MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY

UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار- عنابة

Faculté des Sciences de l'Ingéniorat
Département d'Informatique

Année : 2019

THÈSE

Présentée en vue de l'obtention du diplôme de
Doctorat en Sciences en Informatique

UNE METHODE D'ACQUISITION DE CONNAISSANCES DU DOMAINE D'UN SYSTEME DE RAISONNEMENT A PARTIR DE CAS

Option : Intelligence Artificielle

Présentée par
M^me^ Hioual Ouided

Soutenue le 05 Mai 2019

Devant le jury

President	**M^me^** Yamina MOHAMED BEN ALI	**Prof**. Université de Annaba
Directeur de thèse	**M^r^** Mohamed Tayeb LASKRI	**Prof**. Université de Annaba
Examinateur	**M^r^** Mohamed Chawki BABAHENINI	**Prof**. Université de Biskra
Examinateur	**M^r^** Abdelkrim AMIRAT	**Prof**. Université de Souk Ahras

Remerciements

Le grand merci revient encore et toujours à notre DIEU le tout puissant qui m'a donné le courage et la patience et qui a éclairé mon chemin pour achever ce travail. La réalisation de cette thèse est indissociable de son contexte familial, scientifique et matériel. Nombreux sont ceux qui m'ont soutenu, aidé ou supporté. Je suis reconnaissante à tous d'avoir été là quand il le fallait, de s'être investi, d'avoir partagé, d'avoir contribué autant que possible, pour que cette thèse se fasse. Je tiens tout d'abord à remercier :

- Monsieur. LASKRI Mohamed Tayeb, Professeur à l'université Badji Mokhtar de Annaba, mon directeur de thèse qui a tout d'abord accepté de m'encadrer, puis a soutenu, orienté, enrichi, critiqué, bref, dirigé… ma recherche. Je lui dois évidemment aussi beaucoup pour le soutien moral et quotidien qu'il ma apporté, ainsi que la patience qu'il a su montrer. Je tiens à lui exprimer toute ma reconnaissance et gratitude.

- Les membres du jury qui m'ont fait l'honneur de bien vouloir évaluer mon travail, et plus précisément :
Madame Yamina MOHAMED BEN ALI, Professeur à l'Université d'Annaba , pour l'honneur qu'elle m'a fait, en acceptant la présidence de ce jury.

Monsieur Mohamed Chawki BABAHENINI, Professeur à l'Université de Biskra et Monsieur Abdelkrim AMIRAT, Professeur à l'Université de Souk Ahras qui m'ont fait l'honneur d'accepter de faire partie de mon jury en tant qu'examinateurs.

Dédicaces

Merci mon DIEU de m'avoir permis d'arriver jusqu'ici et de m'avoir donné l'aptitude d'achever ce modeste travail que je dédie particulièrement à mes très chers et adorables parents qui m'ont inculqué toutes les bases de mon savoir, que DIEU me les garde.

A la pensé de ma petite adorable chère sœur qui me manque beaucoup khadidja Soulef.

A mes frères et sœurs : Mohamed Saleh, Noureddine, Ghania, Souad et Ouassila.

A Maifi Lyes.

A Hemmam Sofiane , Amouche Ilhem. & Nezzar Sorya

A Mes adorables neveux et nièce : Kamel Ouassim, Louai Eddine , Imen , Nour Serine, Innes, Amir Abderrahmane et Rahma.

A mes très chères ami(e) s : Nouzha, Amina.

Je n'oublierai pas de dédier ce travail à tous ceux ou celles qui ont ne serai-ce qu'un moment partagé ma vie.

Ouided .H

Résumé

Le raisonnement à partir de cas (RàPC) est une technique d'intelligence artificielle (AI) qui consiste à résoudre un problème en se remémorant des cas passés déjà résolus et en adaptant la solution du cas remémoré au nouveau cas à résoudre. Ce processus est soutenu par plusieurs types de connaissances telles que les connaissances du domaine (CD). Ces connaissances ne sont pas suffisantes et il est donc nécessaire d'acquérir de façon continue des nouvelles connaissances pour adapter de nouveaux problèmes.

Cette thèse présente d'une part une approche d'acquisition interactive des connaissances de domaine dans un système de raisonnement à partir de cas. Et pour cela, nous avons utilisé une version modifiée du système FrakaS (un système pour l'acquisition de connaissances du domaine par analyse interactive d'échecs) pour mettre en œuvre l'approche proposée. La modification consiste à utiliser les logiques de description à la place de la logique propositionnelle, ainsi que la sélection d'un opérateur de révision adéquat pour la révision de la base de connaissances. D'autre part, la thèse présente comment capitaliser les connaissances dans une mémoire d'entreprise en utilisant la méthode REX (Retour d'EXpérience), L'objectif de la méthode proposée est de capitaliser les connaissances qui doivent être stratégiques et de favoriser le retour d'expérience. La méthode a été implémentée pour le domaine médical. Nous nous somme intéressés à la capitalisation des connaissances en informatique on utilisant l'approche de gestion des connaissances (Systèmes Experts). Un système expert générique a été implémenté pour être utilisé dans plusieurs domaines. Et nous achevons notre travail de recherche par une application portant sur le problème d'allocation de charge statique dans un environnement informatique distribué sur un ensemble de processeurs hétérogènes, où les tâches sont indépendantes et unitaires.

Mots clés : Raisonnement à partir de cas, adaptation conservatrice, logique de description, théorie de la révision AGM, raisonnement à partir de connaissances du domaine, Système FrakaS, La méthode REX, Système expert, chainage avant, équilibrage de charge statique, algorithme optimal.

Abstract

Case-based reasoning (CBR) is an artificial intelligence (AI) technique using solutions of former experiences to understand and solve new problems. This process is supported by various kind of Knowledge such as the domain knowledge (DK). This knowledge is however not sufficient and therefore it is necessary to continuously acquire additional ones in order to adapt to new problems.

Firstly, this thesis presents an approach of interactive acquisition of domain knowledge in a case based reasoning system, we used a modified version of the Frakas system to implement the proposed approach. The modification consists in using the description logic instead of the propositional logic, as well as the selection of an appropriate revision operator for the revision of the knowledge base. And second, how to capitalize knowledge in a corporate memory using the REX method (feedback), The objective of the proposed method is to capitalize the knowledge that must be strategic and to promote feedback. The method has been implemented for the medical domain. Then we focused on the capitalization of computer knowledge using the knowledge management approach (Expert Systems). A generic expert system has been implemented for use in several domains. And we finished our work with an application dealing with the problem of static load allocation in a distributed computing environment on a set of heterogeneous processors, where the tasks are independent and unitary.

Keywords: Description logic, Case-based reasoning, Conservative adaptation, based reasoning domain knowledge, belief revision theory (AGM), FrakaS system. The REX method, expert system, forward chaining, static load balancing, optimal algorithm.

الملخص

الاستدلال إبتداءا من الحالة هو تقنية من تقنيات الذكاء الاصطناعي الذي يهدف إلى حل مشكلة بالتذكر و التكيف مع الحالات التي سبق حلها , هده العملية مدعمة بواسطـــــة عدة أنواع من المعارف من بينهـــا معارف المجال, هاته المعارف ليست كافية و بالتالي من الضروري الحصول و بشكل مستمر على معارف جديدة للتكيف مع مشاكل جديدة

هاته الأطروحة تقدم أسلوب تفاعلي لاكتساب معارف المجال بنظام التفكير المبنـــــي على الحالات, في هاته الأطروحة استخدمنا نسخة معدلة من نظام فراكاس لتنفيذ هدا الأسلوب. التعديل في هدا النظام يكمن في استخدام المنطق الوصفي بدلا من المنطق الإقتراحي و كذلك اختيار مشغل المراجعة المناسب لأجل مراجعة قاعدة المعارف في المنطق الوصفي.

وثانيًا ، كيفية الاستفادة من المعرفة في ذاكرة الشركة باستخدام طريقة ريكس (عودة التجربة) ، الهدف من الطريقة المقترحة هو الاستفادة من المعرفة التي يجب أن تكون استراتيجية وتعزيز ردود الفعل من الخبرة. تم تنفيذ هذه الطريقة في المجال الطبي. ثم ركزنا على رسملة مهارات الكمبيوتر باستخدام نهج إدارة المعرفة (الأنظمة الخبيرة)، لقد قمنا بتطبيق نظام خبير عام للاستخدام في عدة مجالات.

وانهينا عملنا مع تطبيق يتناول مشكلة توزيع الأحمال الثابتة في بيئة حوسبة موزعة على مجموعة من المعالجات غير المتجانسة ، حيث تكون المهام مستقلة وحدوية.

الكلمات المفتاحية

الاستدلال إبتداءا من الحالة، التكييف مع المحافظـــة، المنطق الوصفـــي، نظرية المحافظة ، الاستدلال إبتداءا من معارف المجال، نظام فراكاس ، طريقة ريكس، نظـــام الخبراء ، تسلسل للأمـــام ، موازنة تحميل ثابتة ، خوارزمية مثالية.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

1. Contexte du travail

Le raisonnement à partir de cas (RàPC) est un outil d'intelligence artificielle, considéré comme l'outil privilégié de modélisation de l'expérience des utilisateurs et d'apprentissage incrémental de ces expériences. Les systèmes de RàPC ont pour fonction de capitaliser l'expertise terrain sous forme de connaissances dans sa mémoire, de pouvoir raisonner dans un domaine à connaissance réduite, de réutiliser des connaissances analogues pour prendre une décision et d'enrichir la mémoire en ajoutant de nouvelles connaissances d'une façon dynamique.

Le principal problème du raisonnement à partir de cas est de savoir comment trouver des expériences pertinentes et d'en faire usage dans une situation donnée. Par exemple la similitude entre les recettes de poisson à l'orange et de crevette au citron peut sembler évidente pour tout être humain avec l'expérience de la cuisine, mais ce n'est pas le cas pour un système informatique. Le système nécessite une quantité considérable de connaissances pour être en mesure de conclure que, dans la préparation d'une recette, il est possible de remplacer les oranges avec des citrons et que, d'un point de vue, poisson et crevette la cuisson peut être considérée comme similaire et donc interchangeable.

Le point critique du paradigme de raisonnement à partir de cas est que malgré l'efficacité du mécanisme de raisonnement, les résultats ne sont pas satisfaisants si la qualité des connaissances sur lesquelles ils se basent est mauvaise.

Les systèmes experts sont apparus au milieu des années 80, et ont connu une grande difficulté dans la construction des bases de règles constituant la connaissance des systèmes. Les faits, cependant, étaient plus faciles à acquérir du fait qu'ils constituent les connaissances factuelles d'un problème à résoudre. Ceci facilite l'acquisition des nouveaux cas pour enrichir la base de cas dans le raisonnement à partir de cas

Un système de raisonnement à partir de cas est un système qui effectue des raisonnements et, par conséquent, qui s'appuie sur des connaissances. Parmi ces connaissances il y a, bien entendu, les

cas sources, mais beaucoup de systèmes utilisent également d'autres sources de connaissances, telles que les «connaissances du domaine» (aussi connues sous les appellations «ontologie du domaine» ou « théorie du domaine »). Plus ces connaissances du domaine sont exactes (correctes?) et précises (complètes?), meilleures seront les inférences effectuées par le système de RàPC.

2. Problématique et objectifs

Dans un système de raisonnement à partir de cas Il ya une grande différence entre les connaissances du domaine, et la connaissance de l'expert, car les connaissances du domaine sont disponibles mais ne sont pas suffisantes, il est donc nécessaire d'en acquérir de nouvelles. Ce problème est impossible à résoudre complètement dans la plupart des applications, mais nous pouvons encore acquérir de nouvelles connaissances de domaine grâce à l'expert.

L'objectif de cette thèse est de présenter dans un premier lieu une approche d'acquisition interactive des connaissances de domaine dans un système de raisonnement à partir de cas. Nous avons utilisé une version modifiée du système FraKas [[A.Cordier ,2009].] pour mettre en œuvre l'approche proposée. La modification consiste à utiliser les logiques de description à la place de la logique propositionnelle, ainsi que la sélection d'un opérateur de révision adéquat pour la révision de la base de connaissances. Nous nous sommes intéressés au cadre où un tel système donne des solutions cohérentes avec les connaissances du domaine.

Dans le raisonnement à partir de cas, l'adaptation est souvent considérée comme une tâche difficile, par rapport à la récupération qui est considérée plus simple à concevoir et à mettre en œuvre. Dans cette thèse, une approche de l'adaptation appelée adaptation conservatrice est proposée. Cette adaptation conservatrice a été choisie parce que d'une part, elle nécessite une certaine connaissance du domaine, considérée comme une caractéristique importante d'une telle adaptation et d'autre part à cause du système FraKas utilisé qui est basé sur cette dernière, ce qui reflète notre contribution principale à savoir l'acquisition des connaissances du domaine.

Cette adaptation est basée :

- Sur la théorie de révision qui consiste à faire des changements minimales sur le cas source afin d'être cohérant avec le cas cible et pour réaliser cette adaptation, le choix s'est porté sur un opérateur de révision pour réviser la base de cas.

- Un opérateur de révision qui doit satisfaire tous les postulats AGM développé par Alchourrón, Gärdenfors et Makinson.

Et en second lieu comment capitaliser les connaissances dans une mémoire d'entreprise en utilisant la méthode REX (Retour d'EXpérience) [Malvache et al, 93].

La mémoire d'entreprise contient des connaissances du domaine de l'entreprise ainsi que différentes sources de connaissances basées sur une ontologie commune. Cette ontologie permet d'utiliser les connaissances stockées dans cette mémoire par les différentes classes d'acteurs et selon leurs différentes tâches à effectuer.

L'objectif de la méthode proposée est de capitaliser les connaissances qui doivent être stratégiques et de favoriser le retour d'expérience. La méthode a été implémentée pour le domaine médical.

Puis nous nous sommes intéressés à la capitalisation des connaissances en informatique en utilisant l'approche de gestion des connaissances (Systèmes Experts).

Un système expert générique a été implémenté pour être utilisé dans plusieurs domaines. Ce système est composé principalement d'une base de connaissances contenant des connaissances factuelles qui sont des faits spécifiques du domaine et des connaissances déductives permettant de spécifier le comportement du système. Nous avons utilisé un moteur d'inférence en chainage avant qui permet d'inférer de nouvelles connaissances à partir de la base de connaissances du système parmi les différents modes d'invocation des règles.

Et nous avons terminé notre travail par une application portant sur le problème d'allocation de charge statique dans un environnement informatique distribué sur un ensemble de processeurs hétérogènes, où les tâches sont indépendantes et unitaires.

3. Organisation de la thèse

Le manuscrit est structuré en quatre chapitres et une conclusion générale.

1- Dans le premier chapitre, nous présentons quelques définitions et les différentes typologies et modèles de connaissance, nous exposons une synthèse sur ces modèles de connaissance, et nous citons par la suite quelques approches de modélisation des

connaissances et nous terminons le chapitre par les formalismes de représentation des connaissances.

2- Les bases sur le raisonnement à partir de cas (RàPC) sont décrites dans le deuxième chapitre. Nous présentons une méthodologie permettant la résolution des problèmes selon le principe général : des problèmes similaires ont des solutions similaires. Dans ce cadre, le RàPC cherche dans sa base de cas un problème similaire et réutilise sa solution pour résoudre le problème rencontré. Ce problème ainsi résolu devient une nouvelle expérience qui peut être stockée dans la base de cas comme nouvelle acquisition de connaissance.

3- Le troisième chapitre est consacré à la présentation du système FraKaS, une approche pour l'acquisition des connaissances de domaine incrémental basé sur l'exploitation des défaillances du raisonnement (à savoir les échecs d'adaptation) d'un système de raisonnement à partir de cas. Cette approche convient aux situations où l'adaptation produit des solutions qui sont compatibles avec les connaissances du domaine disponible, comme c'est le cas avec l'adaptation conservatrice.

4- Dans le quatrième chapitre de ce manuscrit, nous présentons d'abord une approche d'acquisition interactive des connaissances du domaine dans un système de raisonnement à partir de cas. Une version modifiée du système FrakaS est utilisée pour mettre en œuvre cette approche. La modification que nous avons effectuée consiste à utiliser les logiques de description à la place des logiques propositionnelles, ainsi que la sélection d'un opérateur de révision adéquat pour la révision de la base de connaissances dans les logiques de description. Nous présentons ensuite une implémentation de la méthode REX (Retour d'EXpérience) pour la capitalisation des connaissance d'une entreprise qui est une autre méthode pour acquérir des connaissance du domaine. Nous nous intéressons aussi à la capitalisation des connaissances en informatique en utilisant l'approche gestion des connaissances, en proposant un générateur d'un système expert en utilisant le chainage avant et nous achevons ce chapitre par le choix d'un domaine de puissance comme domaine d'application en distribution pour appliquer notre système expert.

5- Nous terminons ce manuscrit par une conclusion générale qui récapitule le travail réalisé et nous proposons quelques travaux en perspective.

Chapitre 01

Connaissances - Généralités

1.1. Introduction

Les connaissances de l'être humain ne sont pas innées. Selon les philosophes Descartes [Descartes, 1637] et Locke [Locke, 1693], « nos pensées seraient le fruit de notre seule expérience ». L'être humain est donc considéré, à la naissance, comme étant vierge de toute connaissance. Connaître demande un effort dont l'être humain se passerait volontiers, mais l'ignorance constitue un handicap majeur pour l'humanité.

Les principaux points abordés dans ce chapitre portent d'abord sur quelques définitions, nous exposons les différentes typologies, le cycle de vie, le modèle de connaissances, suivis d'une synthèse sur ces modèles de connaissances. Nous poursuivons par citer quelques approches de modélisation des connaissances et nous terminons le chapitre par les formalismes de représentation des connaissances et la réutilisation des connaissances.

1.2. Donnée, Information, Connaissance

1.2.1. Définitions

La donnée est un élément de base qui peut être utilisé pour représenter une information dans des bases de données, à savoir une mesure ou une caractéristique. Nous distinguons deux types de données : les données brutes et les données structurées permettant d'être transférées par un système informatique à distance.

L'information est une donnée interprétée qui représente un fait réel. C'est la donnée complétée par une description qui indique le contexte : de quelle mesure s'agit-il, quand, où et par qui elle a été prise, etc.

La connaissance Selon [Alavi et Leidner, 2001], la connaissance est l'information retenue par l'esprit des individus. Il s'agit des informations personnelles liées à des faits, des procédures, des concepts, des idées, des observations et des jugements. La connaissance est alors un processus qui diffère selon les individus. [Nonaka et Takeuchi, 1997] définissent aussi la connaissance comme un processus humain dynamique de justification de croyances personnelles vers l'atteinte de la vérité.

[Davenport et Prusak, 2000] définissent une connaissance comme étant la compréhension d'une information et de son association à d'autres informations et connaissances préalables (expériences, idées et valeurs) pour pouvoir évaluer et incorporer des nouvelles informations.

La compétence est, dans ce cas, une connaissance structurée pouvant être directement exploitée par les utilisateurs afin d'accomplir une certaine tâche ou action. Cela représente « la connaissance dans l'action » donc une résolution d'un problème qui est après cette action validée ou révisée (actualisée) par le retour d'expérience. Pour notre exemple cela serait un remède pour la défaillance de moteur.

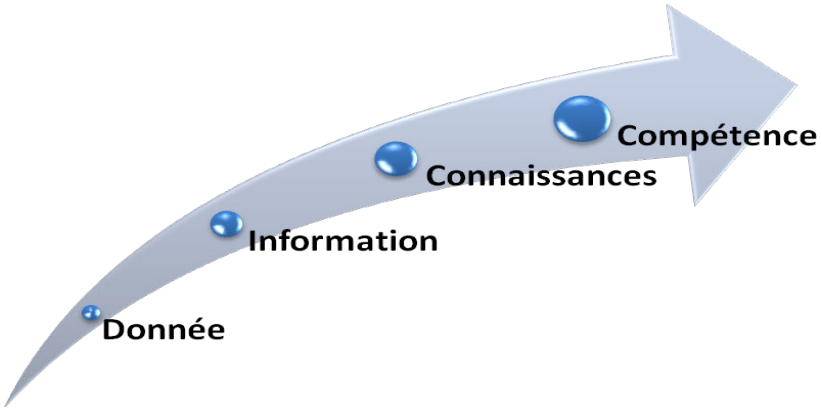


Figure 1.1. Classification des différents types d'information

1.3. Typologies des connaissances

[Nonaka et Takeuchi, 1997] montrent que la connaissance se présente sous deux formes : une forme explicite et une forme tacite. La distinction entre connaissances explicites et connaissances tacites est importante, car, selon qu'elles soient explicites ou tacites, les connaissances ne peuvent être recueillies et transférées de la même façon. Par ailleurs la connaissance peut être déclarative ou procédurale. La distinction entre connaissances déclaratives et procédurales a principalement été développée en psychologie cognitive par Anderson [Anderson,1996].

1.3.1. Connaissance explicite

La connaissance explicite est une connaissance codifiée, exprimable en langage formel et transmissible d'un individu à un autre sous forme de documents réutilisables [Grundstein, 2000]. Cette connaissance est observable, mesurable et de ce fait facilement transférée à travers des méthodes formelles et systémiques [Alavi et Leidner, 1999].

1.3.2. Connaisssance tacite

La connaissance tacite est une connaissance non verbalisée, intuitive et non articulée. Elle est personnelle, difficile à formaliser en parole et souvent liée à un contexte particulier et de ce fait, il

est difficile de la communiquer et de la transférer [Polanyi, 1966] . Les modèles mentaux et les schémas que les humains se forment sur le monde entrent dans cette catégorie de connaissances.

1.3.3. Connaissance déclarative

La connaissance déclarative dite aussi factuelle, propositionnelle ou descriptive est la partie de connaissance qui décrit la réalité des choses. Elle contient les faits, les objets et les principes relatifs à un domaine donné [Lehner et al ,2000]. Les taxonomies sont parmi les plus importantes formes de connaissances déclaratives

1.3.4. Connaissance procédurale

La connaissance procédurale est la partie de connaissance qui répond à la question : « comment réaliser une tâche ? » (les conditions, les sous tâches). Elle correspond aux stratégies d'utilisation des connaissances déclaratives pour la résolution de problèmes. Selon [Lehner et al ,2000]. la connaissance déclarative représente le 'savoir' alors que la connaissance procédurale, associée au savoir représente le 'savoir-faire'.

1.4. La capitalisation des connaissances

1.4.1. Définition

Capitaliser : ce mot peut se résumer par « Savoir d'où l'on vient, savoir où l'on est, pour mieux savoir où l'on va ». Il procède d'une logique « évolutionniste » de la connaissance, à savoir que l'évolution des connaissances (et donc de l'entreprise) se fait toujours à partir d'un patrimoine existant.

[Grundstein, 2000] donne la définition suivante : « Capitaliser les connaissances de l'entreprise c'est considérer les connaissances utilisées et produites par l'entreprise comme un ensemble de richesses constituant un capital, et en tirer des intérêts contribuant à augmenter la valeur de ce capital ».

La capitalisation des connaissances consiste à identifier, documenter et conserver la mémoire des activités qui ont été menées, de telle manière à rendre accessibles, explicites les connaissances afférentes. La capitalisation, au contraire, s'efforce de développer des modèles de formalisation plus structurés et plus élaborés. Il ne s'agit plus de recueillir des connaissances éparses et hétérogènes pour les intégrer dans l'activité mais constituer un modèle global et cohérent de l'ensemble des connaissances attenant à un certain champ.

Capitaliser les connaissances implique la constitution d'un capital intellectuel qui sera ensuite valorisé. On ne peut pas capitaliser toutes les connaissances et donc il est nécessaire de ne considérer que les connaissances stratégiques ou cruciales pour certaines activités. Le processus de capitalisation des connaissances permet de réutiliser, de façon pertinente, les connaissances d'un domaine donné, précédemment stockées et modélisées, afin d'accomplir de nouvelles tâches [Simon, 1996].

La capitalisation de la connaissance est définie dans [Matta et al., 2001] comme « une formalisation d'expérience gagnée dans un domaine spécifique ». Le but est de « localiser et rendre visibles les connaissances de l'entreprise, être capable de les conserver, y accéder et les actualiser, savoir comment les diffuser et mieux les utiliser, les mettre en synergie et les valoriser» [Grundstein, 2000]. Il y a plusieurs méthodologies de capitalisation des connaissances combinant les aspects techniques, humains et d'organisation

1.5. Cycle de vie de la connaissance

Selon Grundstein [Grundstein, 2000], il faut repérer les connaissances, les préserver, les valoriser et les actualiser :

- **Repérer les connaissances cruciales**, c'est-à-dire les connaissances explicites et les connaissances tacites qui constituent le cœur de tout système à base de connaissances. Il faut les identifier, les localiser, les caractériser, en faire des cartographies et les hiérarchiser. Toutes ces actions ont comme objectif commun la création de connaissances.

- Préserver des connaissances : consiste à acquérir les connaissances explicitables, les modéliser, les formaliser et les conserver. Autrement, il faut encourager le transfert de connaissances de type « maître - apprenti » et les réseaux de communication entre les personnes par exemple. L'organisation doit organiser, structurer, et répartir cette connaissance.

D'après Stein et Zwass [Steels, 1993], ce processus est décrit en deux phases :

- (1) une phase d'acquisition et de conservation
- (2) une phase de recherche et de restitution des connaissances.

La première phase s'intéresse à la représentation des connaissances de manière à ce qu'elles puissent être réutilisées par ailleurs par différents membres de l'organisation. C'est une phase particulièrement importante. Elle est décisive pour la réussite ou non des étapes ultérieures. La deuxième phase, de recherche et de restitution, est étroitement liée à la première et notamment aux choix opérés, en ce qui concerne la codification des connaissances.

- Valoriser des connaissances : il faut les rendre accessibles selon certaines règles de diffusion et d'exploitation des connaissances. Ce processus lie la problématique de capitalisation des connaissances à la problématique d'innovation [Nonaka & Konno, 1998] [Grundstein, 1995] .Son objectif principal est la réutilisation des connaissances.

- Actualiser les connaissances : C'est le processus d'évaluation, de la mise à jour et de l'enrichissement des connaissances au fur et à mesure de la création de nouvelles connaissances nouvelles et de l'apport de connaissances externes. Cette phase permet de faire évoluer la base des connaissances afin de la rendre pertinente. La connaissance doit évoluer en s'adaptant aux changements de l'environnement. Cette évolution doit tenir compte de la consistance de la base des connaissances, la suppression, la modification ou l'insertion d'une connaissance doit être validée soit automatiquement par des outils spécifiques soit par des experts du domaine.

1.6. Les différents modèles de connaissances

[Wiig, 1993] a présenté une structure reposant sur trois piliers concernant l'identification des connaissances du domaine, l'évaluation et l'appréciation de ces connaissances relatives à leurs activités, et enfin la gestion des activités relatives à la manipulation des connaissances (basées sur la création, manifestation, utilisation et transfert de cette connaissance).

[Leonard-Burton, 1995] a introduit une architecture comprenant quatre compétences de base qui influencent quatre activités de construction des connaissances. Parmi les compétences de base figurent les systèmes physiques (compétences stockées dans les bases de données, logiciels, etc.), les connaissances et les expériences des employés, les systèmes de gestion (les flux d'informations et leurs échange) et finalement les valeurs et normes de l'organisation. Les quatre activités sur les compétences sont la résolution de problèmes partagée et créative, l'implémentation et l'intégration de nouvelles méthodologies et de nouveaux outils, la création des expérimentations et prototypes, l'importation et l'appropriation des technologies externes.

[Andersen, 1996] présente un modèle qui est constitué de sept processus coopérant sur l'organisation de la connaissance, à savoir la création, l'identification, la collection, l'adaptation, l'organisation, l'application et le partage. Dans ce modèle ni la nature de la connaissance ni la nature des processus individuels ne sont spécifiés.

[Choo, 1996] a présenté un modèle de l'organisation « intelligente ». Cette organisation utilise des informations auxquelles elle donne le sens, créé les connaissances et résout des

problèmes. Ces trois processus sont reliés comme un continuum des activités définissant une organisation qui possède des informations et connaissances afin de réagir intelligemment.

***Le modèle de Van der Spek et Spijkervet introduit* dans [Van der Spek et Spijkervet, 1997]** identifie un cycle comportant quatre phases de gestion des connaissances.

La première phase – la conceptualisation comprend la recherche, la classification et la modélisation de la connaissance.

La deuxième phase – le reflet - comprend l'évaluation de la connaissance selon différents critères.

La troisième phase – l'action - comprend l'amélioration de la connaissance en créant des nouvelles connaissances, en les distribuant et sauvegardant finalement la rétrospection signifiant l'évaluation des résultats, les effets des actions et comparaison des situations avant et après.

[Nonaka et Takeuchi, 1995] ont proposé un modèle identifiant quatre types de « conversion des connaissances » tacites et explicites, par création de connaissance organisationnelle. Dans leur modèle de spirale les connaissances organisationnelles sont crées par socialisation (tacites aux tacites), externalisation (tacites aux explicites), internalisation (explicites aux tacites) et combinaison (explicites aux explicites).

Un modèle de transfert des connaissances, présenté dans [Szulanski, 1996], analyse les difficultés du transfert des connaissances dans une organisation. La structure identifie quatre phases : l'initiation, l'implémentation, l'accélération et l'intégration ainsi que quatre facteurs du transfert des connaissances : caractéristique du transfert (ambiguïté causale), caractéristique de la source de la connaissance (manque de motivation et incrédulité observée), caractéristique du destinataire de la connaissance (manque de motivation, manque de capacité de compréhension, etc.), et caractéristique du contexte (relations, contexte organisationnel, etc.).

[Alavi, 1997] a décrit le processus de la gestion des connaissances dans un service des consultations – KPMG Peat Marwick. Le processus est composé d'une séquence de six phases : l'acquisition, l'indexation, le filtrage, l'enchaînement, la distribution et l'application.

[Demarest, 1997] a proposé quatre processus de production des connaissances, à savoir : la construction (découverte et structuration des connaissances), la dissémination (processus humain et infrastructure technique), l'incorporation des connaissances et leur utilisation (production des valeurs commerciales des clients).

[Taylor, 1997] a décrit un cycle de gestion des connaissances par le développement et l'utilisation des connaissances. Le développement comprend la conceptualisation, la revue,

l'internalisation et le partage des connaissances. L'utilisation se réfère au stockage, à la distribution, à l'application et à la révision des connaissances.

[Beckman, 1996] a introduit une architecture prescriptive contenant neuf pas pour la gestion des connaissances, qui se décline par les verbes suivants : identifier, capturer, sélectionner, stocker, partager, appliquer, créer et vendre.

[Davenport et Prusak, 1997] ont proposé un modèle de gestion des connaissances basé sur quatre phases : la détermination des objectifs, l'acquisition, la distribution et l'utilisation des connaissances.

Le modèle de [Tannembaum et Alliger, 2000] examine quatre aspects de la gestion des connaissances, à savoir leur partage, leur accessibilité, leur assimilation (le fait d'apprendre) et leur application (le fait de les utiliser et faire des décisions). Chacun de ces aspects contribue à l'application des connaissances.

[Rastogi, 2000] affirme que l'organisation doit implémenter un ensemble des opérations afin d'accomplir les objectifs du processus de la gestion des connaissances. Les connaissances doivent être identifiées, tracées, capturées, acquises, stockées, partagées, appliquées afin de générer des nouvelles connaissances comme la dernière étape la plus avancée dans la gestion des connaissances dans une organisation.

