MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

**Faculté des Sciences Exactes et de l'Informatique**
**Département de Mathématiques et d'Informatique**
**Filière : Informatique**

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Ingénierie des Systèmes d'Information**

THEME :

# Le Raisonnement à Partir de Cas dans les Systèmes d'Aide à la Décision Médicale

## Prise en Charge de l'Enfant Victime d'un Accident de la Route

Etudiant(e) : Bourahla Afif

Encadrant(e) : Henni Fouad

Année Universitaire 2016/2017

# Résumé

Le **R**aisonnement à **P**artir de **C**as **(RàPC)** (en anglais **Case-Based Reasoning, ou CBR**) est un type de raisonnement inspiré par le comportement humain qui consiste à utiliser l'expérience pour résoudre les problèmes de la vie quotidienne, à travers la remémoration des situations semblables déjà rencontrées et en les comparant à la situation actuelle pour construire une nouvelle solution qui, à son tour, s'ajoutera à l'expérience. Ce type de raisonnement est basé sur la résolution des problèmes en retrouvant des cas similaires dans la base de cas et en les adaptant au cas considéré.

Dans les différents domaines d'application, le RàPC est devenu une technique très connue. Dans ce travail, il s'agit de proposer un système d'aide à la décision pour la prise en charge d'un enfant victime d'un accident de la route qui est fait en collaboration avec l'établissement hospitalier spécialisé (EHS) de Canastel, Oran-.

**Mots clés :**

RàPC, système d'aide à la décision, accident de la route, urgence pédiatrique

# Dédicaces

*Je dédie ce modeste travail*

*A mes très chers parents, pour leur soutien ; mon père qui s'est sacrifie afin que rien n'entrave le déroulement de mes études, ma mère qui n'a pas cessé de prier pour moi et de m'encourager dans les moments difficiles*

- *A mon cher frère*
- *A mes chères sœurs*
- *A toute la famille*
- *A tous mes amis et touts mes collègues de la promotion d'Informatique (2016 /2017).*

*Merci à tous.*

# Remerciement

*Tout d'abord, louange à « Allah » qui m'a guidé sur le droit chemin tout au long du travail et m'a inspiré les bons pas et les justes reflexes. Sans sa miséricorde, ce travail n'aura pas abouti.*

*Ce projet de fin d'étude s'est déroulé au sein de l'université AbdelHamid Ibn Badis à Mostaganem.*

*En préambule à ce mémoire, je souhaite adresser mes remerciements les plus sincères aux personnes qui ont apporté leur aide et ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.*

*Je tiens à remercier sincèrement et infiniment Monsieur « Henni Fouad», qui, en tant qu'encadreur, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour. « Merci pour tous ses conseils, ses persévérances et ses bienveillances »*

*Je saisie l'occasion pour remercier vivement tout le corps professoral et administratif de la Faculté des Sciences Exactes & Informatique.*

*Mes remerciements aussi aux membres de jury pour l'honneur qu'ils ont fait de bien vouloir analyser ce travail et apporter leurs suggestions.*

*Il m'est particulièrement agréable d'exprimer ma reconnaissance et mess vifs remerciements à mes parents, qui ont toujours su me faire confiance et me soutenir sans compter dans mes études*

*Et enfin, j'adresse une pensé toute particulière à tout ceux qui ont, un jour ou l'autre, croisé mon chemin, que ce soit l'université ou n'importe où dans ce petit monde.*

# Table des matières

# Liste des tableaux

# Liste des figures

# Introduction Générale

L'idée de créer des machines à penser ou machines pensantes revient à la création des premiers ordinateurs. Ce qui donne la naissance à plusieurs systèmes en basant sur la simulation entre le cerveau humain et la machine. Les premiers systèmes « intelligents » reposaient sur l'hypothèse que l'être humain raisonne à partir de règles « SI …. ALORS ». Cela a donné naissance aux systèmes à base de règles ou systèmes déductifs. Ce type de systèmes nécessite une forte connaissance du domaine cible à travers les experts du domaine. De plus, toute modification dans le modèle nécessite un ré implémentation du système ou d'une partie du système.

