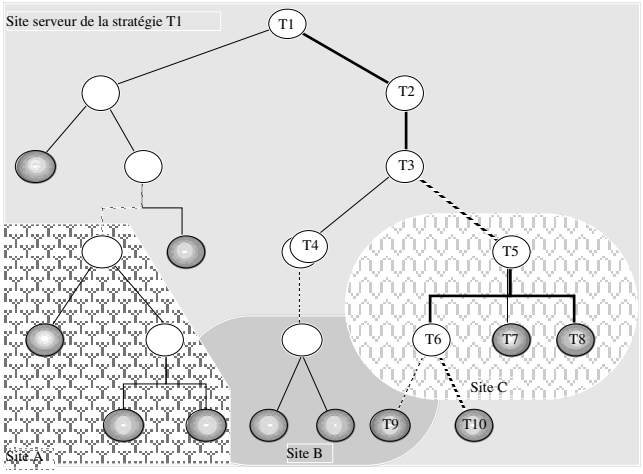



**Figure 7 : exemple de localisation de stratégie. La stratégie T1 fait appel à des stratégies du site A, B et C ainsi qu'à des tâches élémentaires (en gris). Par exemple, lors d'une exécution, la tâche T1 se spécialise en T2 puis T3 ensuite la spécialisation se fait vers une tâche du sîte C (T5). Cette tâche T5 se décompose en trois sous-tâches dont la première (T6) se spécialise en une tâche T10 localisée sur le site serveur de la tâche T1.**

Les bases de connaissance méthodologiques des autres sites ne sont pas vues dans leur intégralité, mais uniquement via les tâches publiques qu'elles contiennent, leurs sous-tâches et les tâches masquées. Il est possible que l'ensemble des bases méthodologiques d'un site soit vide si l'utilisateur ne fait qu'utiliser les stratégies publiques des autres sites.

### 4.2.1 Protocole de publication

Pour publier une tâche, un site doit envoyer à un annuaire le nom de la tâche, et sa signature (type des entrées et des sorties), sa description fonctionnelle et un numéro de version (identificateur de la dernière mise à jour). Cette publication est validée par l'annuaire qui vérifie qu'il n'y a pas de conflit de nom avec des tâches existantes sur le même site. Deux tâches peuvent porter le même nom si elles sont sur deux sites différents. Si elles sont sur le même site, elles doivent avoir la même signature et un numéro de version différent, sinon la nouvelle tâche est refusée. L'annuaire vérifie aussi que la description fonctionnelle est en adéquation avec la signature de la tâche. En effet, la description contient le type des entrées et des sorties pour permettre aux futurs utilisateurs d'intégrer facilement la tâche dans leur stratégie. Une fois ces vérifications effectuées et les conditions respectées, la candidate est ajoutée dans l'annuaire. La tâche devient alors publique et peut être utilisée par les autres sites.

### 4.2.2 Protocole de modifications

Les modifications des tâches publiques et masquées sont possibles. Elles peuvent être complexes à traiter parcequ'elles engendrent des protocoles différents en fonction de leur type (cf. Figure 8). Les types de modification des tâches publiques sont :

- des modifications de signature d'une tâche,
- des modifications de la stratégie sous-jacente à une tâche,
- des modifications du module encapsulé par une tâche élémentaire.

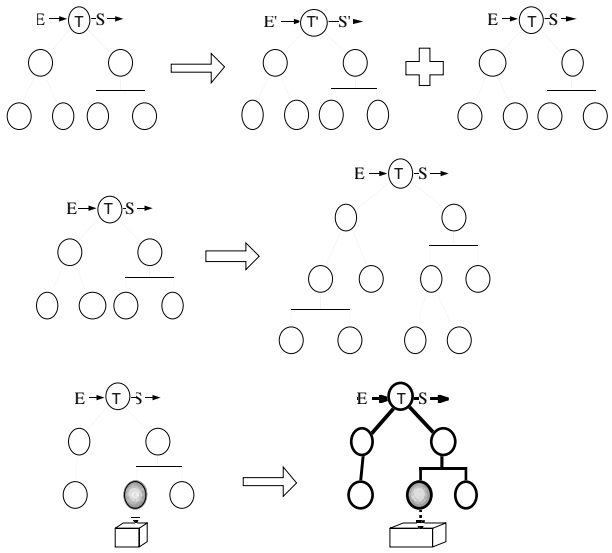



**Figure 8 : Exemple de changement de signature, de stratégie, d'implémentation. Dans le premier cas, la signature de la tâche T doit être modifiée, la tâche T en T' cohabite les deux versions, l'utilisation de la deuxième version est privilégiée pour les nouvelles utilisations, par contre la version précédente est conservée pour satisfaire les utilisateurs antérieurs. Dans le deuxième cas, La décomposition ou la spécialisation d'une stratégie peut être compléter sans que sa signature en soit affectée, dans ce cas seule la dernière version persiste dans la base. Dans le troisième cas, le programme de bibliothèque encapsulé est remis à jour, et si aucune modification de la signature n'a été faite, cette mise à jour est transparente pour l'utilisateur.**