[Probst, 2002] regardent le cycle de la gestion des connaissances comme un processus dynamique (comme Nonake et Takeuchi) constitué de deux cycles – l'un externe et l'autre interne. Le cycle externe est composé de la détermination des objectifs visés par ces connaissances et leur évaluation permettant de contrôler la gestion des connaissances. Le cycle interne est composé des six blocs suivants : identification, acquisition, développement, distribution, utilisation et préservation.

[Mark W. McElroy, 2002] a défini une architecture concernant le cycle de vie de la connaissance qui suppose l'existence de la connaissance une fois produite, sa capture, sa codification et son partage. Le modèle divise donc le processus de la création des connaissances en deux grands processus, l'un de production de connaissances et l'autre d'intégration de connaissances.

[Grundstein, 2000] présente un cycle qui permet de distinguer les notions de gestion des connaissances et de capitalisation des connaissances. Les quatre processus : Repérer, Préserver,

Valoriser, Actualiser concernent le cycle de capitalisation. Le dernier processus : Manager, concerne la gestion de ce cycle – donc la gestion des connaissances.

- **Synthèse sur Les différents modèles de connaissances**

Nous pouvons regrouper ces différents modèles selon la spécificité des modèles.

Le premier groupe des modèles traite des « architectures générales » du fait qu'ils décrivent en général le processus de la gestion des connaissances et les connaissances utilisées. Dans ce cas, nous parlons d'un cycle de vie de la connaissance. Il s'agit de la plupart des modèles présentés, à savoir de ceux de Wiig, Leonard-Burton, Andersen, Choo, Van des Spek et Spijkervet, Davenport et Prusak, Tannenbaum et Alliger, Rastogi, Probst, McElroy et Grundstein. Les modèles diffèrent dans leurs objectifs, les ressources des connaissances utilisées, les activités de la manipulation des connaissances ainsi que dans leur influence sur la conduite de la gestion des connaissances [Holsapple & Joshi, 1999].

Les modèles de Alavi [Alavi, 1997] et de Beckman [Beckman, 1996] proposent quand même un cycle de vie de la connaissance. Ces cycles sont plus spécifiques du fait que Alavi décrit un processus de la gestion des connaissances dans une organisation particulière et Beckman présente une structure prescriptive et non descriptive comme les autres.

Nonaka et Takeuchi [Nonaka et Takeuchi, 1995] ainsi que Szulanski [Szulanski, 1996]s'intéressent dans leurs modèles au transfert des connaissances dans une organisation. Demarest [Demarest, 1997] et Taylor [Taylor, 1997],de leur côté, s'intéressent aux aspects de la production des connaissances dans l'organisation.

Deux modèles montrent une dissociation entre le cycle de la gestion des connaissances et le cycle de la capitalisation. Il s'agit de modèle proposé par Probst [Probst, 2002] dans les travaux anglophones et le modèle proposé par Grundstein [Grundstein, 2000] dans les travaux français.

1.7. Réutilisation des connaissances

La notion de réutilisation des connaissances est très importante. Elle peut être considérée à plusieurs niveaux. On peut souhaiter réutiliser un ensemble de connaissances préexistantes dans la mémoire de projet lors de la conception d'une nouvelle application. On peut également vouloir visualiser une partie des connaissances pour une tâche donnée ou encore dans le cadre d'un "concept métier" particulier, ce qui peut s'avérer très utile dans le cadre d'une conception collaborative par exemple. En effet, la réutilisation d'une sous partie d'un ensemble de

connaissances permet de diminuer le niveau de complexité du travail effectué en éliminant un ensemble de détails non pertinents.

La visualisation des connaissances va de pair avec la notion de granularité. Il n'est pas toujours nécessaire de visualiser un ensemble de connaissances en affichant tous les éléments jusqu'au plus petit niveau de détail. On peut vouloir rester à un niveau macroscopique. Cet aspect est à prendre réellement en compte [Demourioux 98].

Pour réutiliser les connaissances, il est nécessaire de délimiter les contours de "blocs de connaissances" qui deviennent des entités réutilisables. Cette phase du processus est particulièrement complexe. Comment placer correctement les limites pour s'assurer de disposer des connaissances nécessaires et suffisantes pour le problème courant sans pour autant tout sélectionner ?

La notion de point de vue sur les connaissances semble pouvoir apporter une réponse à cette question. Les points de vue révèlent un sous ensemble des connaissances éclairées sous un angle particulier.

1.8. Quelques approches de modélisation des connaissances

Les approches de modélisation des connaissances que nous présentons dans cette partie sont principalement tirés de [Dieng et al., 1999].

1.8.1. Approche orientée documents non informatisés

Il s'agit ici de fournir des synthèses de documents basées sur des connaissances non explicites, contenues par exemple dans les rapports techniques. MEREX en est un exemple développé par RENAULT [Corbel, 1997], notamment pour formaliser des expériences positives et négatives.

1.8.2. Approche orientée base documentaire

Elle vise à favoriser l'accès à une masse de documents susceptibles de contenir des connaissances. Elle se différencie de la précédente en ce que les sources de connaissances sont ici informatisées. Les techniques d'indexation [Corbel, 1997] sont en général préconisées pour faciliter l'accès aux documents contenant ces connaissances, mais pas nécessairement aux connaissances elles-mêmes : l'exploitation du contenu relève d'une toute autre approche, par exemple celles concernant les travaux réalisés dans le cadre du Traitement Automatique du Langage (TAL).

1.8.3. Approche basée sur le Raisonnement à Partir de Cas (RàPC)

La connaissance est ici indirectement codifiée, au moyen de problèmes résolus dans des contextes donnés dont on pense qu'ils sont transposables ; certains auteurs interprètent d'ailleurs le RàPC comme un moyen de codification de connaissances implicites. Cette approche vise principalement à éviter l'éparpillement des connaissances en se focalisant sur des cas d'expertises typiques passés, et à permettre une actualisation du système de connaissances par l'ajout de nouveaux cas [Simon, 1996].

1.8.4. Approche basée sur des systèmes distribués de connaissances

Cette approche est fondée sur le constat que des groupes d'individus, d'organisations, etc., autonomes et hétérogènes, peuvent avoir des intérêts communs, comme par exemple dans le cadre des entreprises virtuelles ou étendues. Un objectif est de favoriser une meilleure collaboration ainsi que le partage de certaines connaissances. Un exemple représentatif d'application concerne les systèmes multi agents.

1.8.5. Approche orientée Systèmes à Base de Connaissances (SBC)

L'approche SBC est basée sur le constat que certaines connaissances cruciales pour une organisation sont tacites et doivent par conséquent être élicitées puis structurées sous forme de modèles. La méthode GAMETH [Grundstein et Rosenthal-Sabroux, 2004] en est un exemple pour repérer les connaissances cruciales, et CommonKADS (voir par exemple [Breuker et Van de Velde, 1994] ou [Schreiber et al., 1994]) un autre exemple, dédié à la construction de modèles.

1.9. Formalismes de représentation des connaissances

Le formalisme de représentation de la connaissance donne la structure de cette connaissance telle qu'elle est manipulée par le système à base de connaissances. La représentation définit une série de symboles et une série d'opérations sur ces symboles qui modélisent le raisonnement que la représentation supporte. Les symboles ont un contenu sémantique mais les opérations qu'on leur associe sont déterminées par leur syntaxe.

Le formalisme de représentation des connaissances est donc un langage formel donné par l'interprétation sémantique de sa structure syntaxique [Biébow & Szulman, 2000]. Différentes techniques de représentation des connaissances ont été proposées dans le domaine de

l'intelligence artificielle telles que les prédicats logiques, les systèmes à base de règles, les réseaux sémantiques, les schémas (frames) et la représentation orientée objet.

1.9.1. Formalisme logique

La logique est l'un des premiers formalismes de représentation des connaissances en intelligence artificielle proposés par [Kaufman, 1987]. Différents langages logiques ont été proposés comme la logique des propositions (définie par un alphabet, des règles de construction des phrases et un calcul de valeurs de vérité pour ces phrases appelées formules), la logique des prédicats du premier ordre (son alphabet inclut des symboles de fonctions), la logique floue (introduit degré dans la valeur de vérité d'une formule), modale (introduit des modalités telles que la possibilité ou nécessité, passé ou futur), linéaire (propositions jouent le rôle de ressources consommables), etc.

Avantages: Pour la logique du premier ordre, il a été prouvé qu'il existe des ensembles de règles (théorèmes et règles d'inférence) qui sont à la fois corrects et complets. Ce résultat est la base de la puissance de la logique comme formalisme de représentation des connaissances. D'autre part, les arguments en faveur de la logique sont les suivants :

- La logique dispose de solides bases théoriques, pouvant s'étendre à de nouvelles utilisations du fait du développement actuel de logiques dites non-classiques telles que les logiques non-monotones, multivaluées, modales .
- Le naturel avec lequel les éléments de connaissance peuvent être exprimés sous forme de formules logiques ; on peut exprimer un fait sans se soucier de ses manipulations.
- La flexibilité et la modularité dues à l'indépendance des assertions résultant du traitement déclaratif de la connaissance qui facilite la modification et l'élargissement d'une base logique de connaissances.

Inconvénients :

- Un des intérêts de la logique est que la cohérence et la complétude sont garanties mais cette caractéristique peut être, dans certains cas, un inconvénient car la logique est une représentation très formelle et mathématique. Elle ne permet pas, par exemple, une manipulation aisée des informations incertaines ou incomplètes.
- Aucune possibilité n'est offerte pour le contrôle, c'est à dire pour le choix des éléments de connaissance adéquats à employer dans une situation donnée. Le système doit le plus

souvent travailler de façon exhaustive et manipuler un grand nombre d'hypothèses pour trouver celles utilisables. Contre la logique subsiste ainsi toujours l'argument de l'inefficacité due à l'explosion combinatoire. Si la base de connaissances devient trop importante, il n'est plus possible de l'utiliser de façon performante.

1.9.2. Systèmes de production

L'élément de base de ces systèmes est la règle de production qui a la forme suivante : SI <condition> ALORS <action>. La partie condition est exprimée par un prédicat logique qui correspond à une affirmation de la base de connaissances qui doit être vraie au moment de validation de la règle. Dans ce cas l'action, qui est la partie exécutable de la règle et exprime des ajouts ou modifications à faire, peut être déclenchée. Ce système comporte trois parties : une base de règles (la connaissance opératoire de l'expert), un contexte ou base de faits (l'état actuel du système) et un moteur d'inférence (contrôle les actions couplées à un module expliquant le raisonnement) [Rousseau, 1988].

Avantages:

Les règles de production permettent de représenter la connaissance sous forme de petits modules indépendants. Cette modularité, facilite l'ajout et la modification des éléments de connaissance. De plus, la connaissance sur le domaine est séparée de son mode de manipulation et peut être donnée en vrac.

- L'uniformité de la représentation facilite la compréhension de la connaissance stockée.

- Le naturel avec lequel les règles condition-action peuvent être utilisées par le spécialiste pour exprimer son savoir fait que les systèmes de production sont très employés. Le raisonnement humain se formalise assez facilement sous forme de règles de type antécédent-conséquence. En effet, des études faites en psychologie ont montré que l'homme exprime habituellement sa connaissance en disant ce qu'il faut faire dans une situation donnée .

- Avec ce formalisme, il est possible de gérer des informations incertaines. C'est typiquement le cas de MYCIN qui autorise ce type d'informations par l'introduction de coefficient de certitude pondérant la véracité de chaque règle. De plus, de tels systèmes peuvent raisonner avec incertitude.

- Les systèmes de production apportent des facilités pour fournir des explications. Cela est dû à la fois aux règles et à la structure de contrôle.

Inconvénients :

- La représentation sous forme de règles de production ne s'adapte pas à tous les domaines d'applications.
- Elle répond très mal au besoin de représenter les éléments de base du domaine d'application. - L'inefficacité d'exécution, due à la modularité et à l'uniformité de la représentation, principalement en ce qui concerne les séquences d'actions, peut devenir importante d'autant plus si le nombre de règles est grand.
- Enfin il est difficile de suivre le flux de contrôle du système. Ceci est dû au fait que les règles ne peuvent pas s'invoquer directement mais sont contraintes de communiquer à travers la mémoire de travail.

1.9.3. Réseaux sémantiques

Les réseaux sémantiques ont été développés par Quillian [Quillian, 68] à partir de travaux faits sur la modélisation en psychologie de la mémorisation associative des êtres humains.

Un réseau sémantique est un graphe composé d'un ensemble de nœuds qui représentent des concepts d'entité, attribut, objet, événement, état, etc. et d'un ensemble d'arcs orientés, étiquetés, qui relient deux nœuds en représentant les relations binaires entre ces concepts. Nous pouvons structurer la connaissance dans une hiérarchie de concepts par les relations sorte-de et est-un qui représente le lien de spécialisation de concepts. Un certain nombre de modes de raisonnement est mis en œuvre, comme l'inférence par héritage ainsi que le filtrage [Capponi, 1995]. Le formalisme de réseaux sémantiques est le premier à structurer la connaissance et à rendre visibles différentes relations existantes entre les objets, ainsi que la notion de distance entre deux concepts.

Avantages:

- Les réseaux sémantiques sont faciles à lire, à comprendre du fait de l'organisation des connaissances permise par le mécanisme d'héritage. De plus, l'emploi du "Pattem-Maching" comme mécanisme de raisonnement apporte une facilité d'utilisation.
- Ils sont capables de gérer correctement les exceptions.

Inconvénients :

- Il manque une sémantique formelle et une terminologie standard aux réseaux sémantiques. Certains types de connaissances (procédurales ou quantifiées) ne peuvent pas être représentés. Par exemple, ce modèle est incapable de traiter la quantification mis en jeu dans la phrase : "tout parent aime ses propres enfants".
- Du fait de la simplicité structurelle des nœuds, un réseau sémantique est complexe dès qu'il contient beaucoup d'informations. Il est ainsi d'une manipulation délicate et difficile à étendre ou à modifier. La modularité apparente n'est pas un réel avantage.
- Il n'existe pas encore de procédures qui manipulent rapidement ces réseaux.
- Enfin, les systèmes qui utilisent ce formalisme ont des difficultés pour fournir des explications sur leur raisonnement.

1.9.4. Graphes conceptuels

Le formalisme des Graphes Conceptuels (GC), introduit par Sowa [Sowa, 1984], tient ses origines des réseaux sémantiques [Quillian, 1968].

Sowa [Sowa, 1984] a enrichi le concept des réseaux sémantiques par l'aspect du concept s'appuyant sur l'étude de la perception en psychologie. Le sens d'un concept se réduit à sa position relative par rapport aux autres concepts dans le réseau sémantique modélisant les connaissances générales du système. A la différence des réseaux sémantiques représentant plusieurs propositions et définitions de concepts dans un même réseau, les graphes conceptuels organisent les différents types d'éléments dans des structures différentes avec la hiérarchie de spécialisation des graphes. Cela permet de manipuler et de vérifier chaque structure par les règles de cohérence.

Avantages:

- C'est un très puissant formalisme de représentation de la connaissance :
  - doté d'une représentation graphique.
  - disposant d'une sémantique formelle (équivalence avec la logique).
- Nombreux travaux en cours (sur les opérateurs, introduction de notions de vraisemblance, de précision, …) .
- Très efficace en analyse du langage naturel.

Inconvénients :

- Problèmes de disjonction.
- Problèmes de négation.
- Problème d'imbrication de quantificateurs.

1.9.5. Schémas (frames)

Dans les années 60, Bartlett [Bartlett, 1964] a présenté l'idée des paquets de connaissances que l'homme utilise dans son activité cognitive quotidienne pour représenter ses expériences précédentes pour interpréter une nouvelle connaissance. A la fin des années 70 cette idée est développée par [Schank, 1973] dans les modèles de représentation du langage naturel et Minsky [Minsky, 1975] dans le premier formalisme informatique. La théorie développée par Minsky présente la notion « frame » et signifie une structure dynamique remémorée qui doit être adaptée pour correspondre à la réalité d'une nouvelle situation rencontrée. Il s'agit donc de la représentation de connaissance pour servir de support de raisonnement sur le monde réel.

Avantages:

- Très intéressant et pratique pour exprimer la connaissance.
- Utilisé pour le développement d'ontologies.

Inconvénients :

- Pas d'équivalence avec la logique (ex : quantifieurs, disjonctions, …).
- Pas d'expression de connaissances incertaines, imprécises, hypothétiques.
- Certaines inférences sont favorisés, d'autres sont difficiles à réaliser.

1.9.6. Représentation objet

La représentation par objet permet de décrire la connaissance sous forme d'objets eux mêmes décrits par leurs caractéristiques. Un objet est une description d'un élément qui est constitué de deux parties : statique et dynamique. La partie statique concerne l'état de cet élément et l'ensemble de ces propriétés (des données et d'informations). La partie dynamique concerne son comportement (les méthodes ou procédures qui manipulent les données et informations).
Les objets collaborent entre eux de deux manières [Capponi, 1995]: plusieurs objets communiquent à l'aide d'envoi des messages, un objet fait référence à un autre objet au niveau de

la description de son état. Les objets similaires dans leur caractérisation et comportement sont regroupés dans les classes ordonnées par la relation de spécialisation permettant de construire une hiérarchie des classes. Cette hiérarchie permet le mécanisme d'héritage qui consiste à récupérer pour une classe donnée les caractéristiques (l'état et les propriétés) et les méthodes définis dans sa surclasse. Il existe différentes relations d'héritages plus ou moins spécialisées. Une classe permet également de créer des instances : représentations des objets concrets ou réels (instanciation), c'est-à-dire elle fournit à l'objet son interface et sa structure.

1.10. Conclusion

Dans ce chapitre nous avons présenté relativement en détail le concept de connaissance, ses différents types, ainsi que les approches de sa modélisation, sa réutilisation et le formalisme de sa représentation. Dans le chapitre qui suit, nous présentons le système de raisonnement à partir de cas dont les connaissances jouent un rôle crucial dans ce type de raisonnement.

Chapitre 02

Le Raisonnement à partir de cas

2.1. Introduction

Le raisonnement à partir de cas (RàPC) est une méthode d'apprentissage et de résolution de problèmes basée sur les expériences passées. Aujourd'hui cette technique s'apparente de plus en plus à une méthode de gestion des connaissances. Toutefois, cette méthode est généralement utilisée quand on a une faible connaissance du domaine étudié, que les problèmes ne sont pas complètement compris et l'expérience représente la plus grande partie des connaissances sur le sujet traité. Lors de la modélisation de la connaissance du domaine, les experts interviewés, loin de maîtriser leur domaine, ne s'expriment qu'en citant des exemples issus de leurs expériences. Les expériences identifiées sont représentées par des cas.

Un système informatique de RàPC met en oeuvre une base de cas composée d'expériences passées dans laquelle peuvent être recherchées des expériences similaires au problème à résoudre. Les recherches actuelles dans ce domaine se focalisent sur une représentation approfondie des connaissances. En effet, on associe aux cas un ensemble de connaissances, qui permet de dire, d'après certains auteurs, que le RàPC est une technique d'implémentation des systèmes à base de connaissances [Althoff, 2001]. Cette méthode est également bien adaptée aux problèmes d'acquisition de nouvelles connaissances. Nous décrirons cette approche de résolution de problèmes dans ce chapitre.

2.2. Fondations de raisonnement à partir de cas

2.2.1. Historiques et aspects cognitifs

En 1977, Schank et Abelson, les chercheurs en science cognitive, ont été particulièrement intéressés par la mémoire humaine et sa partie dans la compréhension et l'explication de situations [Schank et Abelson, 1977], [Schank, 1982]. Ils ont montré la partie cruciale jouée en se rappelant l'expérience passée dans le processus d'apprentissage et de résolution de problèmes, À ce moment-là ils ont développé la notion de scénarios (scripts). Les scénarios nous permettent de structurer la mémoire conceptuelle (le souvenir). Elles sont des structures de connaissances permettant l'organisation de l'information épisodique (dans des «cas»).

Les scénarios viennent d'expériences et sont développés par l'exposition répétée à des situations similaires, comme «aller à un restaurant ». Selon leur approche, les humains auraient un grand nombre de scénarios disponibles dans leur mémoire. Il serait alors assez facile de se rappeler un scénario pertinent et l'appliquer pour réagir dans une situation donnée.

La multiplicité et la large diversité des scénarios sont difficiles à organiser comme une mémoire structurée. Schank propose ainsi des MOPs (Memory Organization Package): des Paquets d'Organisation de Mémoire) qui permettent l'organisation des scénarios dans des paquets de stockage et les trouver plus facilement. Il est donc possible de généraliser et spécialiser les scénarios et ainsi les organiser hiérarchiquement via les MOPs. Les scénarios existant dans MOPs peuvent être rappelés et adaptés pour être réutilisés dans n'importe quelle situation. La notion d'adaptation est très importante, puisqu'il est rare de trouver un scénario qui adapte parfaitement une situation donnée. L'adaptation nous permet ensuite d'effectuer les rajustements nécessaires aux scénarios, prenant en compte la spécificité de la situation actuelle.

Schank observe que les explications jouent un rôle central dans l'apprentissage et qu'il est souvent plus facile de construire une explication en modifiant une explication existante que d'en construire un à partir de zéro. Ceci est la raison pour laquelle il ajoute des modèles d'explication aux modèles MOPs : XPs (modèles d'explication). Ce modèle propose une structure de connaissance explicite, utilisée pour générer, indexer et tester les explications en conjonction avec la mémoire épisodique [Schanck, 1986], [Schanck et al, 1994].

Les recherches initiées par Schank ont été suivies par de nombreuses études. Le modèle MOP en particulier, a été utilisé dans le système de SWALE [Kass et al., 1986]. Ce système permet la génération d'explications à partir de descriptions de situations passées (les descriptions sont appelées cas). Dans la SWALE, MOPs sont utilisés pour guider la compréhension de phénomènes et utiliser les principes de raisonnement à partir de cas pour trouver des expériences passées quand un échec se produit dans l'explication.

Grâce à l'expérience rappelée, l'explication est alors adaptée à la nouvelle situation. Le premier système informatique qui a été qualifié de «système de raisonnement à partir de cas » est appelé CYRUS et a été développé par Kolodner à l'Université de Yale [Kolodner, 1993].
Le modèle de mémoire de cas développé dans CYRUS a été inspiré du modèle de mémoire dynamique de Schank et a servi d'une base pour d'autres systèmes de raisonnement à partir de cas célèbres tels que CHEF [Hammond, 1986] et CASEY [Koton, 1988]. Les premières études traitant explicitement le raisonnement à partir de cas ont ensuite été présentées aux États-Unis à la conférence DARPA en 1988 [Kolodner, 1988]. Depuis lors, ce paradigme a mûri. Les principes fondamentaux ont été définis plus clairement et des nombreuses applications industrielles et expérimentales ont démontré son utilité. Cependant, beaucoup de perspectives

pour la recherche sont toujours ouvertes, en ce qui concerne des implications dans des sciences cognitives et aussi des applications possibles dans l'intelligence artificielle.

2.2.2. Raisonnement à partir de cas et l'intelligence Artificielle

Le raisonnement à partir de cas (RàPC) est un paradigme de l'intelligence artificielle, comme les réseaux de neurones, les algorithmes génétiques, les systèmes multi-agents, réseaux bayésiens, etc. Le RàPC est souvent considéré comme une sorte de raisonnement par analogie. Ce dernier est exploité dans un certain nombre de domaines de l'intelligence artificielle [Gick et Holyoak 1980], [Gentner et al, 2001]. Il est basé sur des notions subtiles telles que la similitude, l'importance relative de certaines données et l'influence des données du problème sur la solution [Py, 1994], ce sont des notions fondamentales que l'on trouve dans le raisonnement à partir de cas également. Dans le raisonnement analogique, des procédés tels que l'appariement ou la récupération considérés comme des processus cognitifs généraux appliqués à des représentations mentales. En revanche, les systèmes de RàPC sont créés dans le but de résoudre une tâche spécifique dans un domaine particulier. Ainsi le RàPC diffère du raisonnement analogique parce qu'il est appliqué à un domaine particulier, tout raisonnement analogique considère les transferts entre plusieurs domaines. Ainsi l'un des actifs du RàPC est sa capacité à mobiliser des connaissances spécifiques au domaine d'application dans le but d'améliorer l'ensemble de l'efficacité du processus de raisonnement.

Un aspect qui distingue le RàPC d'autres méthodologies du domaine de l'intelligence artificielle est le fait qu'il est basé sur la réutilisation des expériences: les solutions sont réutilisées plutôt que construites à partir des connaissances théoriques. Sur ce point, on peut considérer que ce type de raisonnement est plus proche de la façon dont les humains pensent parfois dans la vie réelle. Nous essayons de trouver une solution existante avant de construire l'un des nôtres (via le raisonnement démonstratif, par exemple). Le raisonnement à partir de cas est particulièrement bien adapté à la résolution des problèmes ouverts pour lesquels la théorie de domaine associé est faible ou difficile à formaliser; des cas antérieurs apportent des connaissances spécifiques à la solution du problème. Le RàPC est également un processus de raisonnement progressif , évolutif et incrémental car il acquiert l'expérience chaque fois qu'un nouveau problème est résolu. Contrairement à d'autres systèmes qui, dans le but de résoudre un problème, doit passer par une vaste zone de recherche, les systèmes de raisonnement à partir de cas sont capables de raisonner avec un petit nombre d'expériences.

Au début des années 90, le RàPC est apparu comme une alternative intéressante aux systèmes à base de règles qui commençaient à montrer leurs limites. Dans les systèmes à base de règles, la maintenance de la cohérence de la base devient plus difficile quand de nouvelles règles sont ajoutées. Le RàPC adopte une approche contrastée, l'ajout de nouveaux cas à la base constitue l'un des principes mêmes de ce raisonnement; l'ajout de connaissances donc ne porte pas le risque de mettre en danger le système. A cette époque, des études comme [Slade, 1991] ont montré comment le paradigme de recherche RàPC pourrait apporter des réponses à certaines des limites des approches actuelles de l'intelligence artificielle. Depuis lors, le raisonnement par cas a été utilisé pour diverses tâches (planification, la synthèse, le diagnostic, la prise de décision, etc.) dans des domaines aussi variés que la cuisine, la médecine, le droit, la sécurité routière, la construction mécanique, la gestion des risques, l'écologie, la prévision météorologique, etc.

2.3. Principes fondamentaux du raisonnement à partir de cas

2.3.1. Définition du cas

Un cas est une connaissance qui représente une expérience. Cette expérience permettant au système de RàPC de résoudre des problèmes de différentes natures. Un cas est une expérience passée permettant au système de résoudre des problèmes plus efficacement ou d'éviter les échecs passés : un cas doit donc être utile pour le raisonnement.

- Selon Fuchs [Fuchs et al, 2006], un cas représente la description informatique d'un épisode de résolution de problème.
- Selon Mille [Mille, 2006], la définition d'un cas (dans la base de cas) passe par trois étapes :

  . La première étape concerne « la synthèse » qui consiste à trouver une structure permettant de satisfaire des spécifications.

  . La deuxième étape concerne « l'analyse » qui, à partir d'une structure particulière, consiste à trouver le comportement associé.

  . La troisième étape concerne « l'évaluation » qui consiste à vérifier que le comportement est conforme à ce qui est attendu.

Un cas est constitué de la paire (une partie problème, une partie solution). Chaque partie étant composée d'un ensemble de descripteurs, la plupart du temps définis dans une ontologie dédiée. On peut distinguer deux types de cas : *cas source* et *cas cible*. Le cas source est celui dans lequel

les parties « problème » et « solution » sont renseignées. Donc, c'est un cas dont on va s'inspirer pour résoudre un nouveau problème, un nouveau problème est appelé un cas cible. Les cas sont comparés dans la base de leurs parties problèmes : la description de la partie problème d'un cas cible est comparée à celle de la partie problème de chacun des cas sources disponibles dans la base de cas. Les différences observées entre les descripteurs de problèmes sont alors exploitées par des opérateurs d'adaptation chargés de l'adaptation des descripteurs de solutions. L'adaptation de la solution réalisée pendant cette phase produit une solution candidate pour le cas cible.

Pour organiser les cas dans la base de cas et afin de faciliter la remémoration des cas similaires, on est amené à les indexer. Les méthodes d'indexation peuvent être manuelles ou automatisées. Le choix manuel des indices de cas suppose que l'objectif d'utilisation des cas et surtout des circonstances dans lesquelles les cas seront utiles soit déterminé précisément. Néanmoins les méthodes d'indexation sont de plus en plus automatisées et nous pouvons citer trois méthodes :

- Les méthodes d'indexation basées sur la similarité produisent l'ensemble des indices d'un cas abstrait issu des caractéristiques communes des cas ressemblants.
- L'indexation basée sur l'explication détermine les caractéristique appropriées pour chaque cas. Cette méthode analyse les cas afin de trouver les attributs prédictifs pouvant être utilisés comme des indexes.
- Les méthodes basées sur l'apprentissage inductive qui choisissent les attributs pouvant jouer le rôle des indexes.

Les cas sources (des cas déjà résolus dans le passé) sont stockés dans une base appelée base de cas. Les cas sont mémorisés et organisés en fonction de critères bien déterminés permettant de les retrouver efficacement [Fuchs et al, 1999]. De plus, l'étape d'acquisition d'un nouveau cas permet de faire évoluer la connaissance (apprentissage incrémental). En fonction de la représentation du cas nous distinguons deux organisations principales de la base de cas :

- La mémoire plate où les cas sont organisés de manière linéaire. Cette organisation est la plus simple et les cas sont stockés dans une liste séquentielle.
- La mémoire hiérarchique où les cas sont structurés et organisés suivant une hiérarchie donnée.

L'organisation de la mémoire est très importante car le type d'organisation influencera les étapes du RàPC et notamment les étapes de recherche et d'apprentissage qui exploitent directement la

mémoire. Un équilibre doit être trouvé entre les méthodes de stockage de cas qui préservent la richesse sémantique des cas et de leurs indexes et les méthodes qui simplifient l'accès et la récupération des cas appropriés [Mille et al., 1999].

2.3.2. Carré d'analogie

Le raisonnement de RàPC peut également être présenté par le carré d'analogie introduit dans [Mille et al., 1999]. Ce carré d'analogie est explicité dans la figure qui suit (cf. figure 2.1.) :

- Dans un premier lieu le lien entre la description d'un cas et sa solution (la trace du raisonnement menant à la solution).
- Dans un deuxième lieu, les liens entre la description et la solution du cas source de la base de cas et du cas cible représentant un nouveau problème à résoudre (similarité entre deux problèmes). Dans ce cas nous parlons d'adaptation de la solution du cas cible en fonction de la similarité et de l'adaptation des descripteurs des cas sources similaires de la base de cas au cas cible.

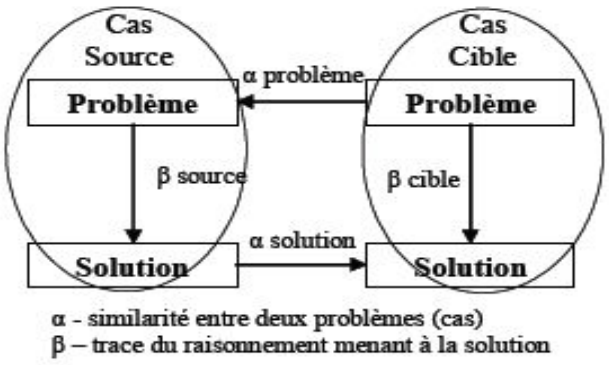


Figure 2.1. Carré d'analogie tiré de [Mille et al, 1999]

Un cas source est sélectionné et dit similaire à partir des valeurs de descripteurs du problème cible et en utilisant la mesure de similarité d. Ensuite les descripteurs solution qui doivent être adaptés car dépendant de descripteurs problèmes source, différents des descripteurs problèmes

cible, sont mis en évidence par les relations de dépendance entre valeurs de descripteurs problème cible et valeurs de descripteurs solution.