Pendant les années 80 du siècle dernier, les travaux de Roger Schank [11] ont abouti à un modèle de mémoire et de raisonnement humains qui ne sont pas basés sur les règles. Ce modèle est basé sur la mémorisation des expériences sous forme d'épisodes dans des « scripts » en mémoire. Ces épisodes peuvent être réutilisés lorsque l'humain est confronté à un problème partiellement similaire à une expérience déjà vécue. Ces travaux sont à l'origine de ce qui sera appelé le raisonnement inductif, ou alors le raisonnement par analogie. Le Raisonnement à Partir de Cas (RàPC, en anglais CBR : Case-Based Reasoning) rentre dans le cadre du raisonnement par analogie.

Les premiers travaux sur le RàPC ont eu lieu pendant les années 90 ce qui a permis de donner les bases théoriques du RàPC. Dès lors, plusieurs systèmes basés sur le RàPC ont vu le jour et se sont développés. Le succès des premiers systèmes a ouvert la voie vers l'utilisation du RàPC comme paradigme de résolution de problèmes dans une grande variété de domaines d'application. Le RàPC est utilisé parfois comme synonyme au raisonnement par analogie. En réalité le RàPC est un cas particulier du raisonnement analogique. Le RàPC résout un nouveau problème P en réutilisant ou en adaptant une solution satisfaisante d'un problème P' préalablement mémorisée, pourvu que P soit jugé « similaire » à P'.

Les accidents de la route posent un problème majeur et un défi pour le domaine médical car ce problème prend les vies des dizaines de personnes chaque jour, y compris les enfants. Ces accidents engendrent des conséquences physiques et psychologique parfois de long terme. La prise en charge rapide d'une victime d'un accident de la route, surtout un enfant, est indispensable car le temps de prise d'une décision est un facteur très important qui peut sauver une vie d'un enfant. Parfois les médecins hésitent de prendre une décision puisqu'il n'y a pas assez de données surtout pour les nouveaux médecins qui n'ont pas d'expérience. Un système d'aide à la décision peut combler un manque d'expérience, mais aussi accélérer la prise en charge.

Dès son développement dans les années 1990, le RàPC a constitué un cadre idéal pour la conception de système d'aide à la décision. Le mode de raisonnement utilisée par cette méthodologie, c'est-à-dire le raisonnement par analogie, correspond le plus souvent au mode utilisé dans les systèmes de diagnostic, en particulier médical. Ces considérations ont été à la base du lancement de ce projet en collaboration avec le service de réanimation de l'établissement hospitalier spécialisé (EHS) de Canastel, Oran. Pour rappel, cet établissement

est spécialisé dans la médecine des enfant (pédiatrie) et comporte pratiquement toutes les spécialités du domaine de la pédiatrie.

Ce projet fait partie d'un projet de recherche de type CNEPRU dont l'intitulé est : SP SIAD/DMWS (Surveillance Pédiatrique : Système Interactif d'Aide à la Décision basé sur le Data-Mining et le Web Sémantique). Ce projet est le résultat d'une collaboration du laboratoire LIO (Laboration d'Informatique d'Oran), et l'EHS de Canastel, Oran. Le projet comprend plusieurs facettes, dont celle développée dans le cadre de ce Master.

Ce mémoire est structuré en cinq chapitres. Le premier chapitre expose les principes de base du RàPC ainsi que son cycle de fonctionnement. Ce chapitre détaille également quelques autres aspects liés au développement d'applications basées sur le RàPC. Le second chapitre est consacré à l'utilisation du RàPC dans des applications d'aide à la décision médicale où on présente une variété de systèmes basés sur le RàPC. Ensuite, le troisième chapitre décrit la plateforme jCOLIBRI qui facilite le développement d'applications dans le cadre du RàPC. Le chapitre expose les principales classes que le développeur doit connaitre et comprendre afin de développer une nouvelle application basée sur le RàPC.

Le quatrième chapitre a pour objectif de décrire les aspects fondamentaux derrière le développement de notre application : structure du cas, fonctions de calcul de similarité, etc. Le cinquième chapitre montre quelques expérimentations faites avec l'application mise en œuvre ainsi que la comparaison avec le système jCOLIBRI d'origine. Une conclusion générale et quelques perspectives clôturent ce mémoire.