La tâche dont la signature est changée doit prendre un autre nom. En effet, il ne peut y avoir deux tâches de même nom et de signatures différentes sur un même site (cf. section 4.2.1). Cette contrainte est motivée par le fait que le changement de signature d'une tâche, après sa publication, entraînerait des incohérences dans les stratégies utilisant cette tâche, les rendant ainsi inutilisables. Ainsi, le site qui a mis une tâche à disposition d'autres sites doit en assurer la persistance. Il existe donc un « invariant » entre le nom et la signature d'une tâche publique, de même qu'entre son nom et sa fonctionnalité.

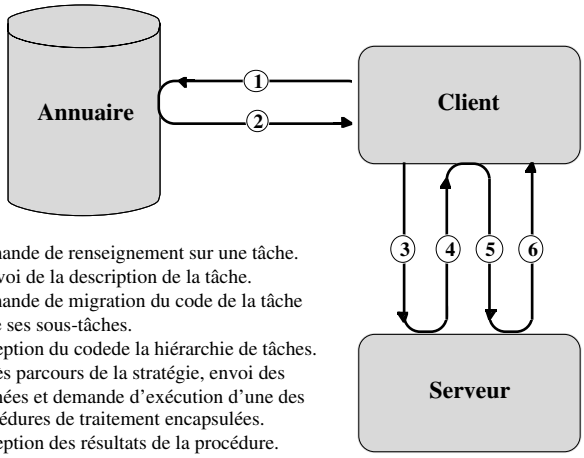
La modification d'une stratégie n'affectant pas la signature de la tâche publique est considérée comme un changement de version de cette tâche. Les stratégies basées sur cette tâche peuvent utiliser l'ancienne version ou la nouvelle. A terme, la version obsolète sera remplacée par la nouvelle. Rendre immédiat les changements de stratégie est déconseillé. En effet, la stratégie de résolution sous-jacente à une tâche contribue à la façon dont l'utilisateur appréhende le domaine. Un changement automatique de

celle-ci perturberait l'utilisateur même si les fonctionnalités de la nouvelle stratégie sont identiques à celles de l'ancienne.

Par contre, le remplacement d'un module de bibliothèque par une autre est, lui, automatique. En effet, lors de la modification d'un module la tâche élémentaire qui l'encapsule ne change pas, mais l'annuaire enregistre un changement de version. Lors des futures résolutions, le nouveau module est exécuté par les clients. S'il se produit une anomalie lors d'une résolution ultérieure, le système permet de savoir s'il y a eu un changement d'implémentation identifiant ainsi une des causes possibles de l'anomalie.

### 4.2.3 Protocole d'utilisation

Les sites clients peuvent consulter l'annuaire et utiliser une tâche qui en fait partie. Ils peuvent utiliser chaque tâche publique comme stratégie à part entière ou comme sous-tâche d'une de leurs propres stratégies. Lors de résolutions faisant appel à des tâches externes, celles-ci sont rapatriées puis exécutées (Figure 9). Le graphe d'exécution de la tâche externe est visualisé sur le grapheur comme dans le cas d'une tâche locale.




**1.** Demande de renseignement sur une tâche.
**2.** Renvoi de la description de la tâche.
**3.** Demande de migration du code de la tâche et de ses sous-tâches.
**4.** Réception du codede la hiérarchie de tâches.
**5**. Après parcours de la stratégie, envoi des données et demande d'exécution d'une des procédures de traitement encapsulées.
**6.** Réception des résultats de la procédure.

**Figure 9 : Description des communications lors d'une résolution utilisant une tâche distante.**

## 5. Architecture de PowerTasks Distribué

Notre ERDP est formé de quatre parties : les bases de connaissances méthodologiques, le logiciel PowerTasks, l'annuaire et les services serveurs (Figure 10).