En d'autres termes, si une valeur de descripteur source dépend d'une valeur de descripteur problème, une modification de la valeur du descripteur problème entraînera une modification «analogue» à la dépendance du descripteur solution correspondant. Cette connaissance est nécessaire pour l'adaptation. En fonction de ces dépendances et des écarts d constatés à corriger, l'adaptation permet de proposer une solution cible candidate qui pourra être vérifiée par rapport à sa conformité aux dépendances particulières qui pourraient exister entre problème et solution cible.

2.4. Le cycle de raisonnement à partir de cas

L'objectif du raisonnement à partir de cas est de résoudre des problèmes courants à partir d'expériences passées. La méthode remémore les situations similaires passées et réutilise des informations et connaissances sur ces situations. Un système informatique de RàPC est un système particulier de gestion des connaissances. Il met en œuvre une base de cas composée d'expériences passées dans laquelle peuvent être recherchées des expériences similaires au problème à résoudre (problème courant).

L'utilisation d'un tel système répond à un cycle de raisonnement, le cycle du RàPC. Selon les sources bibliographiques, ce cycle est décomposé en 3, 4 ou 5 phases. [Fuchs et al., 2006] distingue les étapes de remémoration, adaptation et mémorisation. [Aamodt, 1994] propose de considérer également une phase de « révision » entre « l'adaptation » et la « mémorisation » d'un cas. [Mille et al , 1999] complète ce processus par une étape en amont "d'élaboration" du cas cible (problème courant) (cf. figure 2.2.)

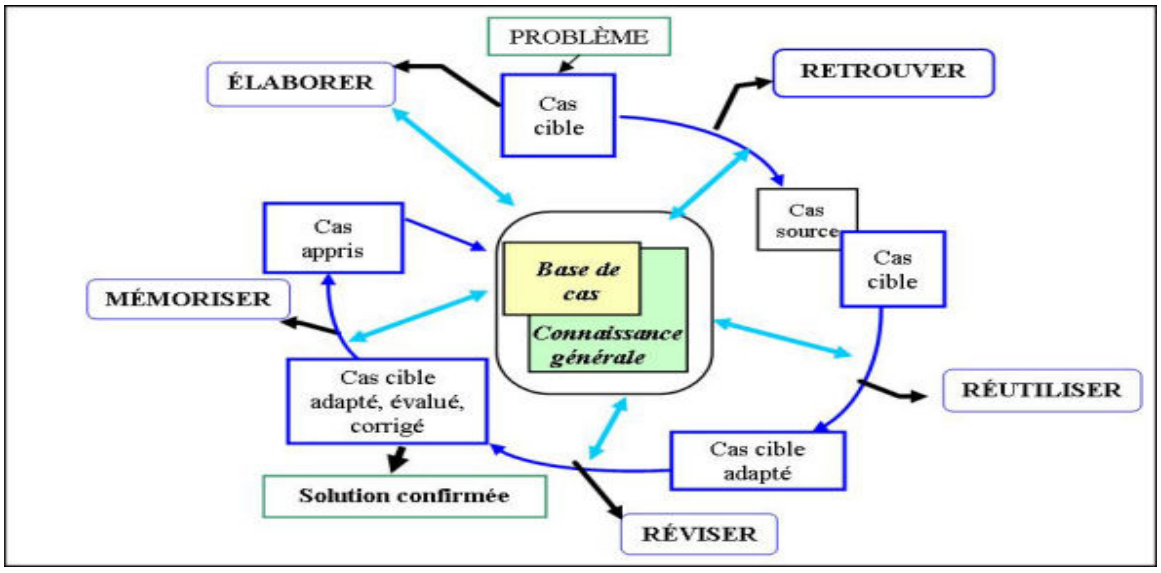


Figure 2.2. Le cycle de raisonnement à partir de cas tiré de [source, Mille 2006]

2.4.1. Élaboration.

Les informations nécessaires à la formulation d'un problème sont collectés et structurées de façon à constituer un nouveau cas : le cas cible. Lors de cette étape, le système sollicite l'utilisateur ou l'environnement extérieur (bases de données, systèmes d'information) pour obtenir l'ensemble des informations nécessaires à la poursuite du raisonnement.

2.4.2. Remémoration.

L'étape de remémoration consiste à chercher dans la base de cas un ou plusieurs cas passés résolus jugés similaires au cas cible. La sélection des cas similaires s'appuie sur une mesure de similarité. Certains systèmes conservent plusieurs cas remémorés pour les combiner par la suite, mais la plupart du temps, un seul cas est conservé pour la suite du processus. On l'appelle le cas source. La sélection du cas source parmi les cas remémorés peut être la conséquence d'une sélection par le système ou bien la résultante d'un choix effectué par l'utilisateur.

2.4.3. Réutilisation.

Cette étape permet d'obtenir une solution au cas cible à partir de la solution du cas source sélectionné qui est tout d'abord recopiée puis éventuellement adaptée afin de satisfaire les

contraintes du problème posé. L'adaptation s'appuie sur des connaissances d'adaptation qui peuvent prendre différentes formes selon les systèmes.

2.4.4. Révision.

Il est possible que la solution proposée par le système ne convienne pas à l'utilisateur ou bien qu'une fois mise en application, elle s'avère inapte à résoudre le problème posé. L'utilisateur a donc, pendant la phase de révision, l'opportunité de modifier, corriger ou même refuser la solution proposée. L'étape de révision permet d'identifier les causes éventuelles des échecs et de proposer des adaptations supplémentaires pour conduire à une solution satisfaisante : il s'agit du cas révisé. Cette étape est à la base du processus d'apprentissage permettant à la fois d'améliorer des connaissances d'adaptations existantes et d'en faire émerger de nouvelles.

2.4.5. Mémorisation.

Traditionnellement, on considère l'étape de mémorisation comme l'étape durant laquelle la base de cas est enrichie par le cas cible révisé. Cette mémorisation implique une mise à jour des index permettant de retrouver les cas et parfois un processus de maintenance pour réorganiser la base de cas. Mais l'étape de mémorisation est également le siège de l'apprentissage d'autres types de connaissances. En effet, c'est durant cette étape que l'on peut concrétiser les efforts effectués durant les autres étapes pour apprendre d'autres connaissances.

2.5. Connaissances utilisées par le RàPC

Un système de raisonnement à partir de cas est un système qui effectue des raisonnements et, par conséquent, qui s'appuie sur des connaissances. Parmi ces connaissances il y a, bien entendu, les cas sources, mais beaucoup de systèmes utilisent également d'autres sources de connaissances.

En s'appuyant sur [Richter, 1995], on identifie quatre principaux types de connaissances utilisées dans différentes étapes du RàPC : la base de cas, les connaissances du domaine, les connaissances d'adaptation et la similarité entre les cas.

2.5.1. Base de cas

En plus des connaissances apportées par les cas individuellement, la base de cas peut contenir d'autres connaissances à travers son organisation. Les cas peuvent être hiérarchisés par niveau

d'abstraction [Branting et al, 2001], un cas particulier traitant d'un problème précis peut être abstrait en un cas moins précis mais couvrant davantage de problèmes. Cette structure est utile pour la remémoration : on cherche d'abord un cas abstrait proche du problème cible puis on affine la solution en cherchant parmi les cas concrets couverts par le cas abstrait retenu. Les cas abstraits permettent aussi d'exprimer des connaissances complémentaires aux cas (mesure de similarité et connaissances guidant l'adaptation) à un niveau plus général [Bergmann et Wilke, 1998].

Les cas peuvent aussi être organisés de façon modulaire, un cas complexe pouvant être décomposé en plusieurs sous-cas apportant chacun une solution à une partie du problème [Smyth et McKenna., 1998]. La réutilisation de plusieurs solutions partielles fait appel à la combinaison de cas : il faut adapter ces solutions partielles au cas cible mais aussi les rendre compatibles entre elles.

2.5.2. Connaissances du domaine (CD)

Elles établissent des propriétés générales du domaine, indépendamment des situations particulières. Typiquement, elles constituent une ontologie du domaine reliant les concepts représentés. Ces connaissances sont complémentaires aux cas, elles complètent leur description en impliquant des connaissances supplémentaires les concernant.

Les connaissances du domaine guident aussi la résolution d'un cas source en établissant des contraintes que doivent vérifier les solutions candidates. Par exemple, si on demande une recette végétarienne, les solutions proposées ne devront pas contenir de viande, et il faudra adapter les recettes remémorées si elles en contiennent. Dans le cadre de l'adaptation, on qualifie ces connaissances de statiques car elles contraignent le résultat de l'adaptation, indépendamment du changement apporté au cas source.

2.5.3. Connaissances d'adaptation

Les connaissances d'adaptation traitent des modifications à apporter à un cas source. On qualifie ces connaissances de connaissances dynamiques car elles portent sur les variations entre cas et guident les modifications à apporter au cas source pour obtenir une solution au cas cible.

En l'absence de méthode exacte de résolution, ce sont ces connaissances d'adaptation qui permettent d'obtenir des solutions candidates au problème posé. Ces connaissances sont plus ou moins fiables dans le sens où il n'est pas certain que la solution candidate obtenue après leur

application à une solution d'un cas source soit satisfaisante. La notion d'effort d'adaptation représente la confiance accordée aux modifications indiquées par une connaissance d'adaptation.

2.5.4. Similarité entre cas

L'évaluation de la similarité entre cas est utilisée lors de la remémoration. Le cas source remémoré est celui qui est le plus similaire au cas cible. Lorsqu'elle est suivie par l'adaptation, le but de la remémoration est de fournir le cas source qui pourra le mieux être adapté, c'est-à-dire pour lequel l'adaptation fournira la solution candidate ayant le plus de chances d'être satisfaisante. La similarité doit donc refléter la capacité d'adaptation des cas sources vis-à-vis du cas cible. C'est le principe de la remémoration guidée par l'adaptation [Smyth et Keane, 1998]. La similarité entre cas est donc déduite de l'effort d'adaptation associée aux connaissances d'adaptation. Pour des raisons d'efficacité, la similarité peut n'être qu'une approximation de la capacité d'adaptation, et par exemple n'être exprimée qu'au niveau des problèmes.

2.6. Modèles de RàPC

Il existe plusieurs modèles pour le raisonnement à base de cas. Ces modèles sont regroupés en trois grandes familles : structurelle, conversationnelle et textuelle.

2.6.1. Le modèle structurel

Le modèle structurel a émergé des premières vagues applicatives de systèmes RàPC. Dans ce modèle, toutes les caractéristiques importantes pour décrire un cas sont déterminées à l'avance par le concepteur du système. Ainsi, le concepteur élabore un modèle de données du domaine applicatif. Tels qu'illustrés à la figure 2.3., les cas sont complètement structurés et sont représentés par des paires (similaire à un "frame" ou à un objet). D'un point de vue applicatif, un attribut représente une caractéristique importante du domaine d'application. Les échelles de valeurs les plus fréquemment utilisées pour structurer les attributs sont les entiers/réels, les booléens et les symboles. La représentation des cas peut être sur un seul niveau ou sur plusieurs niveaux (hiérarchie d'attributs).

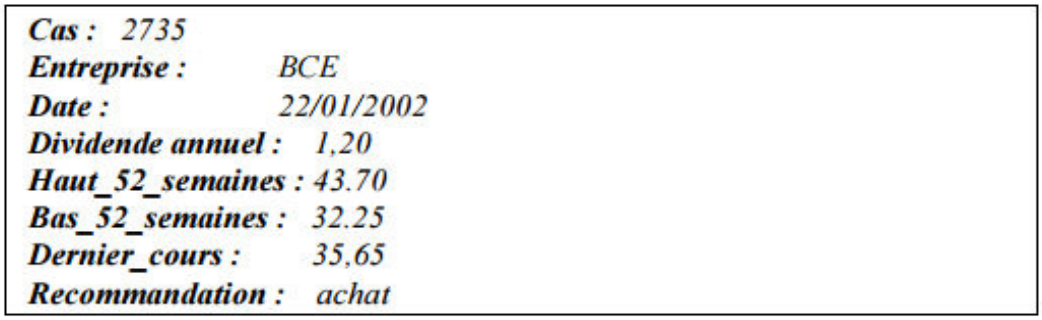


Figure 2.3. Exemple de structuration d'un cas en RàPC structurel tiré de [Luc lamontagne et al, 2002]

La similarité entre deux cas est mesurée en fonction de la distance entre les valeurs de mêmes attributs. Cette distance est fréquemment estimée par les mesures euclidiennes et de Hamming. La similarité globale entre deux cas est habituellement évaluée par une somme pondérée de la similarité de chacun des attributs. Comme les attributs d'un cas n'ont pas tous la même importance et que cette importance varie d'une situation à l'autre, un poids est attribué à chaque attribut de chaque cas. Ces poids permettent de pondérer la similarité globale entre deux cas en accordant un "vote" plus important aux attributs les plus méritants.

Tous les travaux sur l'adaptation de cas sont menés dans le cadre du modèle structurel. L'adaptation peut varier d'une simple substitution de la valeur d'un attribut jusqu'à la restructuration complète d'une solution. Leake [Leake, 1996] identifie environ dix techniques permettant de générer des solutions par substitution, transformation partielle ou dérivation complète. Ces techniques sont habituellement mises en œuvre par des systèmes à base de règles, ce qui nous ramène aux problèmes d'acquisition de connaissance et d'absence de principes généraux pour certains domaines. Pour en limiter les difficultés, certaines approches évitent l'adaptation en sélectionnant, durant la phase de recherche, des cas qui nécessiteront peu d'adaptation [Smyth, 1996].

2.6.2. Modèle conversationnel

Dans l'approche traditionnelle (le modèle structurel), un problème doit être complètement décrit avant que ne débute la recherche dans la base de cas. Cette exigence présuppose une expertise du domaine d'application permettant de bien caractériser une situation à l'aide de

valeurs numériques ou symboliques de sélectionner les principaux facteurs pouvant influencer la résolution de son problème. Toutefois pour certains domaines comme le service à la clientèle, ces aspects sont difficiles à déterminer à l'avance, surtout pour les usagers novices de systèmes RàPC. Le modèle conversationnel a donc été proposé par « Inference Corporation » pour surmonter ces difficultés. Il est actuellement le modèle le plus répandu parmi les applications commerciales du RàPC.

Comme son nom l'indique, le modèle RàPC conversationnel mis sur l'interaction entre l'usager et le système (d'où la notion de "conversation") pour définir progressivement le problème à résoudre et pour sélectionner les solutions les plus appropriées [AHA, 2001]. Un cas conversationnel consiste en trois parties (voir Figure 2.4.) :

- *Un problème P* : une brève description textuelle, habituellement de quelques lignes, de la nature du problème exprimée.
- *Une série de questions et de réponses* Q_A : des index, exprimés sous forme de questions, permettant d'obtenir plus d'information sur la description du problème. Chaque question a un poids représentant son importance par rapport au cas.
- *Une action A* : une description textuelle de la solution à mettre en œuvre pour ce problème. Cette description n'est pas structurée ("free-text").

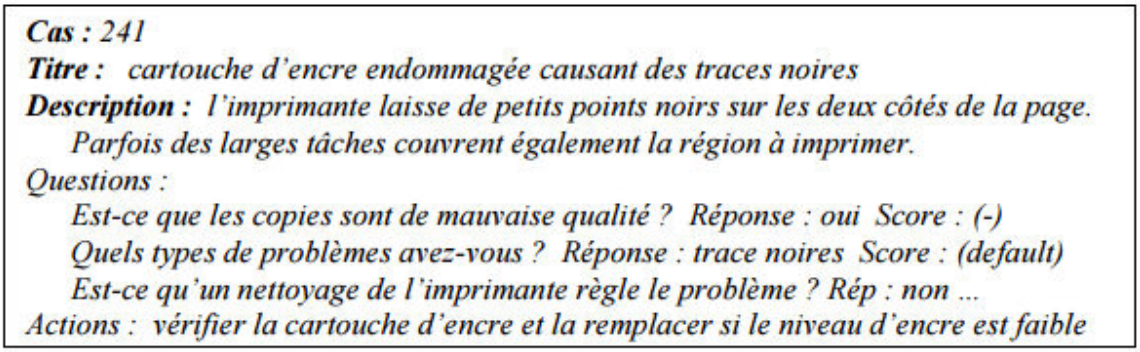

Cas : *241*
Titre : *cartouche d'encre endommagée causant des traces noires*
Description : *l'imprimante laisse de petits points noirs sur les deux côtés de la page. Parfois des larges tâches couvrent également la région à imprimer.*
Questions :
Est-ce que les copies sont de mauvaise qualité ? Réponse : oui Score : (-)
Quels types de problèmes avez-vous ? Réponse : trace noires Score : (default)
Est-ce qu'un nettoyage de l'imprimante règle le problème ? Rép : non ...
Actions : vérifier la cartouche d'encre et la remplacer si le niveau d'encre est faible

Figure 2.4. Exemple de cas pour le modèle conversationnel tiré de [Luc lamontagne et al, 2002]

Cette représentation de cas est donc une extension du modèle structurel avec des attributs de trois types bien précis : description, questions, actions. La notion de trait est étendue à la notion de question afin de pouvoir interroger l'usager. Dans le schéma de résolution du RàPC conversationnel, l'interaction entre le système et l'usager se fait comme suit :

- l'usager fournit au système une brève description textuelle du problème à résoudre et le système calcule la similarité entre cette description et la section "problème" des cas. Le système propose alors à l'usager une série de questions.
- l'usager choisit les questions auxquelles il souhaite répondre. Pour chaque réponse fournie par l'usager, le système réévalue la similarité de chacun des cas. Les questions n'ayant pas reçu de réponse sont présentées par ordre décroissant de priorité.
- lorsqu'un des cas atteint un niveau de similarité suffisamment élevé (i.e. qu'il franchit un seuil), le système propose ce cas comme solution. Si aucun cas n'atteint un degré de similarité suffisant et que le système n'a plus de questions à poser à l'usager, le problème est stocké comme étant non-résolu.

Les systèmes RàPC conversationnels n'effectuent pas d'adaptation des solutions passées. Une des raisons est que la portion 'solutions' des cas n'est pas structurée ("free-text"), ce qui rend difficile la formulation de connaissances d'adaptation. Également, il semble que, pour les applications de type "help-desk", les solutions sont relativement faciles à modifier, même par un préposé inexpérimenté. De plus, l'investissement en temps et en efforts consacrés à développer un système d'inférence qui modifie les solutions est difficile à justifier dans ce contexte opérationnel.

2.6.3. Modèle textuel

Les premiers travaux sur le modèle textuel datent du milieu des années 90 et à ce jour, aucune représentation standard n'est apparue pour ce type de modèle. La description exacte du cas est primordiale dans ces modèles. Par conséquent, la représentation textuelle des cas joue un rôle très important dans la résolution de problème. Par exemple, obtenir le texte d'un jugement légal servant de jurisprudence à une nouvelle cause [Lamontagne et al, 2002]. La différence de ces modèles par rapport aux modèles structurels réside dans la représentation du texte.

Les cas dans les modèles textuels sont semi-structurés ou non-structurés :

- Les cas non-structurés sont les cas qui disposent que d'un seul attribut dont la description est complètement en texte libre (free-text) ;
- Les cas semi-structurés sont les cas qui disposent de plusieurs attributs étiquetés contenant du texte.

Les modèles textuels s'appuient sur la résolution de problèmes à partir d'expériences se trouvant dans des documents textuels d'origine humaine. Deux axes sont identifiés dans les travaux portant sur ces modèles à savoir : structuration de cas textuels et extension du modèle de recherche d'information.

Le premier axe traite les cas textuels. Il a pour objectif de les structurer du mieux possible afin que l'exploitation des méthodes développées soit profitable pour les systèmes de RàPC textuels. Dans cet axe de recherche, les textes sont représentés par un nombre limité de traits qui sont basés sur les caractéristiques du domaine (concepts, catégories, sujets, mots-clés, etc.). Un trait est un attribut ou une caractéristique déterminant la description de problèmes et de solutions du domaine [Lamontagne et al, 2002].

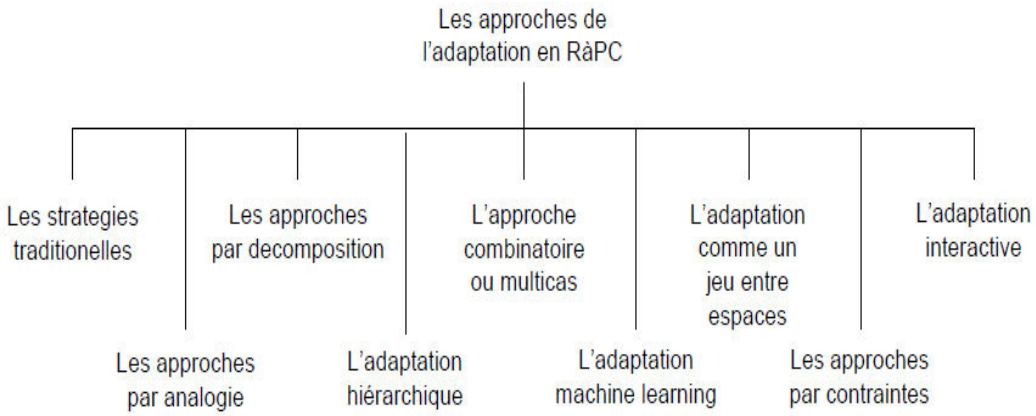
Quant au deuxième axe, une particularité est accordée à la phase de recherche des cas. Les mesures de similarité appliquées lors de cette phase sont définies soit sémantiquement soit par des extensions au modèle vectoriel de recherche d'information . De ce fait, des mécanismes de recherche des cas plus sophistiqués sont élaborés mais tout en gardant un processus d'indexation le plus simple possible. Cet axe de recherche donne la possibilité d'avoir une indépendance pour les applications génériques par rapport au domaine d'application. L'exemple d'une application concrète est représenté dans le projet FAQ Finder (**F**requently-**A**sked **Q**uestions) [Burke et al. 1997]. FAQ Finder est un système de questions réponses basé sur les foires aux questions de USENET. Un FAQ est un cas qui contient la description d'un problème sous forme de questions et la description d'une solution sous forme de réponses. Le système contient plus de 600 fichiers FAQ, ce qui nous amène à quelques dizaines de milliers de FAQs individuels. Les modèles textuels ont donc mené vers les travaux en RàPC Textuel (RàPCT). Un état de l'art sur le RàPC textuel a été abordé dans [Lamontagne et al, 2002]. On peut citer également les travaux de Bentebibel [Bentebibel, 2008] qui a mis en place un système de RàPCT pour la sécurité routière « SAARA » et qui exploite non seulement le modèle textuel mais également le modèle structurel et conversationnel.

La différence entre les trois modèles présentés réside dans la structuration du cas et de son traitement durant les phases du cycle du RàPC. En effet, dans le modèle conversationnel, il n'y a aucun traitement textuel lors de l'interaction entre l'utilisateur et le système. Il se limite à une comparaison de mots clés ou des séquences de « n » caractères découpées d'un texte (*n-grammes*). Le texte est uniquement utilisé pour rendre les questions plus compréhensibles à

l'utilisateur. Quant au modèle structurel, les valeurs des attributs des cas sont des chaînes de caractères sans syntaxe ni sémantique. De plus, contrairement aux modèles textuels, les attributs du cas sont complètement structurés.

2.7. Le processus d'adaptation en raisonnement à partir de cas

L'adaptation en RàPC peut être définie, comme le fait de modifier la solution d'un cas remémoré afin qu'elle puisse répondre aux exigences du problème cible. Plusieurs approches ont été proposées dans la littérature pour effectuer l'adaptation. En effet l'adaptation peut se servir de diverses techniques. Lieber dans [Lieber, 1997] a réalisé un vaste état de l'art dans le but de recenser ces différentes stratégies d'adaptation et leurs différents domaines d'application. La figure 2.5. présente une taxonomie de ces différentes approches.



Source : adaptée de [BAD 09B]

Figure 2.5. Une taxonomie des différentes approches d'adaptation tiré de [BAD, 09B]

2.7.1. Les stratégies traditionnelles d'adaptation

Une classification générale de l'adaptation proposée par [PAL S. K. et SHIU S. C. K , 2004] distingue trois stratégies majeures :

- *La ré-instanciation* est la forme la plus simple d'adaptation, dans laquelle *Sol (source)* est simplement copiée à partir des cas recherchés et appliquée directement, sans modification. Dans [Lieber , 2006] une étude est réalisée portant sur six types d'adaptation par copie : le raisonnement approximatif ou incertain, l'exploitation d'une équivalence entre

représentation de problèmes, l'exploitation d'une relation de généralité entre problèmes, l'exploitation de symétries, le principe de précaution et l'exploitation de dépendances nulles.

- *La substitution* remplace les éléments de la solution qui sont inacceptables parce qu'ils entrainent un conflit ou contredisent les nouvelles conditions du problème.

- *La transformation* est utilisée lorsqu'aucun substitut n'est trouvé. Une solution sera alors dérivée en se basant sur les contraintes et les caractéristiques de la solution requise. Une contrainte spécifie quelles propriétés la solution devrait ou ne devrait pas avoir. La solution doit se conformer aux contraintes, et aucune contradiction ou conflit n'est permis. Pour identifier les contraintes, une certaine connaissance prédéfinie ou heuristique doit être disponible.

Il est à noter que ces stratégies sont complémentaires.

2.7.2. L'adaptation par analogie

L'adaptation peut être classifiée selon deux types d'analogie. Watson dans [Watson et al , 1994] les distingue de la façon suivante : l'adaptation transformationnelle dont le principe est d'appliquer des règles d'adaptation à *Sol(source) (solution du cas source)* et l'adaptation dérivationnelle où l'on réutilise les formules qui ont produit *Sol(source) (solution du cas source)* pour obtenir *Sol(cible) (solution du cas cible)*.

- *L'adaptation transformationnelle* (aussi appelée « structurelle ») ne considère pas comment la solution a été obtenue. Dans ce cas de figure, les cas contiennent des descriptions et des solutions qui sont ensuite réutilisées pour apporter des modifications appropriées à la nouvelle situation [CARBONELL , 1983]. La solution trouvée n'est pas une solution directe pour le nouveau problème, mais il existe une connaissance sous forme d'opérateurs transformationnels qui peuvent être appliqués à la solution remémorée. On distingue deux sous-classes dans l'adaptation transformationnelle :
  - Par valeur : elle consiste à adapter les valeurs des attributs de la solution du cas source (lien avec la substitution de la section précédente).

- Par structure : elle consiste à adapter la structure de la solution du cas source (lien avec la transformation de la section précédente).

- *L'adaptation dérivationnelle* (ou générative) se focalise sur la méthode utilisée lors des épisodes antérieurs de résolution. Il s'agit d'exploiter les connaissances créées lors de la démarche de résolution suivie pour construire de nouvelles solutions capables de s'adapter aux nouveaux problèmes. Dans ce type d'adaptation, on stocke pour chaque cas, la trace des étapes qui ont permis de générer sa solution (avec les informations nécessaires pour sa compréhension : décisions prises, connaissances utilisées…). Cette traçabilité est souvent utilisée en logique de conception.

2.7.3. Les approches par décomposition

Cette adaptation consiste à diviser un problème d'adaptation en sous-problèmes plus simples de façon à les résoudre séparément. Pour ce faire, trois différentes approches ont été proposées :

- Décomposition du problème d'adaptation. Une fois le problème d'adaptation décomposé, chaque problème est résolu indépendamment afin de combiner les solutions partielles en une seule solution *Sol(cible) (solution du cas cible)*.

- L'adaptation différentielle. Les variations entre les cas, à partir d'un ensemble de dépendances entre descripteurs de problèmes et de solutions, sont modélisées localement grâce à une formule basée sur le calcul différentiel. Elle lie ces variations (exprimées comme des différentielles en un point (x, y) donné) aux dérivées partielles pour calculer *Sol(cible) (solution du cas cible)*.

- L'adaptation par reformulations. Elle combine l'adaptation par décomposition et la recherche dans un espace d'états. L'application de reformulations (chacune représentant un ensemble de liens entre l'espace de remémorations et l'espace des adaptations) permet de décomposer la tâche d'adaptation en introduisant des problèmes virtuels.

2.7.4. L'adaptation hiérarchique

Dans [Bentebibel, 2008], une explication a été donnée à propos de ce type d'adaptation : les cas sont stockés à plusieurs niveaux d'abstraction. L'adaptation est effectuée du plus haut niveau vers

le plus bas niveau. Au début, la solution est adaptée au niveau d'abstraction le plus élevé (les détails moins appropriés sont omis). Puis, la solution est affinée par étapes et les détails exigés sont ajoutés. Pour les différents niveaux d'abstraction, l'adaptation hiérarchique peut réutiliser des cas simples ou des cas multiples en utilisant différents détails ou différentes parties des solutions réutilisées .

2.7.5. L'adaptation combinatoire ou multi-cas

Cette adaptation consiste à remémorer plusieurs cas *source* afin de composer *Sol(cible) (solution du cas cible)* à partir des différentes *Sol(source)* remémorées. [Wilke & Bergmann, 1998] a nommé ce type d'adaptation comme l'adaptation compositionnelle. Elle peut être utilisée dans deux situations [ARSHADI N. et BADIE, 2000] :

- Lorsque la solution se compose de différentes parties indépendantes, les composants peuvent être adaptés avec plus ou moins de précision. Cette méthode est efficace s'il y a peu de conflits entre les composants [Wilke & Bergmann, 1998].
- Lorsque la solution ne peut pas être divisée en parties indépendantes mais que les solutions des cas similaires doivent être combinées.

2.7.6. L'adaptation « Machine Learning »

Plusieurs approches de « machine apprenante » ont été développées pour résoudre le problème d'adaptation. Dans ce type d'approche, l'apprentissage produit des heuristiques spécialisées qui se basent sur les différences entre *cible* et *source*. Ces heuristiques peuvent être employées pour déterminer la quantité d'adaptation qui est exigée. Un des avantages de ces techniques est que la connaissance d'adaptation peut être acquise automatiquement à partir de la base de cas. De plus, la maintenance de cette connaissance peut être plus facilement gérée et contrôlée en l'actualisant et en la vérifiant lors de confrontation avec des nouveaux cas. Elles utilisent plusieurs techniques de calcul dont par exemple : les réseaux bayésiens, les réseaux de neurones et les algorithmes génétiques.

2.7.7. L'adaptation par contraintes

Le principe de cette adaptation consiste à éliminer les violations des contraintes dans une solution potentielle en recherchant les substitutions acceptables pour *Sol(cible) (solution du cas cible)*. Le

processus d'adaptation basé sur une approche de contraintes se caractérise comme la recherche des affectations consistantes pour *cible* à l'aide de connaissances disponibles. Ces dernières pouvant être extraites directement de la base ou auprès de l'expert.