**Chapitre 01**

# Le Raisonnement à Partir de Cas

## 1- Historique

L'acte de naissance de l'intelligence artificielle correspond à un programme de rencontres organisées à Dartmouth Collège (Hanover, New Hampshire, USA) pendant l'été 1956, à l'initiative notamment de deux jeunes chercheurs qui, dans des registres différents, allaient fortement marquer le développement de la discipline : John McCarthy et Marvin Minsky, le premier défendant une vision purement logique de la représentation des connaissances, le second privilégiant l'usage de représentations structurées (appelées en anglais 'frames') de stéréotypes de situations pouvant inclure différents types d'information. C'est à cette occasion que l'expression 'Artificial Intelligence' (choisie par McCarthy) [8] fut utilisée pour la première fois de manière systématique pour désigner le nouveau champ de recherche.

## 2- Les systèmes à base de règles (systèmes déductifs)

Selon, P. Denning (1986), « Un système expert est un programme conçu pour simuler le comportement d'un humain qui est un spécialiste ou un expert dans un domaine très restreint ». Un système expert encapsule de la connaissance sous forme de règles et de faits et dispose d'un mécanisme d'inférence lui permettant d'utiliser ces connaissances pour résoudre un problème. Une règle est sous la forme « si tel fait est attesté alors effectuer telle action ». Une action peut être l'ajout d'un fait, le retrait d'un fait ou la modification d'un fait existant dans la mémoire de travail. [4]

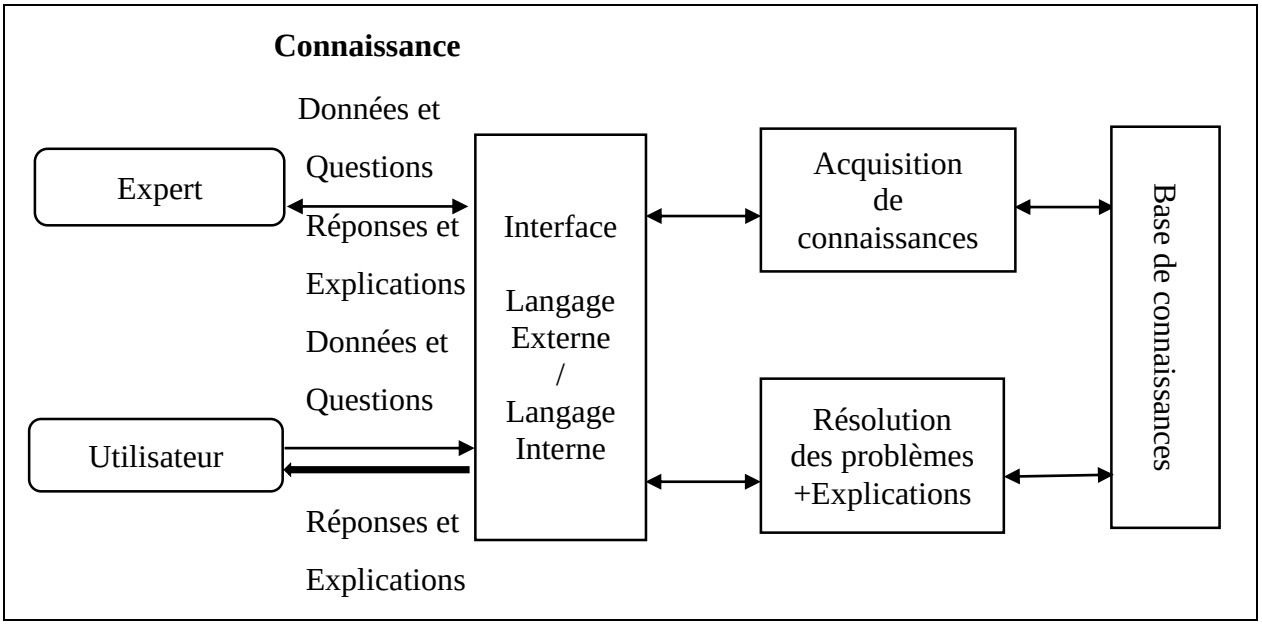


Figure 1.1 : Le schéma général d'un système expert [4]