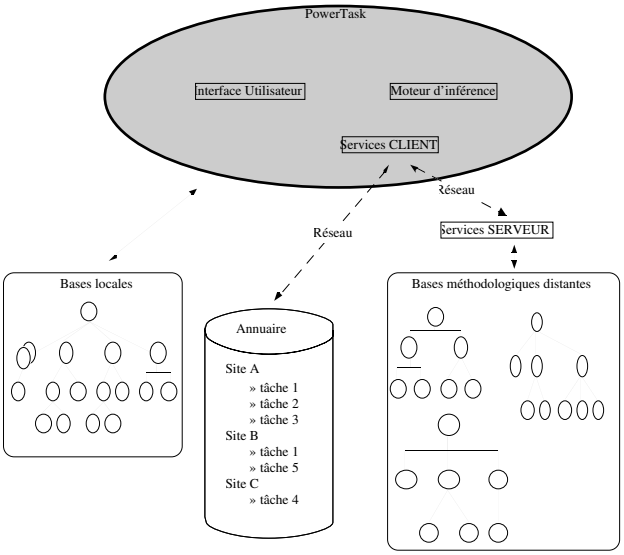



**Figure 10 : Communications entre les bases, l'annuaire, le serveur et PowerTasks (interface, service client et moteur d'inférences). Les communications passant par le réseau sont indiqués par des flèches en pointillé. PowerTasks est localisé sur chaque site utilisateur. Il existe des services serveur sur chaque site contenant des tâches publiques.**

Afin d'obtenir la liste des tâches publiques, les clients consultent un annuaire contenant la description des tâches publiques et des sites qui les ont publiées. Cette description contient la signature de la stratégie et un commentaire décrivant la méthode utilisée.

L'utilisation de bases de connaissances méthodologiques distantes nécessite un recours aux services serveurs du site contenant ces bases. L'utilisation de bases de connaissances méthodologiques locales reste directement pilotée par l'ERDP, comme dans le cas PowerTasks non distribué (cf. Figure 10).

PowerTasks est le cœur du système. Il est formé de l'interface utilisateur, du moteur d'inférence et des services clients (cf. Figure 10). Ces services gèrent les communications avec l'annuaire et les services serveurs des sites dont le système utilise des tâches publiques. Le reste de l'ERDP se comporte selon les pré-requis d'un ERP non distribué (cf. section 1).

Les services serveurs communiquent avec les sites clients qui les interrogent et répondent à leurs requêtes : envoi du code de tâches publiques, lancement de résolutions ou envoi de résultats. Comme on peut le voir sur la Figure 10, ces services forment la liaison entre l'ERDP basé sur un site utilisateur et les bases de connaissances méthodologiques du site serveur.

L'annuaire est interrogé ou amendé par les services clients de l'ERDP (cf. Figure 10). Il est localisé sur plusieurs sites. Il existe un site qui contient la version originale de l'annuaire, c'est le site de référence. Les autres sites dits *miroirs* ne contiennent que des copies de l'annuaire original. Ces sites miroirs ont pour objectif de décharger le site de référence de la consultation des tâches publiques. Ils permettent ainsi de :

- réduire le temps de réponse de l'annuaire aux requêtes des sites,
- augmenter la fiabilité de l'ERDP en cas de panne.

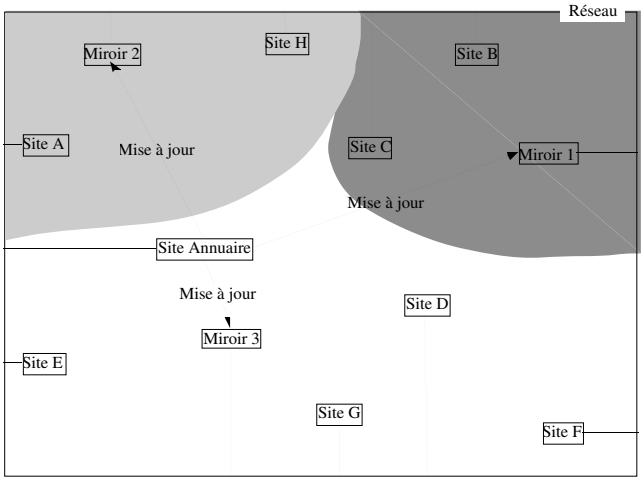



**Figure 11 : Zones d'influence des sites miroirs et rôle central du site de référence.**

Les sites miroirs sont utilisés uniquement pour la consultation des tâches publiques. Chaque site client s'adresse à un site miroir particulier (Figure 11, zones d'influence), le choix est fait en fonction de divers critères (proximité en nombre de noeuds dans le maillage, nombre de connections, etc.). Les requêtes de type consultation formulées par les clients à un site miroir sont satisfaites plus rapidement que dans le cas d'un annuaire mono-localisé puisque de par leur multiplicité, les sites miroirs sont moins chargés que le site de référence puisqu'ils sont uniquement dédiés à ce type de requêtes. La pluralité de ces sites évite aussi les échecs dus aux pannes lors des consultations puisque la connaissance est repliquée sur tous ces sites.