2.7.8. L'adaptation interactive

Ce type d'adaptation se distingue des autres en ce qu'il se déroule grâce à l'intervention ponctuelle de l'utilisateur. D'ailleurs, dans les premières années d'utilisation du RàPC, certains chercheurs [LEAKE D., KINLEY A. et WILSON D., 1996], considéraient l'adaptation automatique comme étant essentielle pour attribuer aux systèmes un haut degré d'autonomie et par conséquent une meilleure capacité à résoudre le problème d'adaptation en émulant l'action d'un expert. Cependant, d'autres auteurs, comme [MARK W., SIMOUDIS E. et HINKLE D., 1996], contredisaient cette idée en soutenant que l'automatisation dans l'adaptation n'était ni faisable pour arriver à une adaptation réussie ni indispensable pour assurer l'efficience d'un système de RàPC. Même si des techniques ont été récemment développées dans ce sens (e.g. machine apprenante, techniques d'exploration de données…), l'automatisation se révèle bien faisable avec des résultats satisfaisants. Toutefois un problème est rencontré car l'efficacité et la performance de ces méthodes dépendent considérablement de l'abondance de cas dans la base.
L'intervention de l'utilisateur peut être réalisée par :

- Une adaptation manuelle.
- Par le biais d'un modèle de préférences de l'utilisateur.
- A travers un processus itératif : le système propose une solution que l'utilisateur révise et modifie selon ses besoins. comme exemple l'adaptation conservatrice qui sera détaillée dans le chapitre 4 puisque c'est le type d'adaptation utilisé dans notre travail.

2.8. Domaines d'application du RàPC

Le RàPC possède de nombreuses applications dans des domaines très divers permettant d'accomplir des tâches de différentes natures. Parmi les domaines dans lesquels le RàPC est plus couramment utilisé, on peut citer citer les domaines médical, industriel, commercial, judiciaire, l'analyse financière, la maintenance, l'enseignement, les services de consultation. On trouve également plusieurs types d'applications comme la gestion des connaissances, la planification,

l'aide à la décision, la classification, la conception, la navigation des robots, la configuration, le maintien des contraintes de temps appliquées à de petits avions, la justification, la sécurité routière et le diagnostic. Selon les auteurs, différentes typologies d'application du RàPC sont proposées. Elles sont dépendantes du domaine abordé et de la nature de la tâche à réaliser.

-Kolodner [Kolodner, 1993] propose deux types d'applications :

- ***les systèmes de résolution de problèmes*** dans lesquels les solutions de nouveaux problèmes sont dérivées en se servant des solutions déjà connues comme des guides. Ils regroupent des tâches de conception, de planification et de diagnostic.
- ***les systèmes interprétatifs*** dans lesquels les situations déjà connues servent de contexte pour les nouvelles situations. Ils regroupent des tâches de justification, évaluation, interprétation et classification.

-Watson et Marir [Watson et Marir, 1994] séparent les applications, selon le type d'utilisation, en deux catégories à savoir ***les applications commerciales et académiques***. Tandis que [Althoff et al., 1995] les décomposent selon le type de la tâche à réaliser :

- ***Les tâches de classification*** comprenant les systèmes de prédiction, de help desk, d'évaluation et de diagnostic.
- ***Les tâches de synthèse/planification*** comprenant les systèmes de configuration, de conception et de planification.

-Althoff et Bartsch-Spörl [Althoff et Bartsch-Spörl, 1996] quant à eux, divisent les domaines d'application du RàPC en deux classes. Celles relatives aux tâches :

- ***analytiques*** qui sont sous forme de situations. Ces situations doivent être classifiées pour retrouver le cas correspondant et exploiter la solution. On retrouve dans ce type de tâche la classification d'objet, le diagnostic médical et la sélection de produits dans le e-commerce ;
- ***synthétiques*** nécessitant une *planification*, une *configuration* et une *construction* de la solution. Elles se présentent sous forme de problèmes auxquels il faut proposer une solution et décrire les conditions d'applications de la dite solution. Ce type de tâche englobe le transport, l'architecture, les systèmes d'aide à la décision, l'enseignement et la gestion des ressources.

-Althoff [Althoff, 2001] introduit la notion de la hiérarchie des applications par rapport à la complexité de la résolution de problème. En effet, il propose quatre niveaux hiérarchiques.

- ***la classification*** se trouve au premier niveau de la hiérarchie dans laquelle la solution du problème est en relation avec la sélection d'une ou plusieurs classes.
- ***le diagnostic*** vient juste après car il est considéré comme une généralisation de la classification. Cette généralisation concerne la connaissance générale du domaine qui va rechercher les informations nécessaires (symptômes) dans un processus de diagnostic.
- ***l'aide à la décision*** se trouve au troisième niveau hiérarchique et distingue les symptômes des solutions de problèmes qui ne sont pas toujours directes ni visibles.
- ***la gestion des connaissances*** est une application plus générale et plus complexe dans laquelle aucune méthode de raisonnement ne peut être utilisée directement.

-Richter [Richer, 2006] propose une catégorisation plus conventionnelle en s'inspirant de la catégorisation faite par [Althoff , 2001]. Les deux types de systèmes de RàPC sont :

- ***résolution de problèmes***, ce sont les systèmes qui s'occupent des tâches analytiques.
- ***prise de décision*** qui concerne les tâches synthétiques.

2.9. Conclusion

Le raisonnement à partir de cas (RàPC) est présenté comme un cas particulier du raisonnement par analogie . Le RàPC consiste à résoudre un problème en réutilisant des expériences antérieures similaires. Un cas est typiquement une situation de problème avec sa solution et éventuellement les moyens de dériver la solution depuis l'énoncé.

Ce chapitre a permis d'introduire la notion et les bases de système de raisonnement à partir de cas ainsi que de présenter ce formalisme. Nous exposons dans le prochain chapitre, l'acquisition de connaissances de domaines à partir d'analyse des erreurs et à travers le système FrakaS.

Chapitre 03

Acquisition de connaissances de domaines à partir d'analyse d'erreurs FrakaS

3.1. Introduction

FIKA (**F**ailure-driven **I**nteractive **K**nowledge **A**cquisition) [A.Cordier, 2009].est une méthodologie qui se focalise sur l'acquisition de connaissances du système de raisonnement à partir de cas qui est "difficile à saisir "et qui doit être capturé " dans le contexte ". C'est une approche interactive et opportuniste du processus d'acquisition des connaissances de l'expert.

Le principe de FIKA est de faire plusieurs observations en utilisant différentes expériences dans l'acquisition de connaissances dans un système de raisonnement à partir de cas (RàPC). Pour implémentée ces principes elle utilise deux modèles : IAKA (**I**nter**A**ctive **K**nowledge **A**cquisition) et FrakaS (**F**ailu**R**e **A**nalysis for domain Knowledge **A**cqui**S**ition).

Ce chapitre présente l'approche FrakaS pour l'acquisition interactive des connaissances de domaine dans les systèmes RàPC. L'approche est basée sur la notion des échecs d'adaptation.

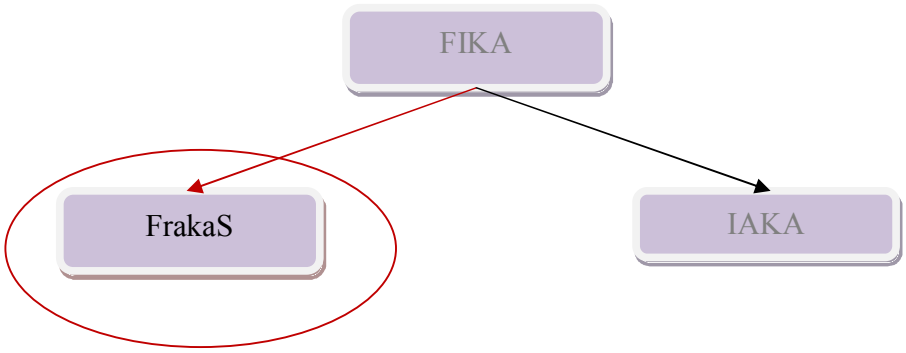


Figure 3.1. FIKA

3.2. Définitions

3.2.1. Approche opportuniste

Le processus d'acquisition de connaissances dans FIKA est déclenché par des «échecs de raisonnement»: qui est l'aspect opportuniste de l'approche. Un échec de raisonnement se produit lorsque la solution proposée par le système est jugée insatisfaisante par l'expert. En supposant que le moteur de raisonnement est correct (en d'autres termes, que le système ne se trompe pas), alors nous devons admettre dans un tel cas que la connaissance utilisée pour résoudre le problème est incorrecte, incomplète ou inadaptée à la situation donnée et doit donc être améliorée. Un problème important est de savoir comment reconnaître un échec de raisonnement et quand commencer le processus d'acquisition interactive [Aamodt, 1989].

3.2.2. Approche Interactive

L'aspect interactif de l'approche FIKA réside dans le fait que l'acquisition de connaissances par construction est faite par les interactions entre l'expert et le système. Le processus d'acquisition de connaissances fait une partie importante du processus de raisonnement à partir de cas. Cette approche est assurée de ne pas détourner l'expert de sa tâche principale. Ensuite, la solution du problème est une combine dans lequel les capacités de raisonnement et les connaissances du système et l'expert sont "mélangées" en une boucle afin d'atteindre la cible [A.Cordier, 2009]. L'acquisition de nouvelles connaissances peut donc bénéficier de cette boucle.

Expert

Un expert en général pour résoudre un problème, il utilise une approche basée sur un «savoir-faire» à la place d'une théorie explicite. Dans notre travail, nous considérons l'expert comme un spécialiste du domaine. L'expert à une connaissance qui le rend capable de comprendre son domaine, il est en mesure d'expliquer pourquoi une solution donnée par un système est satisfaisant et qu'il est en mesure d'identifier les «erreurs» dans la connaissance modélisée ou, dans certains cas, un manque de connaissances [A.Cordier, 2009].

3.3. Acquisition de connaissances de domaines à partir d'analyse d'erreurs(FrakaS)

FrakaS (Failure Analysis for Domain Knowledge Acquisition) est une illustration des principes de FIKA. FIKA définit une approche générale pour l'acquisition interactive et opportuniste des connaissances dans le système de raisonnement à partir de cas. Il définit des stratégies pour apprendre de manière interactive des connaissances de domaine, en exploitant les erreurs du raisonnement et leur correction. Le processus d'apprentissage se produit au cours d'une session de raisonnement à partir de cas [A.Cordier, 2009].

L'acquisition de connaissances dans le Système FrakaS est :

- Opportuniste, car le système exploite les échecs pour déclencher un processus d'apprentissage.
- Interactif, car il implique l'utilisateur pendant la session du système de raisonnement à partir de cas par interaction
- Incrémental, car il ajoute progressivement des morceaux de connaissances aux connaissances du domaine.

FRAKAS

3.3.1. FrakaS principes

Le processus de raisonnement à partir de cas exploite la base de connaissances pour produire une solution candidate. Lorsque la solution candidate est jugée non valide par l'utilisateur, l'expert identifie un sous-ensemble de connaissances incompatibles (notée Inc dans l'algorithme ci dessous). A partir de ce sous-ensemble de connaissances, le système est capable d'apprendre un nouveau morceau de connaissance. Cette nouvelle connaissance est ajoutée à la base de connaissances ainsi, la base de connaissances améliorée permet au système de produire une nouvelle solution candidate pour le problème actuel. Le processus est répété jusqu'à ce que l'expert valide une solution proposée par les systèmes, à savoir jusqu'à ce que le système trouve une solution adéquate pour le problème.

Le processus de RàPC mis en œuvre dans FrakaS exploite une base de cas ainsi que la base de connaissances du domaine. Les cas contenues dans la base de cas sont supposés être compatibles avec la base de connaissance du domaine, elles contiennent souvent des morceaux de connaissances provenant de l'expérience. Ces morceaux de connaissances ne peuvent pas toujours être expliqués par les connaissances du domaine, mais sont néanmoins souvent très utiles et donc précieux. Afin de résoudre un nouveau problème, FrakaS récupère un cas qui est, bien sûr, compatible avec la base de connaissances du domaine [A.Cordier, 2009].

Lorsque l'expert souligne que la proposition n'est pas satisfaisante, cela signifie qu'il y a une incohérence entre la base de connaissance du domaine et certains des descripteurs de problème cible. Ensuite, l'expert doit identifier l'incohérence en marquant les parties du problème et la solution qui sont concernés.

Dès que l'incohérence est correctement identifiée, le système est capable d'apprendre un morceau de connaissances qui lui permettra d'éviter de reproduire l'échec dans d'autres raisonnements. Une fois qu'il est résolu, le cas cible est ajouté à la base de cas en prévision d'une éventuelle réutilisation ultérieure.

Algorithme de FrakaS

Entrée : un problème cible, une base de cas et des connaissances du domaine CD

Début (algorithme)

La remémoration avec le problème cible retourne le cas (srce, Sol(srce)).

Sol(cible) ← Adaptation(CD, (srce, Sol(srce)), cible)

FRAKAS

/* Prise en compte des échecs du premier type */

tant que l'expert trouve que cible ∧ Sol(cible) est incohérent avec ses connaissances

L'expert met en évidence l'échec Inc et un texte explicatif.

CD ← CD ∧ ¬Inc

Le texte explicatif est stocké pour une acquisition des connaissances ultérieure.

Sol(cible) ← Adaptation(CD, (srce, Sol(srce)), cible)

fin (tant que)

/* Prise en compte des échecs du deuxième type */

Si Sol(cible) n'est pas partielle alors sortir

tant que l'expert trouve une inconsistance dans certaines interprétations de cible ∧ Sol(cible)

pour toute interprétation inconsistante I

L'expert met en évidence l'échec Inc et un texte explicatif

CD ← CD ∧ ¬Inc

Le texte explicatif est stocké pour une acquisition des connaissances ultérieure.

fin (pour)

Sol(cible) ← Adaptation(CD, (srce, Sol(srce)), cible)

fin (tant que)

fin (algorithme)

3.4. FrakaS logique de description

Dans les travaux précédents de cordier les principes du système FrakaS ont été implémentés en utilisant la logique propositionnelle et l'adaptation est réalisée en utilisant le principe de l'adaptation conservatrice. Dans notre travail nous utilisons le même système mais le formalisme de représentation des connaissances choisi est celui de la logique de description en utilisant toujours l'adaptation conservatrice puisque ce système se base sur cette dernière.

Cette section décrit les principaux aspects de la mise en œuvre de FrakaS-logique de description, formalisme de représentation des connaissances choisi et les principes de l'adaptation conservatrice.

3.4.1. Formalisme Logique de description

Cette section présente des rappels et des notions de logique de description.

Les logiques de descriptions (LD) ou logiques terminologiques forment une famille de langages de représentation de connaissances inspirée de la logique des prédicats, des réseaux sémantiques et des langages de schémas. Les logiques de descriptions sont largement utilisées pour implémenter des systèmes à base de connaissances. Elles fournissent un modèle symbolique pour la représentation des connaissances du domaine d'une application. Ces logiques s'appuient sur les notions de concepts, de rôles et d'individus pour représenter les connaissances du domaine.

Les logiques de descriptions sont largement utilisées pour implémenter des systèmes à base de connaissances. Elles fournissent un modèle symbolique pour la représentation des connaissances du domaine d'une application donnée. Ces logiques s'appuient sur les notions de concepts, de rôles et d'individus pour représenter les connaissances du domaine [A. Napoli, 1997]. Un concept est un ensemble d'individus, alors qu'un rôle est une relation binaire entre individus. Toutes les logiques de descriptions possèdent les propriétés suivantes :

-Un concept et un rôle possèdent une description structurée, élaborée à partir d'un certain nombre de constructeurs. Une sémantique est associée à chaque description de concept et de rôle par l'intermédiaire d'une interprétation. Les manipulations opérées sur les concepts et les rôles sont réalisées en accord avec cette sémantique [A. Napoli, 1997].

- Les concepts : dénotent un ensemble d'individus, ils peuvent être atomiques, ou définis. Les concepts atomiques sont énoncés et servent à la construction des concepts primitifs et définis. Un concept défini est considéré comme une définition : l'ensemble des expressions associées à un concept défini C joue le rôle d'ensemble de conditions nécessaires et suffisantes (CNS) pour qu'un individu qui les possède soit classé sous ce concept. En revanche, les propriétés d'un concept primitif sont nécessaires mais non suffisantes. Il n'est pas possible d'affirmer qu'un individu est un représentant d'un concept primitif en examinant ses seuls rôles [Baader et al, 2003].

En général un concept est soit un concept atomique (c.-à-d. un nom de concept), ou une expression conceptuelle de l'une des forme suivantes : $\top$, $\bot$, $\neg$ C, C $\sqcap$ D, C $\sqcup$ D, $\forall$r.C , $\exists$r.C, où C et D sont des concepts (atomiques ou non) et r est un rôle.

- Les rôles : correspondent à un lien entre deux éléments et est interprétés comme une relation binaire sur un univers donné. Les rôles peuvent être atomiques, ou complexes

obtenus en utilisant les constructeurs de rôles autorisés par le langage de logique de description.

-**Un individu** d'une logique de description LD correspond à un élément d'un univers donné.

-Les connaissances Il existe deux niveaux de représentation des connaissances, le niveau terminologique (ou intensionnel) et le niveau factuel (ou extensionnel). Les concepts et les rôles sont définis et manipulés au niveau terminologique dans la TBox (**T**erminology **Box**). La description et la manipulation des individus s'effectue au niveau factuel dans la ABox (**A**ssertion **Box**) (voir figure 3.2).

-La base de connaissances dans les logiques de description est constituée d'une Base terminologique (TBox) et une Base des assertions ou faits (ABox).

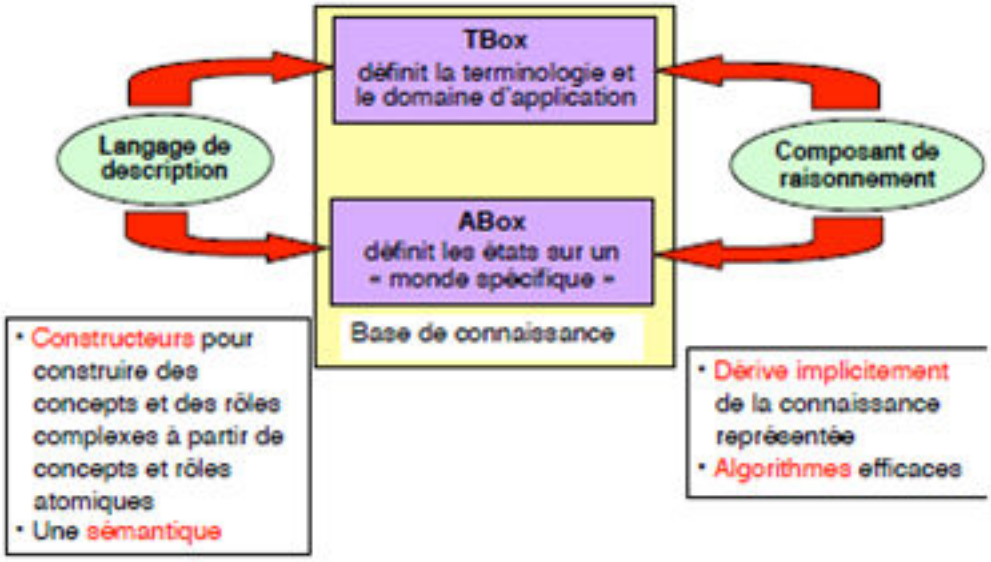


Figure 3.2. base de connaissance - Logique de description tiré de [Baader, 2003]

- La relation de subsomption permet d'organiser les concepts et les rôles par niveau de généralité : intuitivement, un concept C subsume un concept D si C est plus général que D au sens où l'ensemble d'individus représenté par C contient l'ensemble d'individus représenté par D. Une base de connaissances se compose alors d'une hiérarchie de concepts et de rôles.

- Les opérations qui sont à la base du raisonnement terminologique sont la classification et l'instanciation. La classification s'applique aux concepts, le cas échéant aux rôles, et permet de

déterminer la position d'un concept et d'un rôle dans leurs hiérarchies respectives, la construction et l'évolution de ces hiérarchies est ainsi assistée par le processus de classification [A. Napoli, 1997].

NB : *les logiques de descriptions peuvent être utilisées pour exprimer :*

- *Des connaissances intensionnelles sur des concepts et leurs relations,*
- *Des connaissances extensionnelles sous forme d'assertions sur les individus,*
- *Des requêtes sur les concepts et sur les individus.*

3.4.1.1. Base terminologique

3.4.1.1.1. Syntaxe des concepts et des rôles

Les LD forment un sous-ensemble de la logique du premier ordre. Elles n'utilisent ni fonctions, ni variables explicites, mais des prédicats. Les prédicats unaires correspondent aux concepts, les prédicats binaires aux rôles. La description des concepts et des rôles est fondée sur l'utilisation de constructeurs.

Les constructeurs : permettent la combinaison de concepts et rôles atomiques pour former des entités composées :

Les LD se distinguent par les constructeurs qu'elles proposent :

- Plus elles ont de constructeurs, plus elles sont expressives, et ont des chances d'être non décidables ou de complexité très élevée.
- les LD trop peu expressives ne permettent pas de représenter des domaines complexes.

3.4.1.1.2. Sémantique des concepts et des rôles

La sémantique associée aux logiques de descriptions est définie par une interprétation $(\Delta^I , .^I)$. Δ^I est un ensemble non vide qui désigne un domaine d'interprétation et « $.^I$ » est une fonction d'interprétation. Par l'intermédiaire de la fonction « $.^I$ » les concepts sont interprétés comme des sous-ensembles de Δ^I, et les rôles sont interprétés comme des sous-ensembles du produit $\Delta^I \times \Delta^I$. Un modèle d'un concept C est une interprétation ou C^I (l(image de C par I) est un ensemble non vide. Un concept C est satisfiable s'il a un modèle (voir tableau 3.1) .

Terme atomique	*Syntaxe*	*Sémantique*
Individu	a	$I(a) \in \Delta$
Concept atomique	A	$I(A) \subseteq \Delta$
Rôle atomique	R	$I(R) \subseteq \Delta \times \Delta$

Tableau 3.1. Terme atomique

Constructeurs de concepts	*Syntaxe*	*Sémantique*
Concept universel	$\top$	$I(\top)=\Delta$
Concept vide	$\bot$	$I(\bot)= \emptyset$
Conjonction	$C \sqcap D$	$I(C \sqcap D)=I(C) \cap I(D)$
Disjonction	$C \sqcup D$	$I(C \sqcup D)=I(C) \cup I(D)$
Négation	$\neg C$	$\Delta \setminus I(C)$
Restriction existentielle	$\exists R.C$	$\{x \mid \exists y, <x,y> \in I(R) \wedge y \in I(C)\}$
Restriction de valeur	$\forall R.C$	$\{x \mid \forall y, <x,y> \in I(R) \Rightarrow y \in I(C)\}$
Restriction numérique	$\leq nR$	$\{x \mid card(\{y, <x,y> \in I(R)\}) \leq n\}$
Restriction numérique	$\geq nR$	$\{x \mid card(\{y, <x,y> \in I(R)\}) \geq n\}$
Nominaux	$\{a_1, \ldots\ldots, a_n\}$	$\{I(a_1), \ldots\ldots, I(a_n)\}$

Tableau 3.2. Constructeurs de concepts

- Le concept Top (T) est un concept qui est à la racine de la hiérarchie des concepts. Il subsume tous les autres concepts. Tous les individus dans la base de connaissances sont des instances du concept Top.
- Le concept Bottom ($\bot$) est subsumé par tous les concepts de la hiérarchie et il n'a pas d'instance.
- La négation de concept est définie par le constructeur $\neg$. Par exemple, ($\neg$ Relation) est un concept dont les instances sont tous les individus qui ne sont pas des instances du concept Relation.

- La conjonction de concepts (⊓) définit l'intersection de concepts. L'expression (Sommet ⊓ ¬ Relation) définit le concept Région comme la conjonction du concept Sommet et de la négation du concept Relation.
- La disjonction de concepts (⊔) représente l'ensemble des individus qui appartiennent au moins à l'un des concepts de la disjonction.
- La quantification existentielle typée : l'expression ∃R .C définit une contrainte sur les valeurs du rôle R , le codomaine est restreint au concept C, mais elle exige aussi l'existence d'au moins un couple d'individus (x,y) liés par l'intermédiaire de R, tel que y est une instance de C. Par exemple, le concept Relation définit un ensemble d'individus qui ont (au moins) un arc qui est de type Région. La quantification existentielle simple est l'´equivalent d'une quantification existentielle typée avec le concept Top comme restriction du codomaine du rôle R.
- La quantification universelle typée : une contrainte sur le codomaine des valeurs d'un rôle est introduite par l'expression ∀R.C, le codomaine du rôle R est restreint au concept C. Contrairement `a l'expression ∃R.C, cette expression impose une contrainte sur le codomaine de R mais elle n'exige pas l'existence d'individus en relation par le rôle. La quantification universelle simple est l'´equivalent d'une quantification universelle typée avec le concept Top comme restriction du codomaine du rôle R (voir tableau 3.2 et 3.3).
- La cardinalité typée : les constructeurs ≥ nR .C, ≤ nR .C et = nR .C fixent à n le nombre minimal, maximal ou exact des individus y auxquels un individu x est lié par le rôle R en restreignant le codomaine du rôle à un concept particulier C. Une cardinalité simple est équivalente à une cardinalité typée avec le concept Top comme restriction du codomaine du rôle R. Par exemple, l'expression (= 3arc .Région) définit des relations ayant trois arc reliant trois Région, i.e. des relations ternaires [JL. Metzger, 2010].

Constructeurs de rôles	*Syntaxe*	*Sémantique*
Conjonction de rôles	$R \sqcap S$	$I(R \sqcap S)=I(R) \cap I(S)$
Disjonction de rôles	$R \sqcup S$	$I(R \sqcup S)=I(R) \cup I(S)$
Complément de rôles	$\neg R$	$(\Delta x \Delta) \setminus I(R)$
Clôture transitive	R^+	Cloture transitive de I(R)
Rôle inverse	R^-	$\{<y,x> \mid <x,y> \in I(R)\}$
Composition de rôles	$R \circ S$	$\{<x,z> \mid \exists y, <x,y> \in I(R) \wedge <y,z> \in I(S)\}$

Tableau 3.3. Constructeurs de rôles

La définition formelle de TBox

Une TBox contient des axiomes terminologiques de la forme $C \equiv D$ ou $C \sqsubseteq D$. La première sert à énoncer des relations d'équivalence entre concepts, alors que la seconde permet d'exprimer des relations d'inclusion. Une interprétation I satisfait un axiome $C \equiv D$ si et seulement si $C^I = D^I$. Une interprétation I satisfait un axiome $C \sqsubseteq D$ si et seulement si $C^I \subseteq D^I$. Une interprétation satisfait une TBox (est un modèle de TBox) si et seulement si l'interprétation satisfait tous les axiomes de la TBox (voir tableau 3.4).

Axiome de la Tbox	*Syntaxe*	*Contrainte d'interprétation*
Subsomption	$C \sqsubseteq D$	$I(C) \subseteq I(D)$
Inclusion de rôles	$R \sqsubseteq S$	$I(R) \subseteq I(S)$
Transitivité de rôles	Trans(R)	$I(R)= I(R^+)$

Tableau 3.4. Axiome TBox

3.4.1.1.3. Raisonnement dans la base des concepts

Trois types de raisonnements sont effectués dans la partie intensionnelle de la base de connaissances (TBox) qui sont : le test de subsomption, le test de satisfiabilité, et la classification.

Le test de subsomption est un mécanisme sur lequel peuvent se baser les deux autres mécanismes.

- **Test de subsomption :** Déterminer si le concept D subsume le concept C revient à vérifier que le concept D (le subsumant) est plus général que C (le subsumé). Autrement dit, la subsomption revient à vérifier que l'ensemble des instances de C est un sous-ensemble de celui de D. Formellement, C $\sqsubseteq$ D si et seulement si $C^I \subseteq D^I$ pour toute interprétation I.
- **Test de satisfiabilité :** Vérifier qu'un concept D est satisfiable revient à vérifier qu'il a une interprétation I telle que $D^I \neq \emptyset$, sinon D est non satisfiable.
- **Test de classification :** La classification est le processus de base pour construire la hiérarchie des concepts. La classification consiste à positionner un concept par rapport aux autres concepts de la hiérarchie. La relation de subsomption entre le concept à positionner et tous les concepts dans la hiérarchie est vérifiée, puis le concept est placé entre ses subsumants les plus spécifiques et ses subsumés les plus généraux. Le concept D est un subsumant plus spécifique du concept C si et seulement si C $\sqsubseteq$ D et s'il n'existe pas un autre concept F tel que C $\sqsubseteq$ F $\sqsubseteq$ D. Pour les subsumés, D est un subsumé plus général de C si et seulement si D $\sqsubseteq$ C et s'il n'existe pas un autre concept F tel que D$\sqsubseteq$ F$\sqsubseteq$ C.

3.4.1.2. Base des assertions

3.4.1.2.1. Assertions

La base des assertions ou faits contient les connaissances extensionnelles concernant le domaine de l'application. Les assertions concernant l'appartenance des individus aux concepts sont appelées instanciations de concepts, elles sont de la forme C(a), ou l'individu « a » appartient au concept C. De la même façon, les assertions concernant les relations entre les individus sont des instanciations de rôles. Elles sont de la forme R(a, b) ou les individus a et b sont liés par le rôle R (voir tableau 3.5).

Axiome de la Abox	*Syntaxe*	*Contrainte d'interprétation*
Appartenance à un concept	C(a)	$I(a) \in I(C)$
Appartenance à un rôle	$R(a_1,a_2)$	$\{<I(a_1), I(a_2)> \in I(R)$
Identité	$a_1 = a_2$	$I(a_1) = I(a_2)$

Tableau 3.5. Axiome de ABox

3.4.1.2.2. Raisonnement dans la base des assertions

Plusieurs types de raisonnements sont possibles dans la base des assertions(Faits).

- **L'instanciation** : consiste à vérifier qu'un individu x est une instance du concept C.
- **La réalisation** : permet de trouver les concepts les plus spécifiques dont un individu x est une instance.
- **Le calcul de l'extension** : consiste à trouver toutes les instances d'un concept C.