Ces modèles de raisonnement logique déductif demandent une connaissance forte et explicite du domaine concerné. Malgré la réussite de ces systèmes dans les différents domaines d'application, les systèmes experts ont fait l'objet de plusieurs critiques, notamment celles citées par Roger Schank [5] :

a) La construction de la base de connaissance est difficile et très couteuse en temps, notamment à cause de la complexité et du temps nécessaire pour extraire la connaissance à partir des experts des domaines.

b) Incapacité de prise en charge de problèmes qui ne sont pas couverts par la base de règles utilisée. En général, les systèmes experts qui reposent sur une base de règles sont très utiles si la connaissance qu'ils intègrent et bien formalisée, restreinte, avérée et stable

c) Dans le cas où la base de règles d'un système expert n'intègre aucune fonctionnalité d'apprentissage, toute addition au programme existant nécessite l'intervention d'un programmeur.

## 3- Les travaux de Roger Schank

Les études de R. Schank [11] ont montré l'un des inconvénients majeurs de ces systèmes qui est la rigidité. Puisque l'homme ne raisonne pas toujours à partir des règles. Schank a proposé une vue différente du raisonnement basée sur un modèle inspiré du raisonnement de l'être humain et l'organisation de sa mémoire. Il suggère que l'organisation des connaissances des êtres humains est sous forme de paquets de mémoire contenant des épisodes de notre vie suffisamment significatifs pour être mémorisées. Ces paquets d'organisation de mémoire notés MOP (memory organisation paquet) ne sont pas isolés mais interconnectés par nos perspectives vis-à-vis de la progression normale des évènements appelée « *scripts* » par Schank. Il existe une hiérarchie de MOPs dans laquelle les MOPs de grande taille partagent entre eux des MOPs de petite taille. Si un MOP contient une situation dans laquelle un certain problème a été résolu avec succès et la personne se trouve dans une situation similaire, l'expérience antécédente est rassemblée et la personne en question peut tenter de suivre les mêmes étapes pour atteindre une solution. Ainsi, plutôt que de suivre un ensemble général de règles, la ré-application d'un schéma de solution antérieure satisfaisante, lors d'une situation nouvelle mais similaire, peut résoudre le nouveau problème rencontré. En utilisant ces observations qui concernent le raisonnement humain, Schank a proposé les concepts du raisonnement basé sur la mémorisation et de système expert basé sur la mémorisation qui sont caractérisé par : [5]

a) La base de connaissance utilisée est essentiellement déduite à partir de l'énumération de cas spécifique ou d'expériences.

b) Quand un problème est soumis à un système basé sur la mémorisation, et dans le cas où il s'agisse d'un problème non encore rencontré, le système peut raisonner en utilisant des similarités générales afin d'obtenir la solution.

c) La mémoire des expériences utilisée par le système subit des changements et des ajouts à chaque fois qu'un nouveau cas se présente.

## 4- Le raisonnement par analogie

Le raisonnement analogique (RA) est une composante centrale de l'intelligence humaine. Elle fait partie de la pensée inductive, considérée comme un mécanisme important dans l'apprentissage et la résolution de problèmes. Les analogies classiques représentent la mesure traditionnelle du RA utilisée dans les tests de QI.

## 5- Raisonnement à partir de cas [1]

Le **R**aisonnement **à** partir de **C**as (RàPC, en Anglais CBR : Case-Based Resoning) est un paradigme de résolution de problèmes qui, à bien des égards, est fondamentalement différent des autres approches majeures de l'IA. Au lieu de se fier uniquement aux connaissances du domaine du problème, ou faire des associations le long de relations généralisées entre des descripteurs et des conclusions, le RàPC utilise les connaissances spécifiques des situations problématiques (cas) antérieurement expérimentées. Un nouveau problème est résolu en trouvant un cas antérieur similaire et le réutiliser dans la nouvelle situation problématique. Une deuxième différence importante est que le RàPC est également une approche d'apprentissage progressif et soutenu, puisqu'une nouvelle expérience peut être retenue chaque fois qu'un nouveau problème est résolu. Le champ d'application du RàPC a connu une croissance rapide au cours des dernières années, comme en témoigne sa part accrue de communications lors de grandes conférences, les outils commerciaux disponibles et les applications couronnées de succès dans l'utilisation quotidienne. Le RàPC attire les chercheurs puisque [11] :

- Le RàPC n'exige pas un modèle de domaine explicite donc la collecte de connaissance devient une tache de mémorisation de cas antérieurs.
- L'implémentation est réduite à l'identification de propriétés significatives qui permettent d'écrire un cas : c'est une tâche plus facile que celle qui consiste à créer un modèle explicite.
- Un système basé sur le RàPC peut apprendre par l'acquisition de nouveaux cas. Ceci ajouté aux techniques de bases de données rend la maintenance de larges volumes d'information plus facile.