Le protocole de communication réseau est http. Ce protocole a l'avantage de permettre de :

- communiquer avec l'annuaire,
- rapatrier des stratégies de résolution,
- lancer l'exécution des procédures sur les sites serveurs.

## 6. Conclusion et perspectives

PowerTasks est un outil générique de résolution de problèmes conçu autour du modèle à objets développé par la société Ilog et d'un modèle de tâches issu de nos expériences précédentes. Il permet au travers d'une interface utilisateur unique :

- la description de tâches dans un langage approprié autorisant une expression déclarative des stratégies de résolution de problème.
- de piloter l'exécution des tâches, de contrôler et paramètrer l'appel des procédures de bibliothèques,
- de déclencher l'exécution de ces procédures,
- de décrire le problème posé et de visualiser les résultats de la résolution au moyen d'une interface unique,
- d'intervenir directement dans le processus de résolution par un changement de valeur, ou l'ajout, de paramètre, ou un choix de la stratégie de résolution.

Les ERP développés à partir de PowerTasks proposent toutes ces fonctionnalités dans un environnement homogène au moyen d'une interface unique. Celle-ci permet ainsi à l'utilisateur une utilisation simple des bibliothèques et l'accès aux connaissances de résolution sans qu'il ait besoin d'être un expert des techniques qu'il est amené à utiliser.

Les environnements de résolution de problèmes permettent à des experts de travailler sur leur domaine applicatif tout en utilisant des connaissances spécifiques développées par d'autres. PowerTasks autorise en plus l'utilisation de stratégies de résolution et de blibliothèque de programmes qui sont sur d'autres sites. Cet apport facilite la diffusion des compétences. Ainsi des utilisateurs de différents domaines peuvent travailler en partenariat et à distance. Cette possibilité représente un atout majeur dans la résolution de problèmes puisque chaque tâche s'appuie sur la dernière stratégie développée par les experts du domaine concerné et chaque stratégie est construite audessus du code le plus récent et le plus performant. Les stratégies sont écrites avec un code mobile et sont interprétées localement ; ceci permet à l'utilisateur de conserver les mêmes interactions avec le système que dans l'environnement non distribué.

Comme le modèle réparti permet d'avoir plusieurs tâches réalisant la même fonctionnalité, nous envisageons d'introduire des critères de rationalité dans le choix des tâches à exécuter. Ces choix ne se baseraient plus seulement sur les données du problème mais aussi sur la situation géographique de la tâche, sa rapidité [11] ou la qualité des solutions qu'elle fournit. De plus, la possibilité de lancer une partie de la résolution sur un site distant permet de pouvoir paralléliser certaines tâches [6], ce qui pourrait faire appel à des techniques d'intelligence artificielle distribuée de type multi-agent [12]. Cependant nous n'avons pas encore étudié les cas dans lesquels la mise en oeuvre de telles techniques serait appropriée. L'ouverture des environnements de résolution de problèmes vers le réseau engendre donc des possibilités nouvelles comme l'optimisation dans les résolutions.

## Remerciements

Une première maquette de l'environnement distribué a pu être réalisée grâce à B. Angelotti, E. Boisseau, W. Duron et A. Masier dans le cadre d'un stage de DESS en collaboration avec E. Mazeran et P. Guérin de la société Ilog.
Les auteurs remercient G. Bisson, C. Capponi, P. Genoud, J. Gensel et P. Uvietta pour leurs relectures attentives et nos fructueuses discussions.

## Bibliographie

[1] N. Beauboucher, Anaïs: un système de raisonnement à partir de cas dans un environnement de résolution de problèmes, *Actes journées Volcan-IA*, p. 245-260, Clermond-Ferrand (FR), 1993

[2] F. Chevenet, F. Jean-Marie, J. Willamowski, SLOT : a cooperative problem-solving environment in explanatory data analysis, *Actes 49th session of the International Statistical Institute*, p. 255-256, Firenza (IT), Septembre 1993.

[3] M. Chaillot, Une architecture de contrôle réactif pour la résolution coopérative de problèmes, *Thèse de doctorat*, INPG, Grenoble (Fr), 1993.