Exemple :

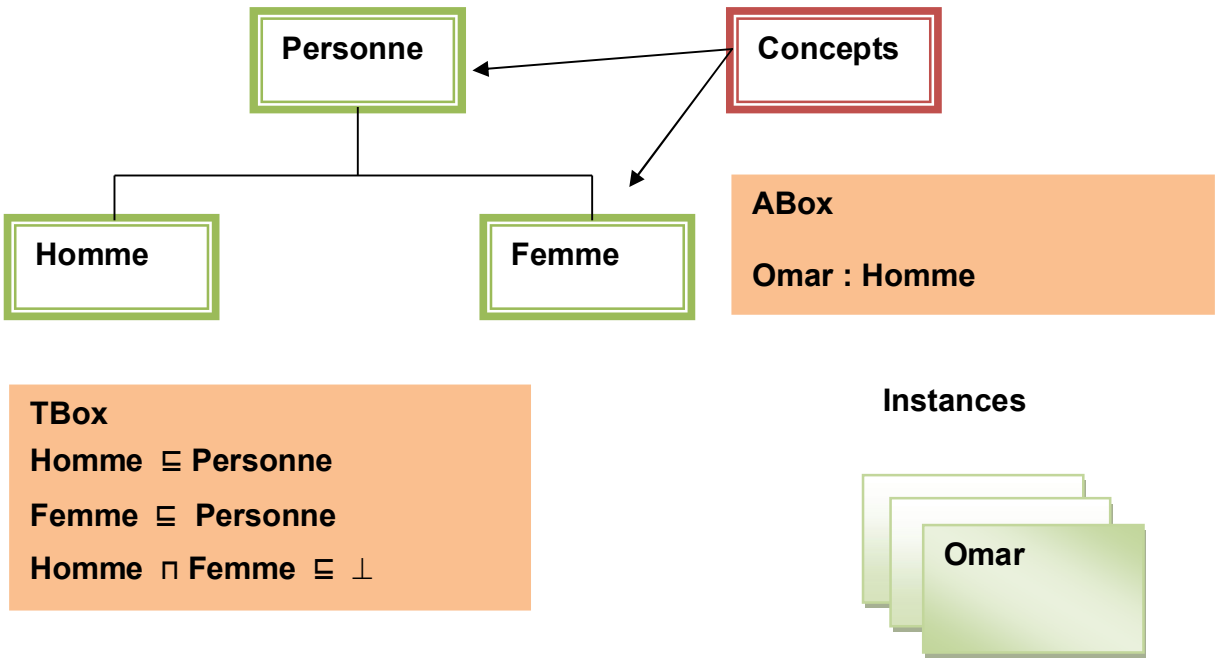


Figure 3.3.Exemple d'une base de connaissance dans la logique de description

3.4.2. L'Adaptation Conservatrice

Le principe du système FrakaS est l'acquisition de connaissances du domaine dans les systèmes de raisonnement à partir de cas en utilisant l'adaptation conservatrice. Cette section décrit brièvement les principes de l'adaptation conservatrice car une bonne compréhension de cette approche d'adaptation est nécessaire pour comprendre comment le système FrakaS exploite les défaillances du raisonnement pour apprendre la connaissance [Lieber, 2007b].
Cette approche de l'adaptation consiste à faire des modifications « minimales » sur le cas source afin d'être cohérent avec à la fois le problème cible et les connaissances du domaine. Elle est formalisée grâce à la notion d'opérateur de révision qu'on va détaille dans la section qui suit.

3.4.2.1. Connaissances utilisées par l'adaptation conservatrice

L'adaptation conservatrice repose sur trois types de connaissances dans les systèmes de raisonnement à partir de cas qui sont:
- Base de connaissance source : données lié au problème source et sa solution. C'est l'ancienne connaissance qui peut être modifiée, mais de façon minimale.
- Base de connaissance cible : données lié au problème cible. Sont les nouvelles connaissances qui ne doivent pas être modifiées au cours du processus d'adaptation.
- Connaissances du domaine : sont les connaissances qui doivent être vrai dans tout contexte.

3.4.2.2. Principe de l'adaptation conservatrice

L'adaptation conservatrice consiste à effectuer des changements sur les connaissances source afin de trouver une solution pour le problème cible, en restant conforme à la fois avec le problème cible et les connaissances du domaine. Ce processus est dit conservateur parce que le changement fonctionne avec un changement minimal tout en restant cohérant avec la définition du problème cible [Lieber, 2006].
L'idée de l'adaptation conservatrice est liée à la théorie de la révision des bases de connaissances [Alchourrón et al, 1985], [Konieczny 1999]: la révision d'une ancienne base de connaissances par un nouveau consiste à faire un changement minime sur l'ancien tout en étant compatible avec ce dernier. La révision d'une base de connaissances est effectuée par un opérateur de révision désigné par « ∘ », l'opérateur de révision joue le rôle d'un opérateur d'adaptation dans le domaine CBR.

3.4.3. Opérateur de révision

L'adaptation conservatrice est une approche de l'adaptation en raisonnement à partir de cas qui s'appuie sur le principe du changement minimal(ou bien révision) du contexte source vers le contexte cible. Cette notion de changement minimal du contexte se retrouve dans la notion d'opérateur de révision des connaissances qu'on va détailler dans cette section.

3.4.3.1. Principe de notion de révision des connaissances

La révision de connaissances M par d'autres connaissances D est une opération qui vise à assimiler M à D en obtenant un résultat cohérent. Pour cela, il peut être nécessaire de remettre en cause une partie de M, afin de la rendre cohérent avec D, cette opération aura lorsque les connaissances M représentent imparfaitement le domaine et que des connaissances D, considérées comme plus exactes, sont acquises [Cojan et al, 2008].

3.4.3.2. Définition

Un opérateur de révision « ∘ » associe à deux bases de connaissances ψ et μ la base de connaissances $\psi \circ \mu$ qui, intuitivement, est obtenue par changement minimum sur ψ pour être cohérente avec μ.

Operateur de révision Dalal

Le principe de l'opérateur de révision de Dalal est que les modèles de la base de connaissances révisée par l'opérateur devraient être les modèles de la nouvelle base de connaissances reçue qui ont une distance minimale avec la base de connaissance révisée [Dalal, 1988].

L'operateur de révision Dalal doit satisfait les postulats de Katsuno et Mendelzon tout en adaptant ces postulats a la logique de description qui sera détaillé dans le chapitre suivant.

3.5. Synthèse

Le tableau 3.6 présente une synthèse des spécificités de FrakaS et leur lien avec les grands principes FIKA.

Description	Principes
Connaissance intensive	La connaissance vient de l'expert de domaine et est recueillie au cours des interactions, sous la forme de connaissance du domaine.
Interactive	L'expert est capable d'interagir avec le système après chaque session de résolution de problèmes. En cas de défaillance, l'expert corrige les connaissances impliquées et le système réessaie pour résoudre le problème avec ses connaissances nouvellement disponibles.
Opportuniste	Les échecs déclenchent le processus d'acquisition de connaissances. En cas de défaillance, l'expert doit corriger avant de pouvoir poursuivre le processus de résolution de problèmes.

Tableau 3.6. Synthèse du système FRAKAS

3.6. Conclusion

Le système FrakaS vise à faciliter l'acquisition des connaissances de domaine. Cette connaissance permet d'adapter des nouveau cas, et dans FrakaS, les connaissances à acquérir sont obtenues uniquement par l'analyse de la solution. En cas d'échec, la solution est analysée par l'expert qui doit identifier les incohérences en utilisant ses propres connaissances. Ainsi, à partir de l'analyse de ces contradictions, le système déduira de nouvelles connaissances qui l'empêchent de répéter l'erreur à l'avenir.

Nous avons présenté dans ce chapitre le système FrakaS, qui est en fait une approche pour l'acquisition des connaissances de domaine incrémental basé sur l'exploitation des défaillances du raisonnement (à savoir des échecs d'adaptation) d'un système de raisonnement à partir de cas.

Cette approche convient aux cadres où l'adaptation produit des solutions qui sont compatibles avec les connaissances du domaine disponibles, comme c'est le cas avec l'adaptation conservatrice.

Dans le chapitre suivant nous détaillons le prototype FrakaS DL (Description Logique) et le formalisme de l'adaptation conservatrice dans la logique de description avec l'opérateur de révision choisi.

Chapitre 04

Discussions de différentes méthodes d'acquisition des connaissances

4.1. Introduction

Le raisonnement à partir de cas est un paradigme de résolution de problèmes qui utilise des expériences passées pour résoudre de nouveaux problèmes. La réutilisation de l'expérience constitue la principale spécificité et la force du raisonnement par cas, le raisonnement se base sur la mémorisation et la réutilisation des situations du passé plutôt que sur l'usage exclusif de la connaissance formelle du domaine. L'exploitation des situations passées est souvent utile; il offre une "base" pour la solution chaque fois que la connaissance du domaine est incomplète. Même si la solution proposée n'est pas forcément idéale, elle est presque toujours satisfaisante dans les cas réels.

Dans un système de raisonnement à partir de cas Il ya une grande différence entre les connaissances du domaine, et la connaissance de l'expert, car les connaissances du domaine sont disponibles mais ne sont pas suffisantes, il est donc nécessaire d'en acquérir de nouvelles. Ce problème est impossible à résoudre complètement dans la plupart des applications, mais on peut encore apprendre de nouvelles connaissances de domaine grâce à l'expert.

Dans ce chapitre présentons dans un premier lieu une approche d'acquisition interactive des connaissances du domaine dans un système de raisonnement à partir de cas. Une version modifiée du système Frakas est utilisée pour mettre en œuvre cette approche. La modification consiste à utiliser les logiques de description à la place des logiques propositionnelles, ainsi que la sélection d'un opérateur de révision adéquat pour la révision de la base de connaissances dans les logiques de description. Dans un second lieu nous présentons une implémentation de la méthode REX (Retour d'EXpérience) pour la capitalisation des connaissance d'une entreprise qui est une autre méthode pour acquérir des connaissances du domaine, puis nous nous intéressons à la capitalisation des connaissances en informatique en utilisant l'approche gestion des connaissances. Nous avons proposé un générateur de systèmes experts en utilisant le chainage avant et nous terminons par le choix d'un domaine de puissance pour appliquer notre système expert qui est le domaine d'application en distribution.

4.1.1. Notions de base, notations et hypothèses sur le RÀPC

Dans un système de Raisonnement à Partir de cas (RÀPC), on peut distinguer deux types de cas : *cas source* et *cas cible*. Le cas source est celui dans lequel les parties « problème » et « solution » sont renseignées. Donc, c'est un cas dont on va s'inspirer pour résoudre un nouveau problème. Le cas source peut aussi contenir une autre partie appelée « information de qualité ».

Cette partie contient des informations sur l'utilisation du cas dans le système. Quant au cas cible, c'est celui qui concerne le problème à résoudre et dont sa partie solution n'est donc pas renseignée.

Le raisonnement à partir de cas peut se voir comme une fonction, qui, à l'énoncé d'un nouveau problème ou problème cible, fait correspondre une solution Sol(cible), en tirant parti d'un ensemble de cas, qui sont des problèmes déjà résolus accompagnés de leurs solutions. Un cas mémorisé, ou cas source (srce), est la donnée d'un couple énoncé de problème – solution (srce, Sol(srce)) ; une base de cas est un ensemble fini de cas Base-de-cas = {(Pi, Sol(Pi))/i = 1, .., n}. Il est classique de considérer le problème cible comme un nouveau cas cas-cible = (cible, Sol(cible), pour lequel il faut construire Sol(cible) en s'aidant d'un cas mémorisé cas-source = (source, Sol(source)).

Un cas est donc l'association d'un problème et de la solution de ce problème : ***cas=(pb, Sol(pb))***.

Un cas source est un cas dont on va s'inspirer pour résoudre un nouveau cas que l'on appelle un cas cible. Un cas source s'écrit : ***cas-source=(source, Sol(source))***

et un cas cible s'écrit donc ***cas-cible=(cible, Sol(cible))***.

Dans notre travail, nous posons que :

- L'adaptation utilisée par le système du raisonnement par cas produit un résultat cohérent avec les connaissances du domaine (mais pas nécessairement avec les connaissances de l'expert).
- Il existe une différence entre une solution qui résout totalement un problème et une solution qui ne le résout que partiellement : celle-ci ne décrit qu'insuffisamment une solution exploitable par l'utilisateur.
- Chaque problème (resp., solution) codé dans le système de RàPC représente un ensemble d'instances de problèmes (resp., de solutions).
- Un problème (resp., une solution) est représenté(e) par un ensemble de descripteurs interprété de façon conjonctive : si pb = {d1, . . . dn}, cela signifie que pb décrit le problème dont les instances vérifient chacun des descripteurs di (i 2 {1, . . . n}).

4.1.2. Principe utilisé

Dans notre travail, nous abordons deux types d'échecs mentionnés dans le travail d'Amélie Cordier et Béatrice Fuchs [Cordier et al, 2007]

- Premier mode d'échec: incohérence de la solution adaptée avec l'expertise. L'expert souligne que, compte tenu de sa connaissance du domaine, l'évaluation «Sol (cible) résout (cible)» est incohérente. Cela pourrait signifier que la solution en elle-même est incohérente (ou irréalisable) ou que la solution est incompatible avec le contexte du problème cible Dans ce cas, on attend de l'expert qu'il mette en évidence la partie la plus petite possible Sol(cible) qui soit incohérente avec ses connaissances et l'énoncé du problème cible. Cette partie de la solution est un sous-ensemble Inc de l'ensemble des descripteurs de Sol(cible). Une première connaissance acquise (et ajoutée à CD) est le fait que « Inc est faux ». On peut alors refaire le raisonnement avec les nouvelles connaissances du domaine.
- Si la solution Sol(cible) proposée par l'adaptation utilisée est partielle, et donc qui n'est pas satisfaisante, l'interaction avec l'expert peut permettre de la préciser. Soit IS, l'ensemble des instances de Sol(cible). Si une telle instance s ϵ IS est jugée satisfaisante par l'expert, elle constitue une solution du problème cible. Si au contraire, elle est incohérente avec ses connaissances expertes, on attend de l'expert qu'il mette en évidence une partie Inc minimale des descripteurs de s. On se ramène alors à l'acquisition des connaissances selon le premier type d'échec : on ajoute à CD « Inc est faux » et on demande à l'expert une explication, source d'acquisition des connaissances avec l'expert et un ingénieur de la connaissance ou Cogniticien.

4.1.3. FrakaS (Acquisition de connaissances de domaines à partir d'analyse d'erreurs)

FrakaS est un prototype qui implante les principes énoncés ci-dessus dans le cadre d'une représentation en logique de description (voir annexe A).

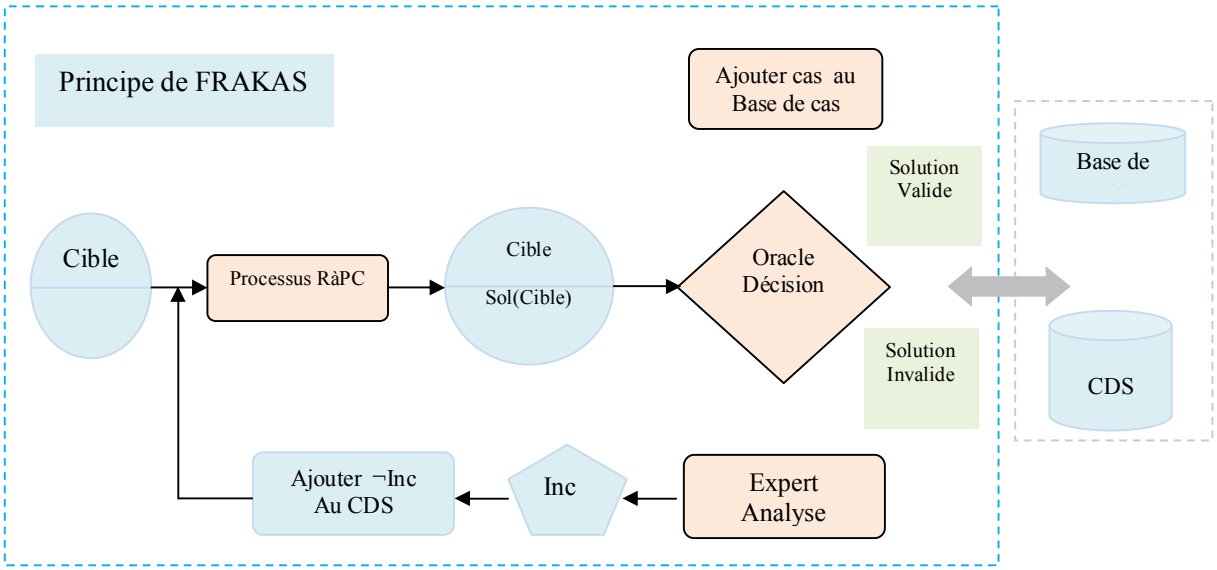


Figure 4.1. Principes de FrakaS

Cette figure décrit les principes majeurs de FrakaS et les liens avec la base de connaissances (à droite de la figure). Les cercles représentent des cas, les rectangles arrondis sont des processus (l'analyse d'expert implique des interactions entre l'expert et le système). Inc est un savoir construit pendant le cycle de raisonnement et qui sera ajouté à la base de connaissances.

Le processus de RàPC exploite la base de connaissances pour produire une solution candidate. Lorsque la solution candidate est jugée non valide (c'est-à-dire qu'elle ne fonctionne pas) par l'oracle (L'oracle "sait" si une solution est correcte ou non. De plus, l'oracle ne fait jamais d'erreurs. Nous pourrions comparer le concept d'oracle que nous utilisons ici à la notion de "résultat"), l'expert doit identifier un sous-ensemble de connaissances incohérentes (noté Inc sur la figure 4.1).

À partir de ce sous-ensemble de connaissances, le système est capable d'apprendre un nouveau savoir. Ce nouveau savoir est ajouté à la base de connaissances. La base de connaissances améliorée permet au système de produire une nouvelle solution candidate pour le problème actuel. Le processus est itéré jusqu'à ce que l'expert valide une solution proposée par les systèmes, c'est-à-dire jusqu'à ce que le système trouve une solution de travail.

4.1.3.1. Formalisme utilisé: logique de description

Nous avons choisi la logique de description (voir figure 4.2) parce qu'elle permet la représentation simultanée et le raisonnement sur la connaissance, tout en réduisant la complexité de l'algorithme par rapport à la logique du première ordre (voir chapitre 3).

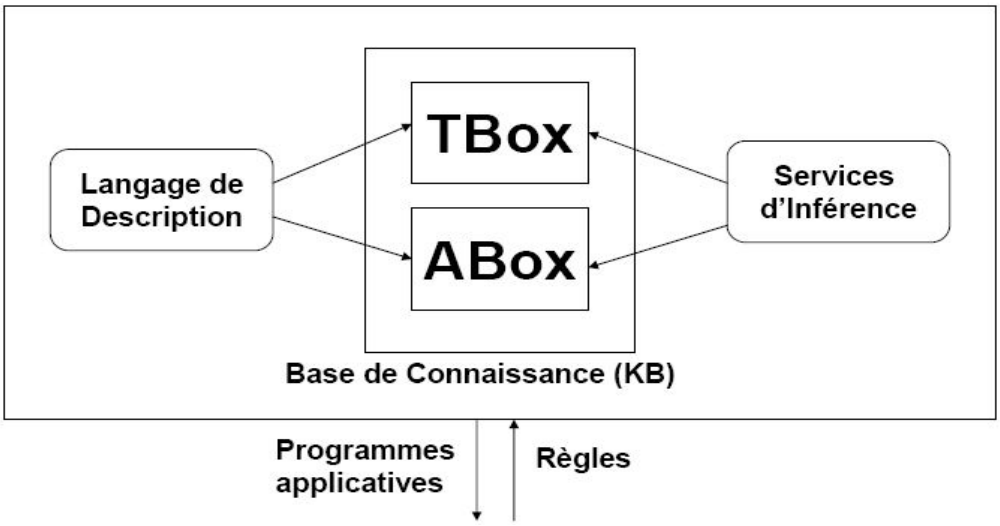


Figure 4.2. Structure générale de système LD

4.1.3.2. Principes de l'adaptation utilisée: L'adaptation conservatrice

La phase d'adaptation dans le cycle du RàPC est le processus proposant une solution à un nouveau problème à partir des solutions appartenant aux cas sources remémorés .

[Fuchs et al, 1999] considèrent l'adaptation comme un plan dont l'état initial est la solution de départ et l'état final est la solution adaptée. [Lieber, 2006] considèrent que la phase d'adaptation consiste à effectuer un raisonnement par analogie : « *sachant que la solution du cas cible est à la solution du cas source ce que le cas cible est au cas source, connaissant le cas source et sa solution ainsi que le cas cible, que vaut la solution du cas cible ?* » L'adaptation termine « l'inférence analogique » en calculant la solution possible au problème du cas cible inspirée de la solution du cas source le plus similaire.

L'étape d'adaptation du raisonnement à partir de cas (RàPC) est réputée difficile à implémenter. Cela, en particulier, parce qu'elle nécessite des connaissances à acquérir, modéliser et représenter.

L'étape d'adaptation d'un système de raisonnement à partir de cas consiste à modifier un cas source en vue de la résolution d'un problème cible.

FrakaS est conçu pour permettre l'acquisition de connaissances dans les systèmes de raisonnement à partir de cas effectuant une adaptation conservatrice. Cette section décrit brièvement les principes de l'adaptation conservative car une compréhension correcte de cette approche d'adaptation est nécessaire pour comprendre comment FrakaS exploite les échecs de raisonnement (et donc les échecs d'adaptation) pour apprendre des connaissances.

L'idée d'adaptation conservatrice est liée à la théorie de la révision des bases de connaissances [Alchourrón et al, 1985] : L'adaptation conservatrice consiste à faire des changements minimales sur le cas source afin d'être cohérent avec à la fois le problème cible et les connaissances du domaine. Le formalisme de cette adaptation est basé sur le concept de l'opérateur de révision (○) lequel associe à deux bases de connaissances Ψ et μ, la base de connaissances Ψ○μ, intuitivement obtenue par changement minimum sur Ψ pour être cohérente avec μ.

4.1.3.3. Adaptation conservative et bases de connaissances RàPC

En RàPC, l'adaptation conservatrice est basée sur trois types de connaissances:
- BCsrce, le contexte lié au problème source et sa solution. C'est la vieille connaissance, il peut être modifié mais doit être modifié au minimum.
- BCtgt, le contexte lié au problème cible. C'est la nouvelle connaissance qui ne doit pas être modifié pendant le processus d'adaptation.
- CSD, la connaissance du domaine. C'est la connaissance qui doit être vraie dans n'importe quel contexte.

L'adaptation conservative consiste à effectuer un déplacement du contexte source vers le contexte cible afin de trouver une solution au problème cible, en restant cohérent avec le problème cible et la connaissance du domaine. Ce processus est conservateur parce que le changement fonctionne avec un changement minime tout en restant cohérent avec la définition du problème cible. En effet, le contexte du problème cible ne peut pas être modifié.

4.1.3.4. Le rôle de l'opérateur de révision dans le processus d'adaptation.

Comme il a été expliqué précédemment, la révision d'une base de connaissances Ψ (les anciennes connaissances) par une base de connaissances μ (les nouvelles connaissances) produit une base de connaissances Ψ○μ (la base de connaissances révisée). La théorie de la révision

garantit à la fois la cohérence de la base de connaissances révisée et un ensemble d'autres propriétés liées à la théorie elle-même. La figure 4.3. illustre ce principe.

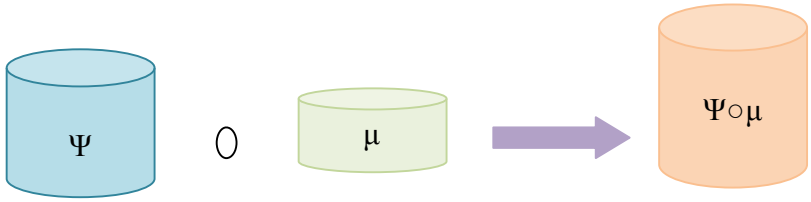


Figure 4.3. Illustration de la théorie de la révision

Le processus RàPC implémenté dans FrakaS exploite une base de cas avec la base de connaissances du domaine du système (indiquée par CDS). Les cas contenus dans la base de cas sont supposés être compatibles avec CDS, ils contiennent souvent des éléments de connaissance provenant de l'expérience. Ces connaissances ne peuvent pas toujours être expliquées par la connaissance du domaine mais sont néanmoins souvent très utiles, c'est pourquoi elles sont précieuses. La figure 4.4. détaille comment l'adaptation est effectuée à l'aide de l'opérateur de révision dans FrakaS. Afin de résoudre un nouveau problème, FrakaS récupère un cas qui est, bien sûr, cohérent avec CDS. Ensuite, le cas récupéré est adapté, en utilisant l'opérateur de révision, pour proposer une solution au problème actuel. L'opérateur de révision est appliqué sur deux bases de connaissances: la première comprend le cas source extrait avec CDS et l'autre le problème cible, ainsi que le CDS. Le résultat de l'adaptation produit une solution compatible avec CDS.

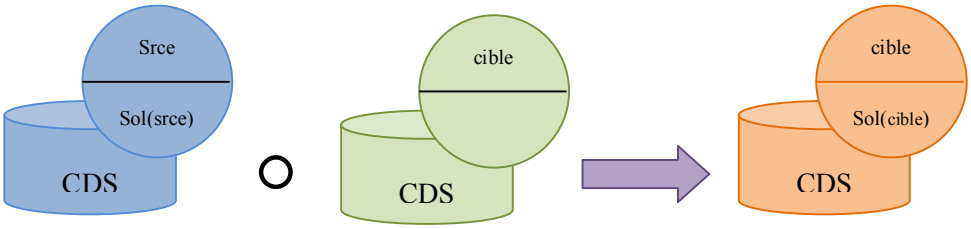


Figure 4.4. Adaptation par révision à FrakaS

4.1.4. Résultats et discussion

L'acquisition de connaissances d'adaptation ce fait par :

- à partir de la base de cas
- par des techniques d'extraction de connaissances

- de manière interactive et opportuniste

Dans notre cas l'acquisition des ces connaissance ce fait d'une manière interactive.

Dans la logique de description, l'adaptation d'un cas source pour résoudre un problème cible est plus complexe. Parce que nos bases de connaissances contiennent deux parties: le TBox et l'ABox, donc il faut appliquer le principe de cette adaptation aux deux parties (c.-à-d. : TBox et ABox). [Hioual et al, 2013].

Le problème de la révision d'une base de connaissance est étroitement lié au problème de la révision des croyances qui a été largement discuté dans la littérature. Certains opérateurs de révision de croyance basés sur la syntaxe ont été adaptés pour réviser les bases de connaissances dans la logique de description. Cependant, ces opérateurs suppriment l'ensemble des axiomes pour résoudre les contradictions logiques et ne sont donc pas affinés.

Dans notre travail, nous considérons uniquement la révision des terminologies dans les logiques de descriptions.

Un opérateur de révision doit satisfaire tous les postulats AGM. L'AGM (développé par Alchourrón, Gärdenfors et Makinson) détermine le comportement d'une fonction de changement c'est à dire un ensemble de conditions ou contraintes que les fonctions de changement doivent satisfaire ou un ensemble de propriétés qu'un opérateur qui effectue la révision doit satisfaire afin d'être considéré comme rationnel [Hioual et al, 2013].

Dans les travaux antérieurs, ils ont principalement utilisé l'opérateur de révision de Dalal [M. Dalal, 1988]. qui est un opérateur important basé sur un modèle indépendant des formes syntaxiques de la base de connaissances à réviser, et ce dernier satisfais tous les postulats de l'AGM, il est donc nécessaire ici d'adapter cet opérateur à la révision des terminologies dans notre travail.

Le principe de l'opérateur de révision de Dalal est que les modèles de la base de connaissances révisée par l'opérateur doivent être des modèles de la base de connaissances nouvellement reçue et qui ont une distance minimale à partir de la base d'origine.

Un opérateur de révision basé sur un modèle dans les logiques propositionnelles est généralement défini par l'ensemble de différences symétriques entre deux interprétations qui sont un ensemble de variables propositionnelles. Cependant, il n'est pas normal d'adapter les opérateurs de révision basés sur un modèle aux DL car les DL ont leurs propres caractéristiques [G. Flouris et al,2005]

En plus de l'opérateur de Dalal, nous définissons la distance entre deux interprétations comme suit :

Definition1 :

(Distance entre les interprétations) Soit I = (Δ, $\cdot^I$) et I' = (Δ, $\cdot^{I'}$) deux interprétations sur le même domaine. Soit $d(M^I,M^{I'}) = |M^I \ominus M^{I'}|$ où M est un nom de concept ou un nom de rôle. La distance entre I et I ', notée d (I, I'), est définie comme suit:

$$d(I,I')=\sum_{A\in L_C} |A^I \ominus A^{I'}| + \sum_{R\in L_R} |R^I \ominus R^{I'}|$$

où $(A^I \ominus A^{I'})$ dénote la différence symétrique entre les ensembles A^I et $A^{I'}$, c-à-d la différence symétrique de deux ensembles est l'ensemble des éléments qui sont dans l'un des ensembles et non dans leur intersection, L_C et L_R sont respectivement les ensembles de tous les noms de concept et noms de rôle.[Fangkai Yang, 1999]

Dans notre travail, nous avons choisi l'opérateur de révision défini dans les travaux de Guilin Qi et Du Jianfeng [Guilin Qi et al, 2009].

Definition2 :

Soit T1 et T2 deux TBox. La distance entre T1 et T2, notée d (T1, T2), est définie par:

$d(T1, T2) = \min_{I\models T1, I'\models T2} |diff(I, I')|$ (|u| cardinalité moyenne de l'ensemble u) La distance entre deux TBoxes T1 et T2 est la cardinalité minimale des ensembles de différences entre les modèles de T1 et les modèles de T2.

Definition3 :

Soit T1 et T2 deux TBox. Un opérateur de révision, noté $\circ$M, est défini de la façon suivante:

$Mod(T1 \circ M T2) = \{I \models T2 \mid \exists I' \models T1, |diff(I, I')| = d(T1, T2)\}$.

les modèles du résultat de notre opérateur de révision $\circ$M sont les modèles de TBox T2 satisfaisant la condition qu'il existe un modèle de T1 tel que la différence entre eux soit égale à la distance entre les deux TBoxes .

L'opérateur $\circ$M vérifie tous les postulats des opérateurs de révision dans les DLs , qui ont été reformulés dans les postulats de Katsuno et de Mendelzon (postulats de KM) [Flouris et al, 2005]

(G1) $Mod(T1 \circ T2) \subseteq Mod(\varphi)$ pour tout $\varphi \in T2$.

(G2) Si $Mod(T1) \cap Mod(T2) \neq \emptyset$, alors $Mod(T1 \circ T2) = Mod(T1) \cap Mod(T2)$.

(G3)Si $T2$ est consistent, alors $Mod(T1 \circ T2) \neq \emptyset$.

(G4) Si $Mod(T) = Mod(T1)$ et $Mod(T') = Mod(T2)$, alors $Mod(T \circ T') = Mod(T1 \circ T2)$.

(G5) $Mod(T1 \circ T2) \cap Mod(T3) \subseteq Mod(T1 \circ (T2 \cup T3))$.

(G6) Si $Mod(T1 \circ T2) \cap Mod(T3)$ n'est pas vide, alors $Mod(T1 \circ (T2 \cup T3)) \subseteq Mod(T1 \circ T2) \cap Mod(T3)$.

(G1) garantit que chaque axiome du nouveau TBox peut être déduit du résultat de la révision.

(G2) dit que nous ne changeons pas la base de connaissances originale s'il n'y a pas de conflit.

(G3) est une condition empêchant une révision d'introduire une incohérence injustifiée.

(G4) dit que l'opérateur de révision devrait être indépendant des formes syntaxiques des bases de connaissances.

(G5) et (G6) ensemble sont utilisés pour assurer un changement minime [H. Katsuno et al, 1992]

4.2. La mémoire d'entreprise et les connaissances du domaine

La mémoire d'entreprise contient des connaissances du domaine de l'entreprise ainsi que différentes sources de connaissances basées sur une ontologie commune. Cette ontologie permet d'utiliser les connaissances stockées dans cette mémoire par les différentes classes d'acteurs et selon leurs différentes tâches à faire.

A côté de l'automatisation et de l'informatisation des équipements industriels les entreprises d'aujourd'hui demandent la préservation et la capitalisation du savoir faire des experts. Un aspect important est donc, en plus d'une bonne gestion du personnel, de mettre en place un système permettant de fournir à un individu l'information utile au moment où il en a besoin, dans les meilleurs délais, et de façon exploitable. La gestion des connaissances est devenue un enjeu stratégique capital pour de nombreuses entreprises pour répondre à un environnement économique, technique et informationnel en perpétuelle évolution. Les entreprises doivent être capables de développer des qualités d'adaptation permanente par le développement des compétences de leurs ressources humaines.