Pour résoudre un nouveau problème, le RàPC se base sur la résolution de situations antérieures similaires en recherchant dans une base de cas. Pour illustrer ce principe, considérons les exemples qui suivent [1] :

a) **Un médecin** : après avoir examiné un patient dans son bureau, le médecin se rappelle d'un patient qu'il a soigné il y a deux semaines. Ce rappel est généralement causé par une similarité importante au niveau des symptômes. Le médecin diagnostique la situation actuelle en se basant sur la situation antérieure, et en prenant en considération les différences entre les deux cas. Ce nouveau diagnostic peut, à son tour, être utilisé dans le futur pour un nouveau patient.

b) **Un ingénieur de forage** : qui a connu deux situations dramatiques de soufflage, se rappelle rapidement l'une de ces situations (ou les deux) lorsque la combinaison des mesures

critiques correspond à celles d'un cas de soufflage. En particulier, il peut obtenir un rappel à une erreur qu'il a faite lors d'une fuite précédente, et utiliser ceci pour éviter de répéter l'erreur une fois de plus.

### 5.1. Résolution de problèmes par cas [1]

Comme le montrent les exemples ci-dessus, le raisonnement en réutilisant les cas passés est courant pour résoudre des problèmes chez les êtres humains. Cette affirmation est également étayée par les résultats des recherches psychologiques. Plusieurs études ont fourni des preuves empiriques sur le rôle dominant de la spécificité, dans la résolution des problèmes humains. Schank a développé une théorie d'apprentissage et du rappel fondée sur le maintien de l'expérience dans une structure de mémoire dynamique et évolutive. D'autres études ont montré que les gens utilisent les expériences passées comme modèles lorsqu'ils apprennent à résoudre des problèmes, en particulier dans l'apprentissage précoce. Des études de résolution de problèmes par analogie montrent également l'utilisation fréquente de l'expérience passée dans la résolution de nouveaux et différents problèmes. Le RàPC et l'analogie sont parfois utilisés comme des synonymes. Le RàPC peut être considéré comme une forme d'analogie.

Dans la terminologie du RàPC, un cas dénote habituellement une situation problématique, une situation qui a été saisie et apprise de manière à pouvoir être réutilisée dans la résolution des nouveaux problèmes est désignée comme cas passé, cas précédent, cas mémorisé ou cas conservé. En conséquence, un nouveau cas ou un cas non résolu est la description d'un nouveau problème à résoudre.

### 5.2. L'apprentissage dans RàPC

Une caractéristique très importante du RàPC est son couplage à l'apprentissage. Le RàPC est également considéré comme un sous-domaine de l'apprentissage automatique. L'apprentissage en RàPC se produit comme un sous-produit de la résolution de problèmes.

Le RàPC favorise l'apprentissage par l'expérience, puisqu'il est généralement plus facile de conserver une expérience concrète de résolution de problèmes plutôt que d'aller vers une généralisation. Cependant, l'apprentissage dans le RàPC nécessite un ensemble bien élaboré de méthodes afin d'extraire les connaissances pertinentes des expériences dans le but d'intégrer un cas dans une structure de connaissances existante et l'indexer correspondant à des cas similaires.

### 5.3. Principe du RàPC [5]

Le principe du RàPC est de raisonner en interpelant des problèmes résolus pour suggérer des solutions pour un nouveau problème similaire. Les pionniers du RàPC énumèrent quatre hypothèses sur le monde qui nous entoure, et qui représente la base de l'approche du RàPC :

a) *Régularité* : les mêmes actions exécutées dans les mêmes conditions ont tendance à produire les mêmes résultats ou des résultats similaires.

b) *Typicité* : les expériences ont tendance à se répéter.

c) *Consistance* : des changements légers dans une situation engendrent uniquement des changements légers dans l'interprétation et la solution.

d) *Adaptabilité* : quand les évènements se répètent, les différences ont tendance à être légères et il est facile de compenser ces légers changements.