[4] B. Chandrasekaran, Generic tasks in knowledge based reasoning high level building blocks for expert system design, *IEEE Expert*, 1986.

[5] B. Chandrasakaran, T.R. Johnson et J.W. Smith, Task-structure analysis for knowledge modelling, *Communication of the ACM*, 35(9), p. 124-137, 1992.

[6] F. Charpillet, A. Boyer, Progress : un modèle d'agent pour la conception de systèmes multi-agents temps réel, Journées Francophones IA distribuée et Systèmes Multi-agents, 1997.

[7] V. Clément, Raisonnements cognitifs appliqués au pilotage d'algorithmes de traitement d'images, *Thèse de doctorat*, Université de Nice Sophia-Antipolis, Nice (Fr), 1990.

[8] I. Delouis, LISA, un langage réflexif pour la modélisation du contrôle dans les systèmes à bases de connaissances. Application à la planification de réseaux électriques, *Thèse de doctorat,* Université de Paris Sud centre d'Orsay, Paris (Fr), 1993.

[9] Y. Demazeau, J. Ferber, Actes de la 1ère Journée Nationale sur les Systèmes Multi-Agents, *eds PRC-GDR IA MARCIA CRIN, France, LIFIA Publ.*, Grenoble, France, décembre 1992.

[10] H. Erickson et al, Task modeling with reusable problem solving method, *AI 2(79)*, p. 293-326, 1995.

[11] E. Fink, Statistical Selection Among Problem-Solving Methods, *Research Report CMU-CS-97-101*, 1997

[12] S. Fujita, V.R. Lesser, Centralized Task Distribution in the Presence of Uncertainly and Time Deadlines, *Proceeding of ICAMS*, Japon, 1996.

[13] E. Gallopoulos et al, Future Research Directions In Problem Solving Environnements For computational Science, *CSRD Report*, N°1259, 1992.

[14] KbuP'95, First international workshop on Knowledge-Based systems for the (re) Use of Program Libraries, *INRIA*, Sofia Antipolis, France, 1995.

[15] Médigue C., Vermat T., Bisson G., Viari A., Danchin A., Cooperative computer system for genome sequence analysis, *Proceedings of 3rd ISMB*, Cambridge, Grande-Bretagne, pp249-258, 1995.

[16] I. Moszer, F. Kunst, A. Dachin,The European Bacillus subtilis genome sequencing project : current status and accessibility of the data from a new World Wide Web site, *Microbiology* 142, p. 261-268, 1996.

[17] F. Postel, P. Zimmermann, A review of the ODE solvers of Axiom, Derive, Maple, Mathematica, Macsyma, MuPAD and Reduce, *Proceedings of the 5th RHINE WORKSHOP ON COMPUTER ALGEBRA*, Saint-Louis, États Unis, 1996.

[18] F. Rechenmann, Knowledge bases and computational molecular biology, dans Nicolaas Mars (éd.), *Towards very large knowledge bases (actes 2nd international conference on building and sharing very large-scale knowledge bases (KBKS),* Enschede (NL), IOS press, Amsterdam, Pays Bas, pp7-12, 1995

[19] J. R. Rice, R. F. Boisvert, From Scientific Software Libraries to Problem-Solving Environnements », *IEEE Computational Science & Engineering*, p. 44-53, 1996.

[20] B. Rousseau, Vers un environnement de résolution de problèmes en biométrie*, Thèse de doctorat*, Université C. Bernard, Lyon, France, 1988.

[21] A. Th. Schreider, B.J. Wielinga, R. de Hoog, J.M. Akkermans, W. Van de Velde, CommonKADS: A Comprehensive Methodology for Knowledge Based Systems Development, *IEEE Expert*, 9(6), p. 28-37, 1994.

[22] M. Thonnat, S. Moisan, Knowledge-based systems for program supervision, *KbuP'95, First international workshop on Knowledge-Based systems for the (re) Use of Program Libraries*, Sofia Antipolis, France, 1995.

[23] J. Van Den Elst, Modélisation de connaissances pour le pilotage de programmes de traitement d'images, *Thèse de doctorat*, Université de Nice Sophia Antipolis, Nice, France, 1996.

[24] S. Weerawarana, E.N. Houstis, J.R. Rice, A. Joshi, and C.E. Houstis, PYTHIA: A knowledge based system to select scientific algorithms, ACM Trans. Math. Software, 22, 1996, p. 447-468.

[25] J. Willamowski, Modélisation de tâches pour la résolution de problèmes en coopération système-utilisateur, *Thèse de doctorat*, Université Joseph Fourier, Grenoble, France, 1994.