La mise en place d'une ME peut consister à rédiger des documents écrits explicitant des connaissances tacites (le modèle le plus simple tel que des fiches mémoire), ou à élaborer cette mémoire à l'aide de support informatique, aboutissant ainsi à un « système documentaire intelligent » , à des bases de connaissances exploitées par un SBC [Simon, 1996], à des système multi-agents , exploitant Internet (modèles les plus complexes). Cette liste n'est pas exhaustive,

mais elle est assez représentative du nombre de techniques et de points de vue différents qui règnent dans le domaine.

Un système de raisonnement à partir de cas est un système qui effectue des raisonnements et, par conséquent, qui s'appuie sur des connaissances. Et D'après [Dieng et al, 1999]. "Une mémoire d'entreprise est la représentation persistante, explicite, désincarnée, des connaissances et des informations dans une organisation, afin de faciliter leur accès, leur partage et leur réutilisation par les membres adéquats de l'organisation, dans le cadre de leurs tâches".

La capitalisation des connaissances dans une organisation a pour objectif de favoriser la croissance, la transmission et la conservation des connaissances dans cette entreprise. La gestion explicite des savoirs et savoir-faire occupe une place de plus en plus importante dans les organisations, ainsi la construction de mémoires d'entreprise, dans un but de préservation et de partage, est devenue une pratique assez courante. Plusieurs méthodes ont été adaptées de l'ingénierie des connaissances à cet effet.

La gestion des connaissances se réfère à la gestion de tous les savoirs et savoir-faire en action mobilisés par les parties prenantes de l'entreprise pour lui permettre d'atteindre ses objectifs. Plusieurs étapes ont été identifiées dans un processus de gestion des connaissances (capitalisation et partage des connaissances), l'explicitation des savoirs tacites identifiés comme cruciaux pour l'entreprise, le partage des savoirs explicites sous forme de Mémoire et l'appropriation et l'exploitation des parties de cette connaissance par les acteurs de l'entreprise.

Le partage et l'appropriation de mémoires d'entreprise sont toujours de véritables goulots d'étranglement au sein des organisations. Les méthodes de gestion des connaissances ne sont pas suffisantes pour permettre une appropriation effective des connaissances par les parties prenantes de l'entreprise. Cependant, l'objectif de la capitalisation des connaissances est le partage et la réutilisation de l'expérience afin d'optimiser le processus d'apprentissage organisationnel.

Aujourd'hui, les techniques de modélisation des connaissances sont relativement bien maîtrisées et ont été largement utilisées. Parmi celles-ci figurent des techniques basées sur des représentations logiques, sur l'utilisation de réseaux sémantiques, de réseaux neuronaux artificiels, sur l'écriture de règles, ou sur des mélanges de ces différentes approches. Il existe plusieurs méthodes et outils sur le marché pour gérer ces différents types de représentation des connaissances.

Dans cette partie, nous nous intéressons à la capitalisation des connaissances d'entreprise en utilisant la méthode REX (Retour d'EXpérience). L'objectif de cette méthode est de capitaliser les connaissances qui doivent être stratégiques et favoriser le retour d'expérience tout en répandant aux quatre points suivants :

- la valorisation de l'expérience,
- l'extraction des connaissances,
- la constitution d'une mémoire d'entreprise
- la mutualisation des expériences.

4.3. Mémoire d'entreprise

Bien que le terme "mémoire d'entreprise" soit nouveau dans le paysage des entreprises, ses origines remontent à la fin du 19ème siècle [Lehner et al, 1998] au début du 20ème siècle, ce terme a été oublié pour réapparaître en 1981. Dans [Hedberg et al, 1981] la première définition d'une mémoire organisationnelle en ce sens que c'est une mémoire qui «établit les structures cognitives du traitement de l'information au sein d'une entreprise».

La mémoire d'entreprise (ME) est donc un recueil des connaissances, les trois définitions suivantes sont exhaustives et méritent d'être citées :

« La mémoire d'entreprise est la présentation persistante, explicite, désincarnée des connaissances et des informations dans une organisation » [Van Heijst et al, 1996].

« C'est l'ensemble de savoirs et savoir-faire en action, mobilisés pas les employés d'une entreprise pour lui permettre d'atteindre ses objectifs (produire des biens ou des services) » [Barthès et al, 1999].

« Elle se caractérise par la volante de préserver pour les réutiliser par la suite ou le plus rapidement possible les raisonnements, les comportements, les savoirs, les connaissances jusque dans leur contradiction et dans leurs diversités » [Pomain, 1996].

La construction d'une mémoire d'entreprise repose sur la volonté de «préserver les raisonnements, les comportements et les connaissances afin de les réutiliser plus tard ou le plus tôt possible, même dans leurs contradictions et dans toute leur diversité»

Il existe plusieurs types de connaissances dans une organisation: la connaissance explicite et la connaissance tacite. Par conséquent, pour toute opération de capitalisation des connaissances, il est important d'identifier les connaissances stratégiques à capitaliser.

4.3.1. Cycle de capitalisation des connaissances

Un des modèles se rapprochant du modèle générique et semblant le plus approprié à nos objectifs, est celui proposé par Grundstein et que nous présentons à la figure 4.5. Ce cycle de capitalisation des connaissances est managé par la gestion des connaissances. Il est composé de quatre phases de repérage, préservation, valorisation et actualisation et reflète également les besoins du développement d'un système d'aide à la décision, chacune déclinée sur plusieurs tâches.

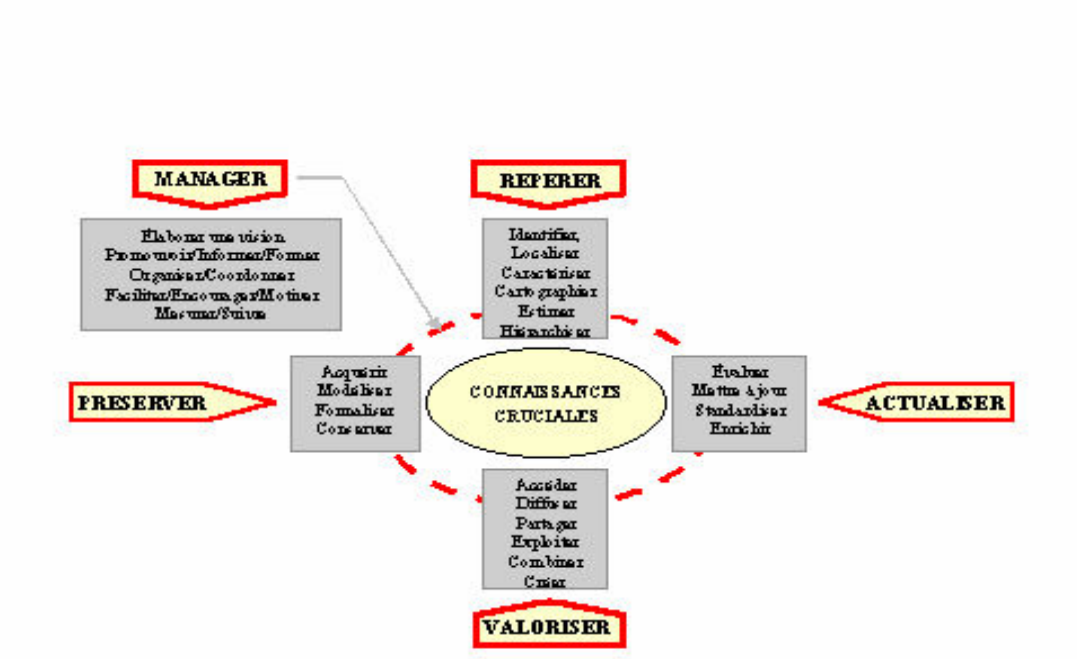


Figure 4.5. Cycle de capitalisation des connaissances tiré de [Grundstein, 2000]

Le processus de capitalisation comporte plusieurs étapes axées sur la notion de savoir-faire stratégique (voir la figure 4.5):

- Le processus de localisation: localisation des connaissances critiques, c'est-à-dire des connaissances explicites et des connaissances tacites nécessaires aux processus de décision et au progrès des processus essentiels qui constituent le cœur des activités de l'entreprise.
- Le processus d'Actualisation: actualisation du savoir-faire et de la compétence: il est nécessaire de les évaluer, de les mettre à jour, de les normaliser et de les enrichir en

fonction des retours d'expérience, de la création de nouvelles connaissances et la contribution des connaissances externes.

- Le processus de valorisation: valeur ajoutée du savoir-faire et de la compétence: il est nécessaire de les rendre accessibles selon certaines règles de confidentialité et de sécurité, de les diffuser, de les partager, de les exploiter, de les combiner et de créer de nouvelles connaissances.
- Le processus de préservation: préservation du savoir-faire et de la compétence, lorsque la connaissance est explicable, il est nécessaire de l'acquérir, de la modéliser, de la formaliser et de la préserver; lorsque la connaissance n'est pas explicable, le transfert de connaissances et de réseaux de communication maître-apprenti entre les personnes, par exemple, devrait être encouragé.
- Le processus de gestion: les interactions entre les différents problèmes mentionnés ci-dessus. C'est là que la gestion des activités et des processus est positionnée pour amplifier l'utilisation et la création de connaissances dans les organisations.

En résumé, "Capitaliser les connaissances de l'entreprise" consiste à repérer ses connaissances cruciales, à les préserver et les pérenniser tout en faisant en sorte qu'elles soient partagées et utilisées par le plus grand nombre au profit de l'augmentation de richesse de l'entreprise ".

Plusieurs méthodologies sont conçues pour la construction de la «mémoire d'entreprise» définie comme «une représentation explicite, désincarnée et persistante de la connaissance et de l'information dans une organisation». Dans la section suivante, nous présentons la méthode choisie pour la construction de la d'entreprise.

4.3.2. Méthode REX (retour d'expérience)

La méthode REX est due à la capitalisation des expériences archivées lors de la réalisation de l'activité de l'entreprise. La finalité est de construire une base de connaissances pour les futures équipes. Le but de cette méthode consiste à constituer des "éléments d'expérience" extraits d'une activité et à renvoyer ces éléments à un utilisateur pour les valoriser. Trois types d'éléments d'expérience existent: des éléments de connaissances documentés, des éléments d'expérience issus d'entretiens avec des experts et des éléments de savoir-faire issus d'une activité particulière [Malvache et al, 1993]

Le principe de base de la méthode consiste à constituer des «éléments d'expériences», extraits d'une activité quelconque et à restituer ces éléments pour qu'un utilisateur puisse les valoriser.

Les éléments d'expérience ainsi définis sont stockés dans une mémoire d'expérience appelée (CEMem) avant d'être restitués (voir Figure 4.6).

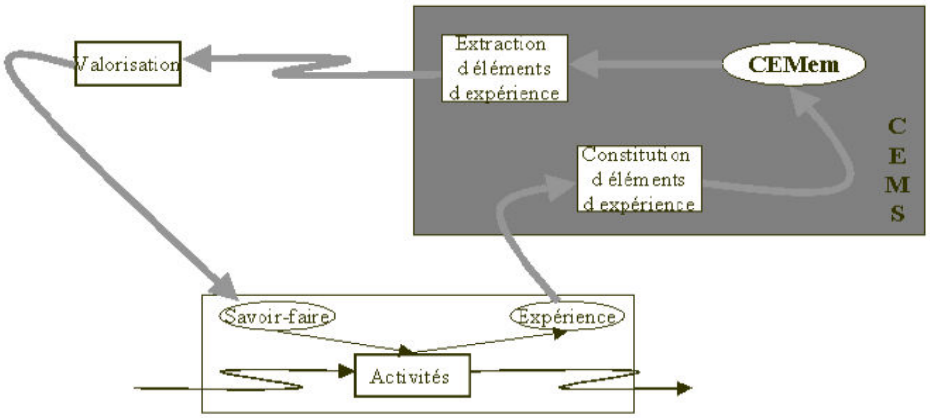


Figure 4.6. Principe de base de la méthode REX tiré de [Matta Nada, 1999]

La méthode REX est divisée en trois étapes qui sont :

1- Analyse des besoins et identification des sources de connaissances (spécification et dimensionnement du futur système de gestion des connaissances): Il est nécessaire de prendre en compte les besoins de l'organisation; ses objectifs sont de construire une représentation de la situation existante en modélisant les principales activités du domaine concerné. Cette modélisation permet d'identifier les flux de connaissances entre ces activités. Il est donc nécessaire de dimensionner le futur système de gestion des connaissances en analysant les sources de connaissance de l'organisation (liste des fonds documentaires ...) puis en estimant les éléments de connaissances qui peuvent être produits à partir de ces sources. Il est également nécessaire d'identifier les spécialistes dans le domaine et d'identifier les actions de valorisation à mener pour faciliter la capitalisation et le transfert des connaissances.

2- Construction et stockage d'éléments de connaissances sous la forme d'une base de données ou d'une mémoire d'entreprise.

3- La création et le fonctionnement du système de gestion des connaissances sont créés.

4.3.2.1. Principes et objectifs de la méthode

REX est une méthode de capitalisation des connaissances.

Capitaliser les connaissances implique la constitution d'un capital intellectuel qui sera ensuite valorisé. On ne peut pas capitaliser toutes les connaissances et donc il est nécessaire de ne considérer que les connaissances stratégiques ou cruciales pour certaines activités. Le processus de capitalisation des connaissances permet de réutiliser, de façon pertinente, les connaissances d'un domaine donné, précédemment stockées et modélisées, afin d'accomplir de nouvelles tâches [Simon, 1996].

L'objectif premier de la méthode REX est donc de capitaliser les connaissances et favoriser le retour d'expérience. Le retour d'expérience se présente comme la description structurée, sous forme de fiches d'expérience (voir figure 4.7).

Entête

Nom: Traversée d'un carrefour
Origine: Expert1, Référence entretien N.3
Auteur: Cogniticien1
Date d'émission: Janvier 1996
Domaine: Psychologie
Contexte: Analyse de stratégies adoptées par un conducteur pour traverser un carrefour

Corps

Observation: Le choix de la stratégie (traverser en une seule fois, versus, traverser en plusieurs fois) dépend d'un certain nombre d'attentes liées au volume du trafic. Si le trafic est faible, il y a de très grandes chances que le conducteur n'ai pas besoin de s'arrêter au milieu. Donc, quelqu'un qui arrive et voit que le trafic est faible, pense n'avoir pas besoin de s'arrêter au milieu et choisit la stratégie de traverser sans arrêt. Et comme il s'attend à pouvoir traverser sans arrêt, on peut faire l'hypothèse qu'il est préparé à ce qu'il n'y ait personne.
Hypothèse: trafic faible implique une stratégie de traversée sans arrêt au milieu.
Trafic dense implique une stratégie de traversée avec arrêt au milieu.
Commentaires: Les attentes du conducteur sont différentes selon la stratégie choisie: - Stratégie de traversée en deux coups: le conducteur simplifie le problème en s'occupant d'abord du flux de gauche, puis du flux de droite.
- Stratégie de traversée en un coup: la solution la plus difficile. La plus dure, parce qu'il faut évaluer le créneau, simultanément de chaque côté, avec des durées de validité d'informations qui se conditionnent mutuellement, et qui supposent que l'on va même faire vite pour traverser.

Figure 4.7. Exemple d'un élément d'expérience dans le domaine de l'analyse d'accident tiré de [Simon, 1996]

a) Eléments d'expérience

Un élément d'expérience (voir Figure 4.8) est typiquement défini par:

- Un entête,
- Une description ou corps,
- Une liste de références.

Le corps est décomposé lui même en trois parties:

- Une description neutre d'un fait,
- Une opinion propre et des commentaires,
- Des recommandations

Le premier entretien est mené d'une manière libre et vise à identifier les personnes concernées par un thème particulier et à collecter leur avis. Le texte recueilli à l'issue de cet entretien sert à identifier plusieurs éléments d'expérience correspondant aux différents faits cités.

Au cours du deuxième entretien, un ensemble provisoire d'éléments d'expérience est présenté à l'expert qui peut modifier leur contenu et les enrichir. L'objectif du troisième entretien est de vérifier si toutes les modifications apportées aux éléments d'expérience ont été considérées. D'éventuelles corrections sont alors introduites.

Les éléments d'expérience sont construits principalement à l'issue des entretiens auprès d'experts et à partir des documents relatant une activité (i.e. documents de synthèse, bases de données). Un questionnaire doit être élaboré. Il forme une base pour le cogniticien dans un entretien avec un expert. Il permet d'élucider les éléments d'expériences et leur description. Trois entretiens d'une demi-journée chacun sont recommandés à ce propos.

a) Mémoire d'expérience

Dans le but de considérer la diversité de vocabulaire et les points de vue utilisés dans une entreprise, un modèle descriptif et un réseau terminologique sont également définis pour constituer une mémoire d'expérience.

b) Le modèle descriptif

Le modèle descriptif permet de représenter les différents points de vue identifiés dans une entreprise. Généralement, une douzaine de points de vue semble raisonnable, au delà de douze, le réseau défini par ces points de vue sera inexploitable. Par exemple, nous pouvons distinguer point de vue géographique, topologique, etc. dans une activité de conception.

Chaque point de vue est représenté par un réseau d'objets (définis sous forme de concepts), reliés entre eux, suivant un réseau sémantique. Un ensemble de catégories de liens est aussi défini

comme: «ensemble/élément», «général/ spécifique», «proximité», «Self évolution». Le modèle descriptif peut ne pas être défini d'une manière exhaustive. Il sera enrichi au fur et à mesure.

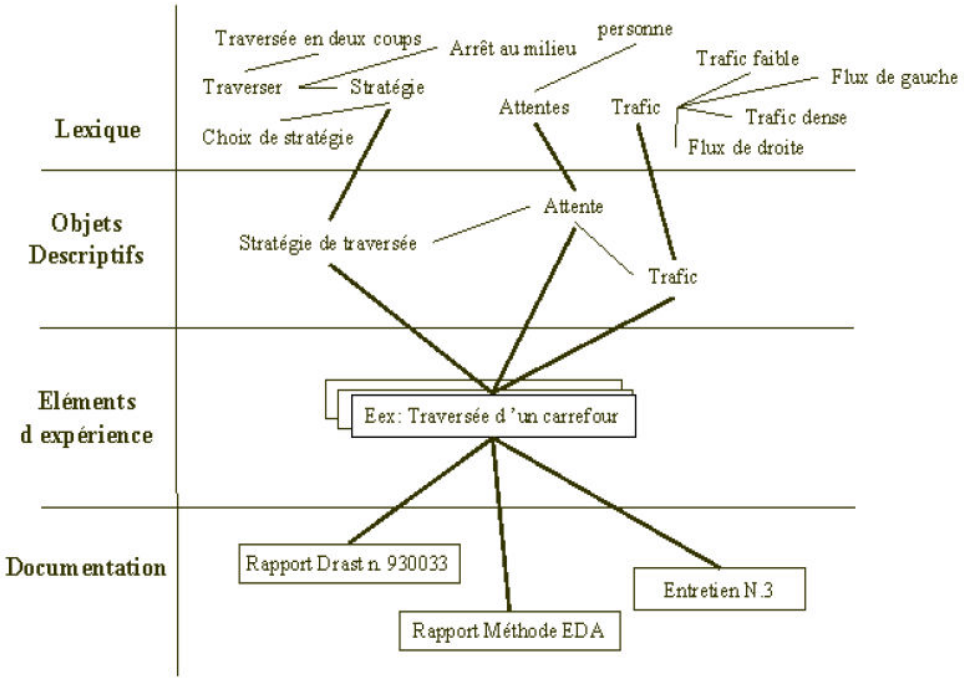


Figure 4.8. Modèle d'une mémoire d'expérience tiré de [Simon, 1996]

Ce modèle est constitué de quatre parties : réseau terminologique ou lexique, modèle descriptif, éléments d'expérience et documents.

c) Le réseau terminologique

Un réseau terminologique nommé aussi lexique est construit pour permettre des requêtes proches de la langue naturelle. Ce réseau est constitué d'objets qui peuvent être des mots ou des phrases nominales, appartenant au vocabulaire du domaine considéré. Le réseau est structuré avec des relations syntaxiques de type "sorte-de" et "concerne".

Un élément d'expérience est considéré comme élémentaire dans la mémoire. Il est rattaché à un ensemble d'objets définis dans les points de vue. Cette opération peut être automatique. Elle est basée sur une reconnaissance lexicale des termes identifiés dans le texte de l'élément d'expérience. Mais le choix final du lien à établir est laissé au cogniticien. Ce type d'association permet une vue descriptive du domaine. La représentation textuelle d'un élément d'expérience

peut être indexée automatiquement en reliant les termes identifiés dans le texte au réseau terminologique défini. La Figure 4.8 montre un modèle d'une mémoire d'expérience.

Un outil informatique est associé à la méthode REX. Il permet d'une part de mémoriser les éléments d'expérience et fournit d'autre part, une interface d'accès à ces éléments. L'interface du système est conçue de manière à permettre des requêtes en langage naturel. La requête ainsi formulée sera analysée par le système, qui en réponse, propose à l'utilisateur un ensemble d'objets candidats correspondant aux termes identifiés dans le réseau terminologique. La description d'un objet choisi, correspondant à sa description dans le modèle descriptif, ainsi que les éléments d'expérience qui y sont rattachés, sont alors présentés.

Le REX peut répondre à quatre objectifs différents et complémentaires :

- Valorisation de l'expérience,
- Extraction des connaissances,
- Constitution d'une mémoire d'entreprise,
- Mutualisation des expériences.

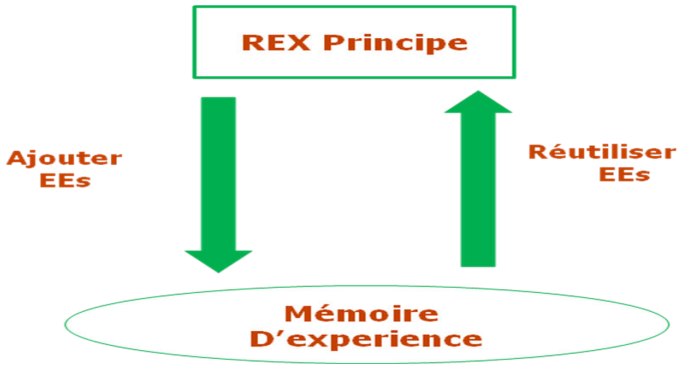


Figure 4.9. REX principe

4.3.3. Partie Implémentations

Dans cette partie en va présenter la méthode choisie REX et cette dernière sera utilisé dans le domaine médicale. Ce domaine a une large application dans différentes organisation tel que Les Hôpitaux. [Hioual et al, 2017]

Le principe de base de cette méthode repose sur la construction des éléments d'expériences et la mémoire d'expériences.

Nous avons divisé notre travail en 2 parties : une pour la collection et l'indexation des expériences et l'autre pour l'accès et l'exploitation [Hioual et al, 2017].

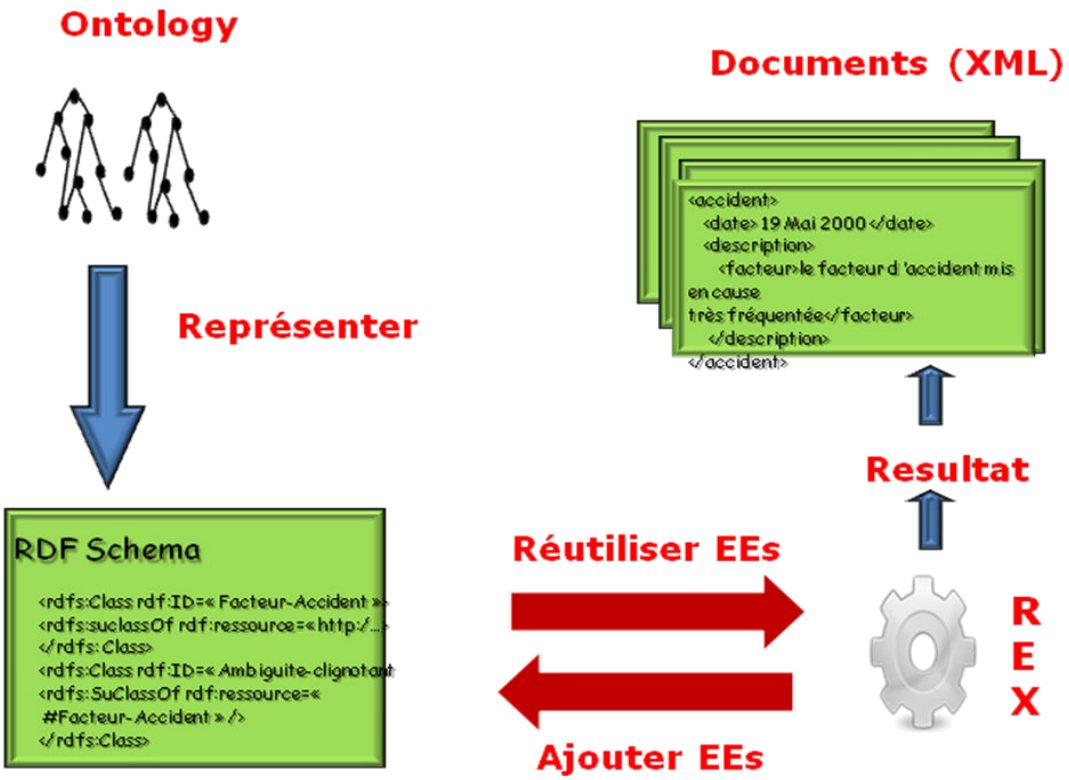


Figure 4.10. Exemple de l'implémentation

4.3.3.1. Collection et indexation des expériences

Cette partie s'occupe de l'enregistrement des comptes rendus des expériences achevés en soignions un accès efficaces par des fichiers d'indexation, ce la veut dire créer des fichiers sous forme RDF qui contiendrons des mots clés qui ont des liens sémantique par apport au contenu des expériences enregistrés.

Un fichier RDF : **Resource Description Framework** est un modèle de graphe destiné à décrire de façon formelle les ressources Web et leurs métadonnées, de façon à permettre le traitement automatique d'un telles descriptions. Développé par le W3C, RDF est le langage de base du Web sémantique. RDF est un modèle, associé à une syntaxe, dont le but est de permettre à une communauté d'utilisateurs de partager les mêmes métas donnés pour des ressources partagées.

En annotant des documents non structurés et en servant d'interface pour des applications et des documents structurés (par exemple bases de données, GED, etc.) RDF permet une certaine interopérabilité entre des applications échangeant de l'information non formalisée et non structurée sur le Web

Exemple d'index :

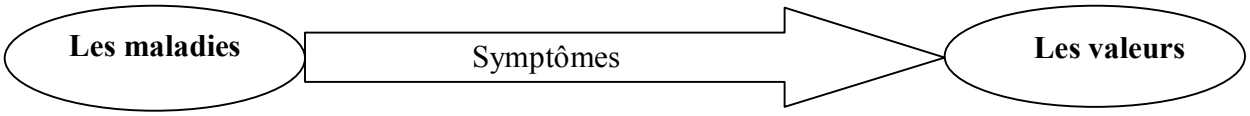


Figure 4.11. Exemple index [Hioual et al, 2017]

Exemple : Code source (Voir annexe B)

4.3.3.2. Accès et exploitation des expériences

Cette partie s'occupe de l'accès aux comptes rendu des expériences emmagasinés on utilisant une recherche sémantique. [Hioual et al, 2017].

Les comptes rendus contiendront des observations faites par des chercheurs sur des maladies :

- ✓ Le compte rendu est enregistré dans un fichier texte portent le nom de la maladie sujet d'étude.
- ✓ Le fichier index (RDF) contient des mots clés qui ont des liens directs ou une ressemblance sémantique dans la description de la maladie (des symptômes)
- ✓ La recherche : s'effectue avec le moteur de recherche SPARQL.
- ✓ Ouvrir le fichier qui correspond plus aux mots clés ajoutés à la recherche.

4.4. La capitalisation des connaissances en informatique

Il existe trois approches principales de la capitalisation des connaissances en informatiques :

- Approche gestion de l'information (Systèmes d'information) relève du management de l'information.
- Approche gestion des connaissances (Systèmes à base de connaissance - Systèmes experts).
- Approche unifiée (Système de capitalisation des connaissances).

Dans cette partie, nous nous intéressons à la capitalisation des connaissances en informatique en utilisant l'approche gestion des connaissances (Systèmes Expert).

En général, les concepteurs de systèmes experts effectuent l'*acquisition de connaissance* grâce à un ou plusieurs interviews avec l'expert ou les experts du domaine. Les humains qui enrichissent le système avec leurs connaissances ne fournissent pas seulement leur connaissance théorique ou *académique* mais aussi des heuristiques qu'ils ont acquises grâce à l'utilisation de leurs connaissances.

Contrairement à la modélisation cognitive, les systèmes experts n'ont pas comme finalité de s'inspirer des théories du fonctionnement du cerveau humain mais ce sont des programmes qui utilisent des stratégies heuristiques pour la résolution des problèmes spécifiques. Le raisonnement effectué par un système expert doit être objet à l'inspection, et ceci en fournissant l'information sur l'état de la résolution du problème et des explications sur les choix et les décisions du système. D'un autre côté, la solution fournie par le système doit être évaluée par un expert humain et ceci dans le but de modifier l'information contenue dans la base de connaissances.

L'étude décrite dans cette partie est une implémentation d'un petit système expert pouvant être utilisé dans plusieurs domaines. Notre système est composé principalement d'une base de connaissances qui est l'élément important d'un système expert. Cette dernière est divisée en deux bases: une base de faits contenant les connaissances factuelles ou des faits spécifiques du domaine, et une base de règles permettant de spécifier le comportement du système. Ces règles sont écrites dans un langage structuré. Nous utilisons le moteur d'inférence chainage avant qui permet d'inférer de nouvelles connaissances à partir de la base de connaissances du système parmi les différents modes d'invocation des règles. Dans notre système générique, nous utilisons une base de faits et une base de règles avec un seul fait.

4.4.1. Implémentations d'un petit système expert

Comme nous l'avons déjà dit, interpréteur des règles ou mécanisme d'inférence, celui-ci est la composante active d'un système expert, qui effectue la sélection et l'application des règles en vue de la résolution d'un problème donné. Est un programme chargé d'exploiter la base de connaissances du système définie précédemment, [Balouchi et al, 2015].Et qui construit automatiquement le raisonnement du système en sélectionnant et en activant les règles pertinentes de la base de connaissances, en fonction de la base de faits. Le moteur d' inférence est en fait un interpréteur de connaissances, physiquement indépendant de la base de connaissances.

Il est le cœur du système expert : «moteur» parce qu'il permet de déclencher tous les rouages du système, « inférence» parce que la construction dynamique de la raisonnement s'opère par des inférences successives, à partir de connaissances disponibles dans la base de faits .

Selon le système de production choisi, différents modes d'invocation des règles existent. En chaînage avant, chaînage arrière ou bien chainage mixte . De façon générale, le moteur d'inférence sera capable de répondre à des questions, de raisonner et de tirer les conséquences impliquées par la connaissance incluse dans le système.

Un système expert est composé de deux parties indépendantes :
– **une base de connaissances** elle-même composée d'une **base de règles** qui modélise la connaissance du domaine considéré et d'une **base de faits** qui contient les informations concernant le cas que l'on est en train de traiter
– **un moteur d'inférences** capable de raisonner à partir des informations contenues dans la base de connaissance, de faire des déductions, etc.

4.4.2. Moteur d'inférence choisie : chainage avant

Nous choisissons le chaînage avant pour implémenter notre système expert, En chaînage avant, Le mécanisme du chaînage avant est très simple : pour déduire un fait particulier, on déclenche une règle dont les prémisses sont connues dans BF. Sa conclusion est ajoutée à BF comme fait connu. La règle est ensuite désactivée (une règle n'a besoin d'être déclenchée qu'une fois au plus, puisque ses prémisses restent dans BF). L'algorithme tourne ainsi jusqu'à ce que le fait à déduire soit connu ou qu'aucune règle ne soit plus déclenchable. On a choisie ce moteur d'inférence parce que il est le plus simple et moins couteux (voir annexe C).

4.4.3. Fonctionnalité

Le moteur d'inférence ou encore système cognitif qui contrôle l'activité du système par un cycle à trois phases.

La première phase : Cette étape s'appelle le filtrage et recherche, c'est la sélection d'un ensemble de règles dont la condition est satisfaite par l'état actuel de l'espace de travail. s'il existe un jeu de substitution de variables rendant deux formules logiques identiques.

La deuxième phase : C'est la résolution des conflits. Consiste à choisir parmi les règles exécutables obtenues à l'étape précédente, un sous-ensemble restreint d'une ou plusieurs règles qui seront effectivement exécutées. [Pierre Ardiri , 2012].

la troisième phase : c'est généralement l'introduction ou la suppression d'informations dans l'espace de travail provoqué par l'exécution proprement dite des règles choisies à la phase deux .

Stratégie de recherche d'une règle

- Stratégie basée sur la complexité: les règles dont le nombre de ces faits/conditions est le plus grand est les plus prioritaires.
- Stratégie basée sur la simplicité: les règles dont le nombre de leurs faits/conditions est le plus petit est les plus prioritaires.
- Stratégie en profondeur d'abord.
- Stratégie en largeur d'abord.
- Dans l'ordre d'écriture dans la base ;
- Au hasard ;
- Choix de la règle concluant sur le but

4.4.4. Méthode de travail

Dans notre travail nous appliquons cet algorithme sur une base de connaissance qui contient tous les faits et les règles dont un expert humain aurait besoin.

La base de connaissance est constituée de deux sous-composantes qui sont: La base de faits et la base de règles

Les faits peuvent être des faits entrants ou des nouveaux faits, c'est -à-dire des faits qui sont inférés à partir de l'application des règles.

- Au début on demande a l'utilisateur de faire entrer la base de fait :

La base de fait contient les connaissances factuelles ou des faits spécifiques du domaine. les faits concernant un sujet précis, appelé « domaine ». Les faits comprennent des définitions, des associations, des mesures, des probabilités, des observations, des contraintes ou bien des hypothèses.

Code source : la base de fait (voir annexe C)

- On a déclaré un tableau **BF** de type « string » qui demande a l'utilisateur d'entré le nombre de fait. Ensuite l'utilisateur remplie le tableau avec les faits .

Après on a criée un autre tableau **BF1** et on fais un test sur les fait entrer par l'utilisateur dans le premier tableau s'il y a pas un fait redoublé on le rajoute dans le deuxième tableau si non, s'il y a des redoublons on ne fait rien .

- Après on demande a l'utilisateur la 2éme étape, le remplissage de la base des règles :

La base de règles permet de spécifier le comportement du système. Ces règles sont écrites sous formes de langage structuré, correspondent au langage de programmation qui permettra au moteur d'inférence de calculer ou de produire un diagnostic dans notre cas. La base de règles explique les liens entre les faits ou les données de la base de faits.

On a choisie la structure suivante pour les règles :

SI « condition » Alors « Conclusion »

Code source : la base de règle (voir annexe C)

- pour la base de règle, on a choisie une base dans la partie condition contient une seul condition (fait) et qui donne une sel conclusion

Au début , on a criée deux tableaux :

- **BRp1 :** le premier tableau pour les faits de condition
- **BRp2** : le deuxième tableau pour les faits de conclusion .

Après on demande a l'utilisateur de donnée le nombre des règles , en suite on demande de remplir les tableau .

A la fin nous ajoutant un autre tableau **BR** pour faire la concaténation des deux tableaux , pour afficher correctement l'ensemble de règle .

- A la fin : réalisation du moteur d'inférence

Le moteur d'inférence est un programme qui construit automatiquement le raisonnement du système en sélectionnant et en activant les règles pertinentes de la base de connaissances. un moteur d'inférence est nécessaire pour relier la description d'un problème aux capacités d'analyse d'une situation donnée et Décide quelles règles sont satisfaites par les faits fournis, donne la priorité aux règles et exécute la règle prioritaire la plus élevée. De façon générale, le moteur d'inférence sera capable de répondre à des questions, de raisonner et de tirer les conséquences impliquées par la connaissance incluse dans le système. Pour cela, il s'oppose à l'informatique traditionnelle où l'exécution d'un programme correspond à un enchaînement de procédures spécifiques, dans un ordre complètement ou partiellement programmé.

Code source : moteur d'inférence Chainage avant (voir annexe C)

Ce code source représente la classe principale qui faire une appelle a les autre classe .

Après que l'utilisateur finira le remplissage de la base de fait et la base de règle , on demandera a lui d'entrer un fait pour le testé .

Au début on faire un test sur la base de fait si le fait existe alors en affiche un message que le fait existe dans la base de fait , sinon on passe a notre moteur de recherche , avec le principe de chainage avant , on a criée une autre table pour ajoute les nouvelle fait qui ramenée depuis la base de règle, après le test sur les deux précédents tableaux de base de fait et le tableaux qui contient la partie de condition des bases de fait .

S'il y a une règle déclenchable le fait de la conclusion et rajouter dans le nouveau tableau et ainsi de suite . en refaire la même chose .Le programme s'arrêterait soit en a trouver la solution soit saturation aucune règle n'est dèclanchable

S'il ya plusieurs règles déclenchable en vas choisir la première règles qui apparaisse dans la base de règles et elle ne sera pas déclencher une autre fois

4.4.5. Domaine d'application des systèmes experts

Les systèmes experts ont été conçus pour résoudre certains types de problèmes comme en médecine, en droit, en chimie, en éducation etc. Il existe un immense éventail de tâches techniques qui requièrent l'application d'un bon nombre de connaissances que la plupart des gens ne possèdent pas. Ces tâches en général peuvent être accomplies par des experts qui ont accumulé les connaissances requises. les systèmes experts sont en mesure de fournir une logique de contrôle capable d'intégrer l'acquisition de données en temps réel, des algorithmes numériques nécessaires pour l'analyse de l'écoulement de puissance, de la stabilité, de la fiabilité etc. et l'expertise requise dans le traitement et l'interprétation de l'information. Pour faire un résumé des différentes applications des systèmes experts dans les domaines de puissance est présenté dans cette section:

–**Applications en systèmes de puissance:** Traitement des signaux d'alarme, distribution, opérations, protection, sécurité, contrôle, test des équipements, entretien , etc ..

–**Applications en distribution:** Automatisation, design, surveillance de sous-station, localisation de condensateurs, configuration de circuit d'alimentation, allocation de charge, contrôle de sous-station, opération des condensateurs, design de sous-station, support logistique de sous-station.

– **Applications en planification:** Prédiction de la consommation, gestion, gestion des projets, génération, transmission, coordination de l'isolation, planification des opérations.

– **Applications en opérations:** analyse de congestion, organisation de la génération commutation optimale, prédiction de la consommation, planification des opérations environnement, contrôle de la fréquence de charge, organisation des opérations.

–**Applications systèmes de sécurité en puissance:** Sécurité dynamique, contrôle des situations de panique, surcharge des lignes, contrôle des dégâts. [CARLOS ORTIZ ,1995].

4.5. Application en distribution des systèmes experts

Si en prend l'application en distribution plus précisément le problème d'allocation de charge statique dans un environnement informatiques distribués sur un ensemble de processeurs hétérogènes, où les tâches sont indépendantes et unitaires.

Le problème de l'équilibrage de charge consiste à allouer des données au départ (fractions d'une application donnée), puis éventuellement à les redistribuer sur un ensemble de processeurs

afin de minimiser leur temps de traitement. Par contre, elles pénalisent les tâches à faible durée de vie, majoritaires dans un système multiutilisateurs

Un des obstacles majeur à l'obtention de bonnes performances réside dans l'équilibrage de la charge entre les différents processeurs. En effet, le temps de réponse d'une exécution parallèle est égal au temps de réponse du processeur le plus chargé. Une mauvaise distribution de la charge de travail peut donc entraîner une chute de performance importante.

Dans cette partie, nous avons implémenté une version modifiée d'un algorithme d'équilibrage de charge statique entre plusieurs processeurs hétérogènes et dont les tâches sont indépendantes. Mentionné dans les travaux de [J.Jack, Dongarra et al, 2007] dans les environnements statiques, l'algorithme modifié est basé sur deux paramètres qui sont le temps d'exécution et la probabilité de défaillance.

4.5.1. Équilibrage de charge statique

Dans Le système d'équilibrage de charge statique, la performance des nœuds est déterminée au début de l'exécution [M.Ali ,2010], Le gestionnaire d'équilibrage de la charge répartit les tâches sans exécuter de requête pour connaître la charge actuelle sur les processeurs du système. L'algorithme utilisé ici doit donc pouvoir calculer les charges à l'avance, afin de permettre un équilibrage de charge efficace. Donc, la charge produite par une tâche doit être fournie par la tâche elle-même ou par d'autres sources. De même, les différentes capacités des processeurs ou des nœuds de réseau doivent être connues. L'équilibrage de charge statique est plus simple et plus stable que dynamique, c'est pour cette raison que nous avons choisi de traiter cette catégorie, Cette catégorie elle-même est divisée en deux sous catégories qui sont optimales et sous optimales. Dans cette partie , nous nous intéressons à la sous-catégorie optimale [Md. Firoj et al., 2012].

4.5.2. Implémentations d'un Algorithme d'allocation optimal fiable modifié

Nous prévoyons toutefois que les règles d'équilibrage de charge dynamique surpassent les mesures statiques, mais ces règles doivent réagir à l'état actuel du système. Cela rend les politiques dynamiques nécessairement plus complexes que les politiques statiques. L'une des préoccupations est que les frais généraux engendrés par une telle complexité peuvent annuler les avantages de l'équilibrage dynamique de la charge [Hioual et al, 2018].

Dans les algorithmes d'équilibrage de charge statique, les taches sont alloués aux processeurs au moment de la compilation en fonction des performances des nœuds. Aucune modification ne

sera apportée à l'exécution une fois les tâches attribuées aux processeurs. Le nombre de tâches dans chaque nœud est fixé dans l'algorithme d'équilibrage de charge statique

Le travail proposé dans cette partie est l'utilisation d'un algorithme d'équilibrage de charge statique.

Cet algorithme modifié est basé sur les résultats d'un algorithme optimal existant déjà, dont le principe est décrit dans la section suivante.

Algorithme optimal

Cet algorithme se base sur le principe de faire une distribution d'un ensemble de taches indépendantes sur un ensemble de processeurs et cela en respectant le temps d'exécution de chaque processeur de façon à affecter au processeur ayant le temps d'exécution le plus petit le maximum de tache a exécuté pour obtenir un temps d'exécution optimal et par conséquent obtenir un meilleur équilibrage des charges [A.Legrand and Y.Robert,2005].

Algorithme 1 : Allocation Optimal (voir annexe D)

Initialement on affecte à chaque processeur un temps d'exécution compris entre 0 et 1 puis en calcul la charge de chaque processeur suivant son temps d'exécution c-à-d calculer le nombre de taches affecter a chacun(voir la ligne N°2 de l'algorithme).

En suite en calcul la somme des taches affecter et on la compare avec la charge initial, si on obtient une égalité qui est impossible dans ce cas puisque dans la formule utilisé dans l'algorithme en prend la partie entière seulement, donc en recommence l'opération en affectant les taches restantes aux processeur qui vérifie cette formule (min { $t_i \times(n_i+1)$).

Enfin en arrivant au résultat ($\sum_{i=1}^{m} n_i$<N) en calcul le temps optimal qui correspond au processeur le plus long.

Algorithme d'allocation optimal fiable modifié

Pour résoudre le problème de l'équilibrage de charge entre plusieurs processeurs, nous avons utilisé l'algorithme d'allocation optimale fiable modifié à tâche unitaire indépendante . Ceci utilise le résultat d'un algorithme optimal qui existe déjà .

L'algorithme modifié suppose que le processeur a le plus petit temps d'exécution. sa probabilité de tombé en panne augmente en raison de son utilisation fréquente. Dans ce cas, un

processeur dont le temps d'exécution est plus long peut avoir la possibilité de charger une tâche en tant qu'autre processeur. [Hioual et al, 2018]

Cette proposition évite qu'un processeur qui a un temps d'exécution important ne peut pas avoir de charge.

Donc, dans notre algorithme, afin de calculer la charge de chaque processeur, nous proposons de multiplier le temps d'exécution (en ordre croissant) et la probabilité de tombé en panne (en ordre décroissant).

Algorithm 2 Algorithme d'allocation optimal fiable modifié (voir annexe D)

Initialement, pour chaque processeur, une durée d'exécution, une probabilité de défaillance sont affectées et triées par ordre croissant en fonction du produit (ti * bi), puis nous calculons la durée d'exécution optimale basée sur le résultat de l'algorithme optimal en fonction de la formule (M = q * Mopt) avec q nombre aléatoire compris entre (1,1 et +). Nous réessayons le même travail de l'algorithme optimal qui consiste à calculer la charge de chaque processeur.

Évaluation des résultats

Dans cette section, nous présentons et discutons les résultats expérimentaux de l'algorithme proposé. Notre système hétérogène est constitué de 10 processeurs différents.

Nous avons évalué les performances de notre algorithme proposé en le comparant avec l'algorithme optimal. Nos expériences sont présentées dans le tableau 4.1 [Hioual et al, 2018].

Index	Algorithme optimal	Algorithme d'allocation optimal fiable modifié
M=10, N=10	0.8092626068765834	0.5693465712276581
M=10, N=20	0.8091174416740206	0.6782884960389196
M=10, N=55	2.9343992848504596	0.9865698646181081
M=10, N=76	1.0171492605480703	0.9080642269793924
M=10, N=100	2.132567979224129	1.0000050068653484
M=10, N=124	3.6982639095857226	1.2397414087079151
M=10, N=200	4.805219848100306	3.1443928125787926
M=10, N=350	11.539574568539892	4.94904062305052
M=10, N=400	18.268866833202715	11.81588188130805

Tableau 4.1. Résultats d'exécution des algorithmes

Remarque:

M nombre actuel de processeurs

N nombre actuel de tâches

- La première colonne représente le nombre de processeurs (M) et le nombre de tâches (N).
- La deuxième colonne représente le temps optimal après l'exécution de l'algorithme optimal dans différents cas.
- La troisième colonne représente le temps optimal après l'exécution de l'algorithme d'allocation optimal fiable modifié dans différents cas.

Les résultats obtenus après l'exécution de l'algorithme d'allocation optimal fiable modifié nous donnent un bon équilibrage de charge, alors que ceux obtenus par l'algorithme optimal sont approximatifs et très éloignés des résultats de l'algorithme d'allocation fiable optimal modifié.
Cette variance des résultats est due à l'utilisation d'un autre paramètre dans l'algorithme d'allocation fiable optimal modifié qui est la probabilité de tombé en panne: l'algorithme optimal

classe les processeurs en fonction du temps d'exécution, mais le second les classe en fonction du produit (bi * ti), donc un processeur dont le temps d'exécution est minimal par rapport au premier algorithme peut prendre une grande charge, ce même processeur peut perdre son classement dans le deuxième algorithme en raison de l'utilisation du nouveau paramètre (probabilité de tombé en panne).

Pour résoudre ce problème, nous proposons quelques modifications sur l'algorithme d'allocation fiable optimal cité dans les travaux de [J.Jack, Dongarra et al,2007] consistant à multiplier le temps d'exécution (en ordre croissant) et la probabilité de tombé en panne (en ordre décroissant). Avec cet algorithme modifié, nous avons obtenu un bon équilibrage de charge par rapport à l'algorithme optimal (voir tableau 4.2).

Exemple:

Lorsque le nombre de tâches N = 200 et le nombre de processeurs M = 10

Temps d'exécution optimal de:

Algorithme Optimal = 0.8092626068765834

Algorithme d'allocation fiable optimal modifié = 0.5693465712276581

Le tableau ci-dessous indique pour chaque processeur: son temps d'exécution, sa probabilité de tombé en panne et sa charge.

- La première colonne représente le nombre de processeurs (M).
- La deuxième colonne représente le temps d'exécution pour chaque processeur.
- La troisième colonne représente la probabilité de tombé en panne de chaque processeur
- La quatrième colonne représente la charge de chaque processeur lorsque nous utilisons l'algorithme optimal.
- La dernière colonne représente la charge de chaque processeur lorsque nous utilisons l'algorithme d'allocation fiable optimal modifié

Si nous prenons les lignes 1 et 2 du tableau ci-dessous, nous remarquant que le temps d'exécution du processeur n ° 1 est inférieur à celui du processeur n ° 2, mais que la répartition de la charge utilisant l'algorithme optimal est de 4 pour le processus n ° 1 et de 44 pour le second processur qui est une mauvaise distribution si on le compare au second algorithme dont le processeur n°1 a une charge de 27 et le second à 4 qui est une très bonne affectation [Hioual et al, 2018].

N° de processeurs	Temps d'exécution	Probabilité de tombé en panne	La charge de processeur	La charge de processeur1
1	0.2531038099098497	0.8165435232144264	4	27
2	0.8074930112528398	0.2992409774751406	44	4
3	0.7305904208846935	0.305024712213016702	11	3
4	0.25919653042495416	0.9779898941576612	5	24
5	0.7392568580963457	0.49558396931055504	11	5
6	0.7806177027951934	0.29648073832828514	5	3
7	0.8384233501055515	0.2563282751909879	10	2
8	0.13334908826862668	0.9935366037199109	4	29
9	0.9519213329858853	0.27063924366699776	3	1
10	0.7725732696203601	0.038143588509292226	3	2

Tableau 4.2. Résultats de la répartition de la charge à l'aide des algorithmes

Les résultats obtenue peux être sauvegarder dans une base de connaissance pour les réutiliser dans le future c.-à-d. Pour chaque processeur en peut sauvegarder ses caractéristiques ainsi que la charge qui peut la supporté.

4.6. Conclusion

Dans ce chapitre nous avons présenté en premier lieu comment acquérir des connaissances dans un système de raisonnement par cas en utilisant le système FrakaS combiné avec la logique de description. FrakaS vise à faciliter l'acquisition de la connaissance du domaine. Cette connaissance, bien qu'utilisée pour adapter les cas, n'est pas censée être liée aux cas. Dans FrakaS, l'identification des connaissances à acquérir se fait non pas en analysant le raisonnement, mais en analysant la solution. En cas d'échec, la solution est analysée par l'expert qui doit identifier les incohérences dans la solution en utilisant ses propres connaissances. Le système est capable de déduire, à partir de l'analyse de ces incohérences, de nouvelles connaissances qui lui permettront d'éviter de répéter l'erreur à l'avenir. Le principe de ce système vise à acquérir des

connaissances qui ne sont pas déjà présentes dans le système par interaction avec l'environnement.

Notre travail est basé sur l'adaptation conservatrice puisque le système FrakaS l'utilise, et nous avons utilisé la méthode REX pour capitaliser les connaissances dans une mémoire d'entreprise. Nous avons ensuite proposé un générateur de systèmes experts en utilisant le chainage avant qui peut être utilisé dans plusieurs domaines, et nous avons terminé avec l'application de ce système dans un domaine de distribution.

CONCLUSION GENERALE ET PERSPECTIVES

Le système de raisonnement à partir de cas (RàPC) est un paradigme de raisonnement qui s'appuie sur la remémoration de problèmes passés antérieurement résolus, pour résoudre de nouveaux problèmes. Ce paradigme s'appuie sur plusieurs connaissances. Parmi ces connaissances il y a les cas sources, mais beaucoup de systèmes utilisent également d'autres types de connaissances, telles que les « connaissances du domaine ». Les inférences effectuées par le système de raisonnement à partir de cas seront meilleures si ces connaissances sont complètes et correctes.

Les connaissances du domaine sont disponibles mais ne sont pas suffisantes ce qui impose un vide avec les connaissances de l'expert, donc il faut toujours acquérir de nouvelles.

C'est dans le cadre de cette problématique que nous avons effectué notre travail de recherche. Cette thèse présente d'une part une approche interactive pour l'acquisition de connaissances du domaine fondée sur des échecs d'un système de RàPC. Cette approche a été implémentée dans une nouvelle version du système FrakaS, pour générer un prototype qui s'appuie sur une représentation en logique de description au lieu de la logique propositionnelle. Nous nous sommes restreints au cadre où un tel système donne des solutions cohérentes avec les connaissances du domaine.

Dans le raisonnement à partir de cas, l'adaptation est souvent considérée comme une tâche difficile basée généralement sur des heuristiques, par rapport à la récupération qui est censé être plus simple à concevoir et à mettre en œuvre. Une approche de l'adaptation, appelée adaptation conservatrice est également présentée. Celle-ci nécessite une certaine connaissance du domaine, où une caractéristique importante de l'adaptation conservatrice s'illustre par le fait que la connaissance de l'adaptation fait partie des connaissances du domaine. D'autre part, nous avons présenté notre contribution principale qui se situe en l'acquisition des connaissances du domaine

au vu du système FraKas auquel nous avons apporté une modification et qui est basé sur l'adaptation conservatrice.

Cette adaptation est basée :

- Sur la théorie de révision qui consiste à faire des changements minimes sur le cas source afin d'être cohérant avec le cas cible et pour réaliser cette adaptation nous avons choisi un opérateur de révision pour réviser la base de cas et puisque nous avons utilisé la logique de description où dans ce formalisme, l'adaptation d'un cas source pour résoudre un problème cible est plus complexe. Notre base de connaissances contient deux parties: la TBox (Terminologie Box) et ABox (Assertion Box), ce qui nous oblige d'appliquer le principe de cette adaptation à ces deux parties. Dans notre travail, nous ne considérons que la révision des parties terminologies de la logique de description. En effet, la révision de la partie « assertion » est très compliquée à réaliser en la comparant à la partie « terminologie ».
- Sur un opérateur de révision qui doit satisfaire tous les postulats AGM, l'AGM (développé par Alchourrón, Gärdenfors et Makinson) détermine le comportement d'une fonction de changement c'est-à-dire un ensemble de propriétés qu'un opérateur qui effectue la révision doit satisfaire pour être considéré comme rationnel dans notre travail. Nous avons adapté ces postulas à notre opérateur choisi et au formalisme de la logique de description.

Nous avons utilisé deux types d'échec qui sont mentionné dans les travaux de Amélie Cordier et al à savoir : Un échec du premier type est caractérisé par une contradiction de la solution inférée par le système de RàPC avec les connaissances de l'expert (quoiqu'il soit en cohérence avec les connaissances du domaine). Une analyse de cet échec permet alors de mettre en évidence les descripteurs menant à cette contradiction et ainsi d'enrichir les connaissances du domaine. Un autre échec du deuxième type est caractérisé par le fait que la solution n'est que partielle : il manque des informations pour pouvoir l'utiliser. Si une analyse des instances de solution permet de mettre en évidence que certaines d'entre elles contradictoires, cette analyse permet d'acquérir de nouvelles connaissances. Enfin, l'explication textuelle fournie par l'expert est une source pour acquérir de nouvelles connaissances.

D'un autre côté, comment capitaliser les connaissances dans une mémoire d'entreprise en utilisant la méthode REX (Retour d'EXpérience), la mémoire d'entreprise contient des connaissances du domaine de l'entreprise ainsi que différentes sources de connaissances basées sur une ontologie commune. Cette ontologie permet d'utiliser les connaissances stockées dans cette mémoire par les différentes classes d'acteurs et selon leurs différentes tâches à faire.
L'objectif de la méthode proposée est de capitaliser les connaissances qui doivent être stratégiques et de favoriser le retour d'expérience. La méthode a été implémentée pour le domaine médical.
Ensuite nous nous sommes intéressés à la capitalisation des connaissances en informatique en utilisant l'approche de gestion des connaissances (Systèmes Experts).
Un système expert générique a été implémenté pour être utilisé dans plusieurs domaines. Ce système est composé principalement d'une base de connaissances contenant des connaissances factuelles qui sont des faits spécifiques du domaine et des connaissances déductives que sont les règles permettant de spécifier le comportement du système. Nous avons utilisé un moteur d'inférence en chainage avant qui permet d'inférer de nouvelles connaissances à partir de la base de connaissances du système parmi les différents modes d'invocation des règles.
Nous avons terminé notre travail avec une application portant sur le problème d'allocation de charge statique dans un environnement informatique distribué sur un ensemble de processeurs hétérogènes, où les tâches sont indépendantes et unitaires.

Perspectives

Dans cette thèse nous avons utilisé une nouvelle version de système FrakaS, un prototype basé sur la logique de description qui nous a permis d'envisager certaines orientations futures de recherche telles que :

- L'étude plus avant le cas de la révision de la partie assertion de la base de connaissance qui est un point très important à réaliser.
- La proposition d'autre formalisme d'adaptation telle que l'adaptation différentielle.
- L'implémentation d'une une nouvelle version du système FrakaS, fondée sur le formalisme da la logique floue.
- L'implémentation de cette nouvelle version sur une situation réelle.
- Trouver une solution pour le problème de l'appropriation des connaissances.

- Implémenter un générateur d'un système expert qui utilise une base de faits et une base de règles avec plusieurs conditions et conclusions on utilisant le chainage arrière comme second moteur d'inférence.
- Modifier ou ajouter d'autres paramètres pour améliorer les résultats de l'algorithme d'équilibrage de charge statique.

BIBLIOGRAPHIE

[Aamodt, 1989] Aamodt, A. (1989). Towards Expert Systems that Learn from Experience. In Proceedings of the DARPA Case-Based Reasoning Workshop, pages 181–187, Pensacola Beach, Florida. Morgan Kaufmann.

[Aamodt, 1990] Aamodt, A. (1990). Knowledge-Intensive Case-Based Reasoning and Sustained Learning. In Aiello, L., editor, Proceedings of the 9th European Conference on Artificial Intelligence (ECAI'90), pages 1–6. Pitman Publishing, London.

[Aamodt, 1991] Aamodt, A. (1991). A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning. PhD thesis, Trondheim University.

[Aamodt, 1994] Aamodt, A. (1994). Explanation-driven Case-Based Reasoning. In Wess, S., Althoff, K. D., and Richter, M. M., editors, Topics in Case-Based Reasoning, volume LNAI837, pages 274–288, Berlin. Springer.

[Aamodt, 2001] Aamodt, A. (2001). Modeling the knowledge contents of CBR systems. In Proceedings of the Workshop Program (at ICCBR'01), pages 32–37, Vancouver.

[Aamodt, 2004] Aamodt, A. (2004). Knowledge-Intensive Case-Based Reasoning in Creek. In Proceedings of the 7th European Conference on Case-Based Reasoning (ECCBR'04), pages 1–15, Madrid, Spain. Springer Berlin / Heidelberg.

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(1):39–59.

[Abbott, 1884] Abbott, E. A. (1884). Flatland, a romance of many dimensions. E.Books, Second revised edition edition.

[Alavi, 1997] Alavi M.(1997), KPMG Peat Marwick U.S. One Gain Brain, Harvard Business School (Case) .

[Alavi et Leidner, 2001] M. Alavi and D. Leidner.(2001). Knowledge Management and Knowledge Management Systems: conceptual foundations and research issues; MISQuarterly Vol. 25 No.1, pp.107-136.

[Alavi et Leidner, 1999]M. Alavi, and D. Leidner, Dorothy (1999) "Knowledge Management Systems: Issues, Challenges, and Benefits," Communications of the Association for Information Systems: Vol. 1 ,

[Alchourrón et al., 1985] Alchourrón, C. E., Gärdenfors, P., and Makinson, D. (1985). On the Logic of Theory Change: partial meet functions for contraction and revision. Journal of Symbolic Logic, 50:510–530.

[A.Legrand and Y.Robert,2005] A.LegrandandY.Robert,"AlgorithlmiqueParallèle",Dunod,2005.

[Althoff, 2001] Althoff K.D.(2001), Case-Based Reasoning. S.K. Chang (Ed.), Handbook on Software Engineering and Knowledge Management, pp. 549-588.

[A.Cordier, 2009]A.Cordier(2009), "Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning", These Lyon 1 University

[Andersen, 1996] Andersen A.(1996), The Knowledge Management Assesment Tool: External Benchmarking Version, The American Productivity and Quality Center, Winter.

[Arcos and De Mantaras, 1997] Arcos, J. L. and De Mantaras, R. L. (1997). Perspectives: A Declarative Bias Mechanism for Case Retrieval. In Leake, D. and Plaza, E., editors, Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR'97), pages 279–290. Springer Berlin / Heidelberg.

[Armengol and Plaza, 1994a] Armengol, E. and Plaza, E. (1994). A Knowledge Level Model of Case-Based Reasoning. In Proceedings of the European Workshop on Case-Based Reasoning (EWCBR'94), pages 53–64, London, UK. Springer-Verlag.

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel Schneider, P., editors (2003). The Description Logic Handbook. Cambridge University Press, cambridge, UK.

[Bartlett, 1964] Bartlett F.(1964), Remembering (5th ed.). Cambridge University Press.

[Bentebibel, 2008] R.Bentebibel, (2008),Raisonnement à aprtir de cas textuel pour la sécurité routière :Le systéme SAARA.

[Barthès et al, 1999]Barthès, J. P.(1999) : Capitalisation et transfert des connaissances au sein des PMI. Journée Mémoire d'Entreprise. Europôle Méditerranéen de l'Arbois. Les Milles.

[Bergmann et Wilke, 1998] Bergmann, R. and Wilke, W. (1998). Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning. In Prade, H., editor, Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98), pages 53–57. John Wiley and Sons.

[Branting et al., 2001] Branting, K., Hastings, J., and Lockwood, J. (2001). CARMA: A Case-Based Range Management Advisor. In Proceedings of the 13th Innovative Applications of Artificial Intelligence Conference (IAAI'01), pages 3–10, Seattle, Washington. AAAI Press.

[Balouchi .B et al., 2015] Balouchi B, Nikoo MR, Adamowski J. (2015). Development of expert systems for the prediction of scour depth under live-bed conditions at river confluences: Application of different types of ANNs and the M5P model tree. Appl Soft Comput.;34:51-9.

[Breuker et Van de Velde, 1994] J. Breuker and W. Van de Velde (eds.)(1994). The CommonKADS Library for Expertise Modelling, IOS Press, Amsterdam, The Netherlands.

[Capponi, 1995] Capponi C.(1995), Identification et exploitation des types dans un modèle de connaissances à objet, Thèse de doctorat, Université Joseph Fourier, Grenoble.

[Carbonell, 1984] Carbonell, J. G. (1984). Learning by analogy: Formulating and generalizing plans from past experiences. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, Machine Learning: An Artificial Intelligence Approach, pages 137–162. Springer - Berlin, Heidelberg.

[Carbonell, 1986] Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, Machine Learning: An Artificial Intelligence Approach, pages 371–392. Morgan Kaufman Publishers: San Mateo, CA.

[Carbonell and Veloso, 1988] Carbonell, J. G. and Veloso, M. (1988). IntegratingDerivational Analogy into a General Problem Solving Architecture. In Proceedings of the 1st Workshop on Case-Based Reasoning, pages 104–124. Tampa, FL: Morgan Kaufmann.

[Cheng and Hüllermeier, 2008] Cheng, W. and Hüllermeier, E. (2008). Learning Similarity Functions from Qualitative Feedback. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08), pages 120–134. Springer - LNAI 5239.

[Cho et al., 1999] Cho, K. D., Lee, J. P., Kim, K. T., and Hwang, S. C. (1999). A Learning of Adaptation Knowledge for Case-Based Reasoning System. In Proceedings of the 17th IASTED International Conference - Applied Informatics, pages 673–675, Innsbruk, Austria. ACTA Press, Anaheim, Calgary, Zurich.

[Choo, 1996] Choo C.(1996), An integrated Information Model of the Organization : The Knowing Organization.

[Cojan et al, 2008] Cojan, J. and Lieber, J. (2008). Conservative Adaptation in Metric Spaces. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08), pages 135–149. Springer - LNAI 5239.

[Compton and Jansen, 1988] Compton, P. and Jansen, R. (1988). Knowledge in Context: a strategy for expert system maintenance. In Proceedings of the 2nd Australian Joint Artificial Intelligence Conference, pages 292–306.

[Cordier, 2004] Cordier, A. (2004). Gestion des Connaissances pour des Systèmes à Base de Connaissances hybrides. Master's thesis, Université Claude Bernard Lyon 1.

[Cordier and Fuchs, 2005] Cordier, A. and Fuchs, B. (2005). Un assistant pour la conception et le développement des systèmes de RàPC. In Després, S., editor, Actes du 13ème atelier de raisonnement à partir de cas (RàPC'05), Nice.

[Cordier et al, 2007] Cordier, A., Fuchs, B., Lieber, J., and Mille, A.(2007). Acquisition de connaissances du domaine d'un système de RàPC : une approche fondée sur l'analyse interactive des échecs d'adaptation - le système FrakaS.

[Cordier et al., 2008a] Cordier, A., Fuchs, B., Lana de Carvalho, L., Lieber, J., and Mille, A. (2008). Opportunistic Acquisition of Adaptation Knowledge and Cases - The IakA Approach. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08), pages 150–164. Springer - LNAI 5239.

[Corbel, 1997] Corbel J.C.(1997), Méthodologie de retour d'expérience : démarche MEREX de Renault. Connaissances et savoir-faire en entreprise, Hermès, pp. 93-110.

[Dalal, 1988] Dalal, M. (1988). Investigations into a theory of Knowledge Base Revision: Preliminary Report. In Proceedings of the 7th National Conference on Artificial Intelligence, pages 475–479, Saint Paul, Minnesota, USA.

[Davenport et Prusak, 2000] Davenport, T., & Prusak, L. (2000). Working Knowledge: how organisations manage what they know? Harvard, USA: Harvard Business School Press.

[Davenport & Prusak, 1997] Davenprot T.H. et Prusak L.(1997), Information Ecology : Mastering the Information and Knowledge Environment. Oxford, New York.

[Demarest, 1997] Demarest M.(1997), Understanding Knowledge Management. Long Range Planning, Vol. 30, No.3, pp. 374-384.

[Descartes, 1637] René Descartes.1637.Discours de la méthode.

[Dieng et al., 1999] Dieng R.(1999), Corby O., Giboin A. et Ribière M. (1999), Methods and Tools for Corporate Knowledge Management. International Journal of Human-Computer Studies, 51 :567-598, Academic Press.

[Flouris et al, 2005] G. Flouris, D. Plexousakis, and G. Antoniou.(2005), On applying the AGM theory to DLs and OWL. In Proc. of ISWC, pages 216–231.

[Fuchs et al., 2006a] Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (2006). A general strategy for adaptation in Case-Based Reasoning. Research report - RR-LIRIS-2006-016.

[Fuchs et al., 2006b] Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (2006). Une première formalisation de la phase d'élaboration du raisonnement à partir de cas. In Actes du 14ème atelier de raisonnement à partir de cas (RàPC'06), Besancon.

[Fuchs and Mille, 1999] Fuchs, B. and Mille, A. (1999). Une modélisation au niveau connaissance de la tâche d'adaptation en raisonnement à partir de cas. In Trousse, B. and Mille, A., editors, Actes du 7ème atelier de raisonnement à partir de cas (RàPC'99), pages 27–36, Palaiseau, France.

[Fuchs and Mille, 2000] Fuchs, B. and Mille, A. (2000). Une modélisation au niveau connaissance du raisonnement à partir de cas. In Aussenac-Gilles, N., editor, Actes des journées francophones d'acquisition des connaissances (IC'00), pages 3–10, Toulouse.

[Fuchs et al., 1995] Fuchs, B., Mille, A., and Chiron, B. (1995). Operator decision aiding by adaptation of supervision strategies . In Proceedings of the 1st International Conference on Case-Based Reasoning (ICCBR'95), pages 23–32, Sesimba, Portugal. Springer.

[Fuchs et al., 1999] Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (1999). Towards a Unified Theory of Adaptation in Case-Based Reasoning. In Schmitt, S. and Vollrath, I., editors, Proceedings of the 3rd International Conference on Case-Based Reasoning (ICCBR'99), pages 104–117, Munich, Germany. LSA, University of Kaiserslautern.

[Gick and Holyoak, 1980] Gick, M. and Holyoak, K. (1980). Analogical Problem Solving. Cognitive Psychology, 12:306–355.

[Guilin Qi et al, 2009]Guilin Qi, Jianfeng Du.(2009), Model-based Revision Operators for Terminologies in Description Logics, IJCAI

[Grundstein, 2000] Grundstein M.(2000), Management des connaissances de l'entreprise: problématique, axe de progrès, orientations.

[Grundstein, 2002] Grundstein, M. (2002). Gameth : un cadre directeur pour repérer les connaissances cruciales pour l'entreprise, Lamsade Université Paris-Dauphine 18 pages.

[Grundstein, 1995] Grundstein M.(1995), La capitalisation des connaissances de l'entreprise, 1996.

[Grundstein et Rosenthal-Sabroux, 2004] Grundstein, M., & Rosenthal-Sabroux, C. (2004). GAMETH, A Decision Support Approach to Identify and Locate Potential Crucial Knowledge. In D. Remenyi (Ed.), Proceedings 5th European Conference on Knowledge Management (pp. 391 – 402). Reading, UK: Academic Conferences Limited.

[Hammond, 1986] Hammond, K. (1986). CHEF: A model of case-based planning. In Press, A., editor, Proceedings of the 5th National Conference on Artificial Intelligence, pages 267–271, Menlo Park, CA.

[Hedberg et al, 1981] Hedberg, B.(1981) :How Organizations Learn and Unlearn. In P. Nystrom & W. H. Starbuck (Eds.), Handbook of Organizational Design (Vol. 1). London: Cambridge University Press.

[Hioual et al, 2018] O.Hioual, MT.Laskri, SM.Hemam, O.Hioual, L.Maifi.(2019), Towards An Implementation of A Modified Static Load Balancing Algorithm To Minimize Execution Time, Recent Patents on Computer Science, volume 12, Issue 1, pp69-74.

[Hioual et al, 2013] O.Hioual, MT.Laskri, L.Maifi.(2013), Knowledge Acquisition with system FrakaS-DL, International Conference on Control Engineering & Information Technology(CEIT'13),Proceeding Engineering & Technology- Volume 2, pp158-163.

[Hioual et al, 2017] O.Hioual, MT.Laskri, L.Maifi.(2017), Towards REX method for capitalizing the knowledge of a corporate memory, International Conference on Intelligent Decision Technologies, Proceedings of the 9th KES International Conference on Intelligent Decision Technologies (KES-IDT 2017), pp 206-215 – Part II, Part of the Smart Innovation, Systems and Technologies book series (SIST, volume 73)

[Holsapple & Joshi, 1999] Holsapple C.W. et Joshi K.D.(1999), Description and Analysis of Existing Knowledge Management Frameworks. Proc. of the 32nd Hawaii International Conference on System Sciences, IEEE.

[J.Jack Dongarra et al, 2007] J.Jack, Dongarra et al,(2007), Bi-objective Scheduling Algorithms for Optimizing Makespan and Reliability on Heterogeneous Systems, SPAA'07.

[Kass et al., 1986] Kass, A., Leake, D., and Owens, C. (1986). Explanation patterns: Understanding mechanically and creatively., chapter SWALE: A program that explains, pages 232–254. Lawrence Erlbaum, Hillsdale, NJ, Schank, R. edition.

[Kaufman, 1987] Kaufman A.(1987), Nouvelles logiques pour l'intelligence artificielle. EditionsHermès, Paris, France, Mai.

[Kolodner, 1988] Kolodner, J., editor (1988). Workshop on case-based Reasoning, DARPA 88, Clearwater, Florida. Morgan Kaufmann, San Mateo.

[Kolodner, 1993] Kolodner, J. (1993). Case-Based Reasoning. Morgan Kaufmann, San Mateo, CA.

[Koton, 1988] Koton, P. (1988). Reasoning about evidence in causal explanations. In Press, A., editor, Proceedings of the Seventh National conference on Artificial Intelligence, pages 256–261, Menlo Park, CA.

[Lamontagne et al, 2002] Lamontagne L. et Lapalme G.(2002), Raisonnement à base de cas textuels – état de l'art et perspectives. Revue d'Intelligence Artificielle, Hermès, Paris, 16(3), pp. 339-366.

[Leake, 1996] Leake, D., editor (1996). Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press/MIT Press.

[Lehner et al, 1998] Lehner,. F., Maier, R., Klosa, O.(1998) , Organisational memory Systems Application of advances database and network technologies in organizations, research paper No 19 departments of Business Informatics III University of Regensburg.

[Lehner et al ,2000] S.LehnerD.HojaJ.Schulz-Stellenfleth.(2000), Marine parameters from synergy of optical and radar satellite data, Volume 29, Issue 1, Pages 23-32

[Lieber, 1997] Lieber, J. (1997). Raisonnement à partir de cas et classification hiérarchique. Application à la planification de synthèse en chimie organique. PhD thesis, Université Henri Poincaré, Nancy 1.

[Lieber, 1999] Lieber, J. (1999). Reformulations and Adaptation Decomposition. In Schmitt, S. and Vollrath, I., editors, Proceedings of the 3rd International Conference on Case-Based Reasoning (ICCBR'99), Munich, Germany. LSA, University of Kaiserslautern.

[Lieber, 2006] Lieber, J. (2006). A Definition and a Formalization of Conservative Adaptation for Knowledge-Intensive Case-Based Reasoning Application to Decision Support in Oncology (A Preliminary Report). Research Report.

[Lieber, 2007a] Lieber, J. (2007). Application de la révision et de la fusion des connaissances à l'adaptation et à la combinaison de cas. In Cordier, A. and Fuchs, B., editors, Actes du 15ème atelier de raisonnement à partir de cas (RàPC'07), pages 119–129, Grenoble. Plateforme AFIA.

[Lieber, 2007b] Lieber, J. (2007). Application of the Revision Theory to Adaptation in Case-Based Reasoning: the Conservative Adaptation. In Weber, R. and Richter, M., editors, Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR'07), pages 239–253, Belfast, UK. Springer-Verlag Berlin Heidelberg, LNAI 4626.

[Leonard-Burton, 1995] Leonard-Burton D.(1995), Wellspring of Knowledge. Boston: Harvard Business School Press.

[Lieber et al., 2001] Lieber, J., Bey, P., Boisson, F., Bresson, B., Falzon, P., Lesur, A., Napoli, A., Rios, M., and Sauvagnac, C. (2001). Acquisition et modélisation de connaissances d'adaptation, une étude pour le traitement du cancer du sein. In Actes des Journées d'ingénierie des connaissances (IC'01), pages 409–426, Grenoble, France. Presses Universitaires de Grenoble.

[Lieber et al., 2005] Lieber, J., d'Aquin, M., Badra, F., and Napoli, A. (2005). Case-Based Treatment Recommendations for Breast Cancer. Research Report.

[Locke, 1693] Locke,j.1693. Éducation et politique: Some Thoughts Concerning Education de John Locke.

[Luc lamontagne et al, 2002] Luc Lamontagne, Guy Lapalme.(2002), Raisonnement à base de cas textuels Etat de l'art et perspectives

[M.Ali ,2010] M.Ali , A,(2010), Load balancing in distributed computer systems, International Journal of Computer Science and Information Security., 8 (4).

[Md. Firoj et al., 2012] Md. Firoj Ali1, R.Zaman Khan,(2012), The study on load balancing strategies in distributed system, International Journal of Computer Science & Engineering Survey.

[Malvache et al, 1993]Malvache, P., Prieur, P.(1993) :Mastering corporate experience with the Rex method. Proc. of ISMICK'93, Compiègne.

[Mark W. McElroy, 2002] Mark W., Simoudis E. et Hinkle D.(2002), Case-Based Reasoning: Expectations and Results. Leake, D.B. (Ed) Case-Based Reasoning: Experiences, Lessons and Future Directions, pp.269-294, MIT Press, 1996.

[Matta et al., 2001] Matta N., Ermine J.L., Aubertin G. et Trivin J.Y.(2001), Knowledge. Capitalization with a knowledge engineering approach : the MASK method. Proc. Of IJCAI'2001 workshop on Knowledge Management and Organizational Memory.

[Mille et al., 1999] Mille, A., Fuchs, B., and Chiron, B. (1999). Raisonnement fondé sur l'expérience : un nouveau paradigme en supervision industrielle ? Revue d'intelligence artificielle, 13:97–128.

[Mille, 2006a] Mille, A. (2006). From case-based reasoning to traces-based reasoning. Annual Reviews in Control, 30(2):223–232.

[Minsky, 1975] Minsky.M.(1975).,The Psychology of Computer Vision, chap. A framework for representingknowledge, pp. 211–281. – New York, USA, McGraw-Hill.

[Nonaka et Takeuchi, 1995] Nonaka I. et Takeuchi H.(1995), The Knowledge-Creation Company : How Japanese Companies Create the Dynamics of Innovation. New York/Oxford, Oxford University Press.

[Nonaka et Takeuchi, 1997] Nonaka, I. et Takeuchi, H. (1997), La connaissance créatrice : La dynamique de l'entreprise apprenante, De Boeck Université.

[Nonaka & Konno, 1998] Nonaka, I. and Konno, N. (1998) The Concept of Ba. Building Foundation for Knowledge Creation. California Management Review, 40, 40-54.

[Pierre Ardiri , 2012]. Pierre Ardiri , 2012. Développement D'un Système Expert D'aide A L'orientation Destiné A Des Adolescents ; thèse de Master.

[Polanyi, 1966] Polanyi, M. (1966). The tacit dimension. London, Routledge and Keoan Paul.

[Py, 1994] Py, M. (1994). Un modèle conceptuel de raisonnement par analogie. Revue d'Intelligence Artificielle, 8:63–99.

[Probst, 2002] Probst G.(2002), Managing Knowledge, Building Bloskc for Success. ISBN 0471-99768-4, Wiley, Sussex, England.

[Quillian, 1968] R.Quillian.(1968),Semantic memoryin Minsky (ed)Semantic Information processing,MIT press.

[Rastogi, 2000] Rastogi P.N.(2000), Knowledge Management and Intelectual Capital – The new virtuous reality of competitiveness. Human Systems Management.

[Richter, 1995] Richter, M. M. (1995). The Knowledge Contained in Similarity Measures. Invited Talk of the First International Conference on Case-Based Reasoning (ICCBR'95).

[Rousseau, 1988] Rousseau B.(1988), Vers un environnement de résolution de problèmes en biométrie– apports des techniques de l'intelligence artificielle et de l'interaction graphique. Thèse de doctorat, Université Claude Bernard, Lyon.

[Schank, 1973] Schank, R. (1973), Identification of Conceptualizations Underlying Natural Language. In Schank and Colby (eds.) Computer Mode~s of Thought a~ Language. San Francisco: W.H. Freeman and Company.

[Schanck et al., 1994] Schanck, R. C., Kass, A., and Riesbeck, C. K., editors (1994). Inside Case-Based Explanation. Lawrence Erlbaum Associates.

[Schank and Abelson, 1977] Schank, R. C. and Abelson, R. P. (1977). Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures. Lawrence Erlbaum, Hillsdale, NJ.

[Schank, 1982] Schank, R. C. (1982). Dynamic Memory: A Theory of Reminding and Learning in Computers and People. Cambridge University Press, New York, NY.

[Schank, 1986] Schank, R. C. (1986). Explanation Patterns: Understanding Mechanically and Creatively. Lawrence Erlbaum, Hillsdale, NJ.
[Simon, 1996] Simon G.(1996), Knowledge Acquisition and Modeling for Corporate Memory: Lessons learnt from Experience. Proc. Of KAW'96, Banff, Canada, pp. 11-18

[Slade, 1991] Slade, S. (1991). Case-Based Reasoning: A Research Paradigm. AI Magazine,12(1):42–55.

[Smyth, 1996] Smyth, B. (1996). Case-Based Design. PhD thesis, Trinity College, Dublin, Ireland.

[Smyth and McKenna, 1998] Smyth, B. and McKenna, E. (1998). Modelling the competence of case-bases. In Smyth, B. and Cunningham, P., editors, Proceedings of the 4th European Workshop on Case-Based Reasoning (EWCBR'98), pages 208–220. Springer.

[Smyth et Keane, 1998] Smyth, B. and Keane, M. T. (1998). Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. Artificial Intelligence, 102(2):249–293.

[Sowa, 1984] Sowa J.F.(1984), Conceptuel Structures: Information Processing in Mind and Machine. Reading (Mass.), Addison-Wesley Publishing Company.

[Steels, 1993] Steels L.(1993), Corporate knowledge management. Proc. of ISMICK'93, Compiègne,France, pp. 9-30.

[Szulanski, 1996] Szulanski G.(1996), Exploring Internal Stickiness: Impediments to the Transfer of Best Practice Within the Firm. Strategic Management Journal, 17 (Winter Special Issue), pp. 27-43.

[Tannembaum & Alliger, 2000] Tannenbaum S.I. et Alliger G.M.(2000), Knowledge Management : Clarifying the Key Issues. ISBN 0967923913, IHRIM.

[Taylor, 1997] Taylor R.(1997), Unisys Decision Support Systems Programme. City Gate London, février, 1996.

[Van der Spek et al, 1997] Van der Spek R. et Spijkervet A.(1993), Knowledge Mangement: Dealing Intelligently with Knowledge. Knowledge Management and Its Integrative Elements, Liebowitz J. & Wilcox L.C. (Eds.), CRC Press, New York.

[Van Heijst et al, 1996] Van Heijst, G., Van der Spek ,R. , Kruizinga. E.(1996): Organizing Corporate Memories. Proc. of KAW'96, Banff, Canada, pp. 42.1-17.

[Watson and Marir, 1994] Watson, I. and Marir, F. (1994). Case-Based Reasoning: A Review. Knowledge Engineering Review, 9(4):355–381.

[Wiig, 1993] Wiig K.M.(19931), Knowledge Management Foundations. Arlington: Schema Press.

Annexes

Annexe A : FraKas (FailuRe Analysis for domain Knowledge AcquiSition/ Acquisition de connaissances de domaines à partir d'analyse d'erreurs)

Algorithme de FrakaS

Entrée : un problème cible, une base de cas et des connaissances du domaine CD

Début (algorithme)

La remémoration avec le problème cible retourne le cas (srce, Sol(srce)).

Sol(cible) ← Adaptation(CD, (srce, Sol(srce)), cible)

/* Prise en compte des échecs du premier type */

tant que l'expert trouve que cible ∧ Sol(cible) est incohérent avec ses connaissances

L'expert met en évidence l'échec Inc et un texte explicatif.

CD ← CD ∧ ¬Inc

Le texte explicatif est stocké pour une acquisition des connaissances ultérieure.

Sol(cible) ← Adaptation(CD, (srce, Sol(srce)), cible)

fin (tant que)

/* Prise en compte des échecs du deuxième type */

Si Sol(cible) n'est pas partielle alors sortir

tant que l'expert trouve une inconsistance dans certaines interprétations de cible ∧ Sol(cible)

pour toute interprétation inconsistante I

L'expert met en évidence l'échec Inc et un texte explicatif

CD ← CD ∧ ¬Inc

Le texte explicatif est stocké pour une acquisition des connaissances ultérieure.

fin (pour)

Sol(cible) ← Adaptation(CD, (srce, Sol(srce)), cible)

fin (tant que)

fin (algorithme)

Principe de l'algorithme

Le processus de raisonnement à partir de cas exploite la base de connaissances pour produire une solution candidate. Lorsque la solution candidate est jugée non valide par l'utilisateur, l'expert identifie un sous-ensemble de connaissances incompatibles (notée Inc dans l'algorithme ci dessous). A partir de ce sous-ensemble de connaissances, le système est capable d'apprendre un

nouveau morceau de connaissance. Cette nouvelle connaissance est ajoutée à la base de connaissances ainsi, la base de connaissances améliorée permet au système de produire une nouvelle solution candidate pour le problème actuel. Le processus est répété jusqu'à ce que l'expert valide une solution proposée par les systèmes, à savoir jusqu'à ce que le système trouve une solution adéquate pour le problème [A.Cordier, 2009].

Annexe B : Méthode REX (retour d'expérience)

- ✓ **Création de la requête**

```
String key = search_field.getText();
 String Query = " SELECT ?maladie WHERE { ?y  <http://www.w3.org/2001/maladie-
rdf/3.0#symptome>  \"" + key +"\" . \n ?y  <http://www.w3.org/2001/maladie-rdf/3.0#name>
?maladie .  }";TextBox1.setText(Query);
```

- ✓ **Ouverture d'un fichier RDF**

```
InputStream in = new FileInputStream(new  File("D:/vc-db-1.rdf"));
créer le model de l'enthologie
Model model=ModelFactory.createMemModelMaker().createDefaultModel() ;
model.read(in,null);
System.out.println(model);
in.close();
```

- ✓ **Exécuter la requête**

```
String queryString = TextBox1.getText()
com.hp.hpl.jena.query.Query
q = QueryFactory.create(queryString);
QueryExecution qe = QueryExecutionFactory.create(q, model);
 ResultSet results = qe.execSelect();
```

- ✓ **Récupérer le résultat en xml**

```
TextBox2.setText("");
 if (results.hasNext())
```

```
{
String s = ResultSetFormatter.asXMLString(results);
TextBox2.setText(s);
String xmlRecords = TextBox2.getText();
```

- ✓ **Extraire les mots clés du fichier XML**

```
DocumentBuilder db = null;
 try {
 db = DocumentBuilderFactory.newInstance().newDocumentBuilder();
 InputSource is = new InputSource();
 is.setCharacterStream(new StringReader(xmlRecords));
Document doc = db.parse(is);
NodeList nodes = doc.getElementsByTagName("result");
String values = "";
for (int i = 0; i < nodes.getLength(); i++)
  {
Element element = (Element) nodes.item(i);
values += element.getTextContent().trim() + "\n";
   }
 suggs.setText(values)
  }
```

Principe de REX

REX se décompose en trois étapes : l'analyse des besoins et l'identification des sources de connaissances (spécification et dimensionnement du futur système de gestion des connaissances), construction et mémorisation d'éléments de connaissances sous forme d'une base de données ou une mémoire d'entreprise appelée CEMem, et enfin la mise en place et l'exploitation du système de gestion des connaissances créé. Pour constituer la mémoire d'expérience un modèle descriptif et un réseau terminologique sont définis [Malvache & Prieur, 1993].

Annexe C : Moteur d'inférence - Chainage avant

Algorithme : chaînage avant

ENTREE: BF, BR, F

F est le fait à déduire

tant que F n'est pas dans BF et qu'il existe dans BR une règle applicable **faire**

choisir une règle applicable R

BR = BR − R (désactivation de R)

fin tant que

si F appartient à BF alors

F est établi

sinon

F n'est pas établi

fin si

Code source : la base de fait

```
import java.util.Scanner;
public class base_de_fait{
        public String[] BF1;
        public  void BF(){
      /* Création_de_la_matrice*/
        Scanner sc = new Scanner(System.in);
        int n;
        System.out.println("entré le nbr de base de fait ");
        n = sc.nextInt();
       String[] BF = new String[n];
       BF1= new String[n];
      // Remplissage :
      String f;
      f = sc.nextLine();
```

```
      for (int i=0; i< BF.length;i++)
        System.out.println("entré le fait ");
         f = sc.nextLine();
        BF[i]=f;
         }
    // teste le fait s'il existe déja ou non
      BF1[0]=BF[0];
      int c=0;
      int k=0; int j=0;
      while (( k< BF.length) && (j< BF.length )) {
        int i=0;
         while (i<j){
                  boolean x = BF1[i].equalsIgnoreCase(BF[k]);
                  if(x==true)
                  { c=1;System.out.println("c= "+c);
                  i=BF.length;}
                  else
                  { i++;System.out.println("c= "+c);} }
         if(c==0)
         { BF1[j]=BF[k];
         k++;j++;c=0;}
        else { j=j; k++;c=0;}
      }
      // Affichage
      for(int h=0; h<j; h++){
      System.out.println(BF1[h] + " ");
        }}}
```

Code source : la base de règle

```
import java.util.Scanner;
public class test {
        public String[] BRp1;
```

```
    public String[] BRp2;
    public  void brt(){
            /* Création_de_la_matrice*/
    Scanner sc = new Scanner(System.in);
    int n;
    System.out.println("entré le nbr de base de régle ");
    n = sc.nextInt();
  String[] BRp1 = new String[n];
  BRp1= new String[n];
  String[] BRp2 = new String[n];
  BRp2= new String[n];
  // Remplissage :
  String f;
  f = sc.nextLine();
  for (int i=0; i< BRp1.length;i++){
    System.out.println("entrer les fait de condition ");
     f = sc.nextLine();
    BRp1[i]=f;
     }
  for (int i=0; i< BRp2.length;i++){
     System.out.println("entrer les fait de conlusion ");
      f = sc.nextLine();
     BRp2[i]=f;
      }
  String[][] BR = new String[n][3];
  for(int i = 0; i < n; i++){
    BR[i] = new String[3];
     }
// Remplissage : ****
  for(int i = 0; i < BR.length; i++){
       for(int j = 0; j < 9; j++){
              BR[i][0] = BRp1[i];
              BR[i][1] = ("=>");
```

```
                BR[i][2] = BRp2[i];
            }
        }
        // affichage **********
    for(int i = 0; i < BR.length; i++){
        for(int j = 0; j < 3; j++){
            System.out.print(BR[i][j] + "");
        }
        System.out.println();
    }}}
```

Code source : moteur d'inférence Chainage avant

```
import java.util.List;
import java.util.*;
import java.awt.*;
public class principal {
        public static void main (String []args){
                base_de_fait  basefaitObject = new base_de_fait();
                basefaitObject.BF();
                test testbrObject = new test ();
                testbrObject.brt();
                Scanner sc = new Scanner(System.in);
  System.out.println("entrer le fait qui vous pouvez le testé ^__^");
            String F;
            F = sc.next();
              int i=0;
              int j=0;
              boolean contain = false;
              for(i=0 ;i<basefaitObject.BF1.length ;i++){
                if( F.equals(basefaitObject.BF1[i])){
                    contain = true ;
              }
```

```
            }
             if(contain) {
      System.out.println("le fait et existe  dans la base de fait");
             }
             else{
      System.out.println("le fait et n'existe pas dans la base de fait");}
    // le moteur d'inférance
             String[] nouveaufait = new String[i];
                 nouveaufait = new String[i]
 //tableau pour les nouveau fait que nous ajoutant depuis la base de régle
                 int k=0 ;
        //boucle pour testé le fait avec les nouveau fait
                 while (F != nouveaufait[k]){
 // boucle pour testé les faits du base de fait avec les faits du partie condition du base de régle

                for (i=0 ;i< basefaitObject.BF1.length;i++){
for ( j=0; j<testbrObject.BRp1.length ; j++ ) {
if(basefaitObject.BF1[i].equals(testbrObject.BRp1[j]))break;{
        nouveaufait[k]= testbrObject.BRp2[j]   ;
                    }
             j++;}
                 }
              System.out.println(nouveaufait[i]);
              k++; }
             System.out.println("le moteur sa marche ");
            } }
```

Principe de l'algorithme

Le mécanisme du chaînage avant est très simple : pour déduire un fait particulier, on déclenche une règle dont les prémisses sont connues dans BF. Sa conclusion est ajoutée à BF comme fait connu. La règle est ensuite désactivée (une règle n'a besoin d'être déclenchée qu'une fois au plus, puisque ses prémisses restent dans BF). L'algorithme tourne ainsi jusqu'à ce que le fait à déduire soit connu ou qu'aucune règle ne soit plus déclenchable [Pierre Ardiri , 2012].

Annexe D : Algorithme optimal et Algorithme d'allocation optimal fiable modifié

Algorithme 1 : Allocation Optimal

Distribution ((t1,….,tp),M)

{initialisation : calcul de valeurs n_i telles que $n_i \times t_i \approx constante$ et $n_1+n_2+\ldots+n_m \leq N$}

1- pour i=1 à m :

2- $n_i=\left\lfloor \frac{\frac{1}{t_i}}{\sum_{i=1}^{m} \frac{1}{t_i}} \times N \right\rfloor$

{Incrémenter itérativement les n_i qui minimisent le temps d'exécution tant que $\sum_{i=1}^{m} n_i$<N}

3- tant que $\sum_{i=1}^{m} n_i$<N :

4- trouver k∈{1,…..,N} tel que tant que $t_k \times (n_k+1)$ =min { $t_i \times (n_i+1)$ }

5- $n_k = n_k+1$

6- renvoyer $(n_1+n_2+\ldots+n_k)$

NB :

M : Le nombre des processeurs.

N : Le nombre des taches.

n_i : La charge du nième processeur.

t_i : Le temps d'exécution du nième processeur.

Principe de l'algorithme 1

Cet algorithme se base sur le principe de faire une distribution d'un ensemble de taches indépendantes sur un ensemble de processeurs et cela en respectant le temps d'exécution de chaque processeur de façon à affecter au processeur ayant le temps d'exécution le plus petit le maximum de tache a exécuté pour obtenir un temps d'exécution optimal et par conséquent obtenir un meilleur équilibrage des charges [A.Legrand and Y.Robert,2005].

Algorithme d'allocation optimal fiable modifié

Algorithm 2 Algorithme d'allocation optimal fiable modifié

Distribution ((t1,….,tp),M)

Distribution ((b1,….,bp),M)

Entrée: q ∈ [1.1, + ∞ [

Calcule M = q Mopt utilisant l'algorithme 1.

Triez le processeur en augmentant ti

Triez le processeur en diminuant bi

Trier le processeur en augmentant biti

X ← 0

pour i = 1: m

si X <N

ni ← min(n-x, $\left\lfloor \frac{M}{ti} \right\rfloor$)

SINON

ni ← 0

X ← X + ni

Où :

ni: La charge du processeur ih.

ti: le temps d'exécution du ième processeur.

bi: la probabilité d'échec.

Mopt: temps d'exécution optimal.

Principe de l'algorithme 2

L'algorithme modifié suppose que le processeur a le plus petit temps d'exécution. sa probabilité de tombé en panne augmente en raison de son utilisation fréquente. Dans ce cas, un processeur dont le temps d'exécution est plus long peut avoir la possibilité de charger une tâche en tant qu'autre processeur [J.Jack Dongarra et al, 2007].