### 5.4. Le cycle du RàPC

Au plus haut niveau de généralité, le cycle général du RàPC peut être décrit par les quatre processus : Retrieve-Reuse-Revise-Retain. Pour résoudre un nouveau problème (nouveau cas) :

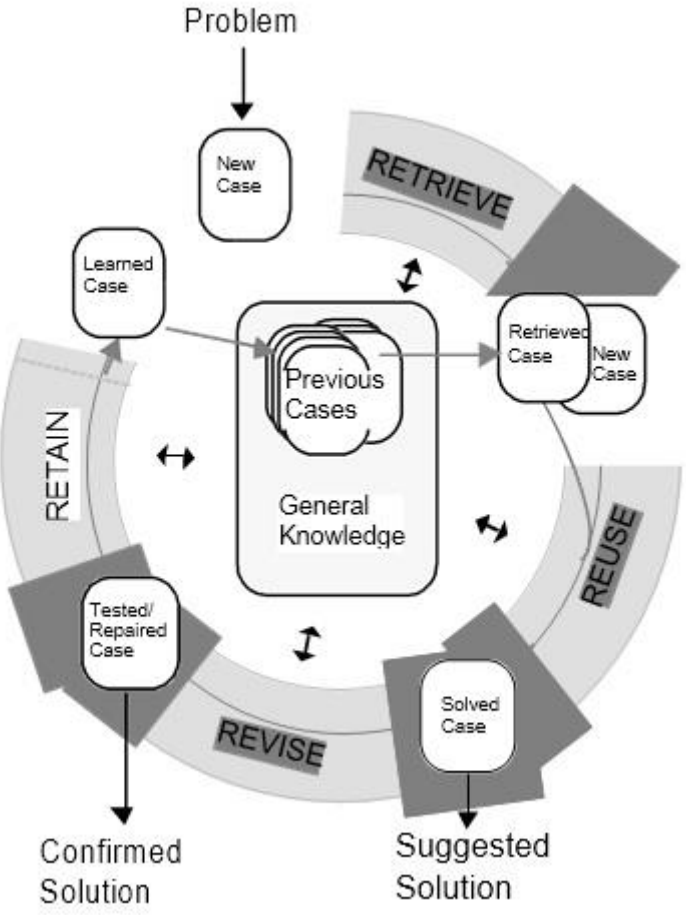


Figure 1.2 : Le cycle du RàPC [1]

**a)** RÉCUPÉRER (RETRIEVE) le ou les cas les plus semblables.

**b)** RÉUTILISER (REUSE) les informations et les connaissances dans ce cas pour résoudre le problème.

**c)** REVISER (REVISE) la solution proposée.

**d)** RETENIR (RETAIN) les parties de cette expérience susceptibles d'être utiles pour la résolution de problèmes futurs.

Comme l'indique la figure 1.2, les connaissances générales sont considérées comme un soutien pour ce cycle, ce soutien peut varier de très faible (ou aucun) à très fort, selon le type de méthode utilisée. Par connaissances générales, on veut dire les connaissances générales dépendant du domaine, par opposition aux connaissances spécifiques incorporées par les cas. Par exemple, diagnostiquer un patient en récupérant et en réutilisant le cas d'un patient précédent, le modèle d'anatomie, les relations causales entre les états pathologiques peuvent constituer les connaissances générales du système RàPC. Un ensemble de règles peut avoir le même rôle.

### 5.5. Problèmes du RàPC [1]

Comme pour l'IA en général, il n'existe pas de méthodes RàPC universelles adaptées à chaque domaine d'application. Le défi dans le RàPC, comme ailleurs, est de trouver des méthodes qui sont adaptées à la résolution de problèmes et l'apprentissage dans des domaines de sujets particuliers et pour des environnements d'application particuliers. Les problèmes fondamentaux abordés par la recherche sur le RàPC peuvent être regroupés en cinq domaines. Un ensemble de solutions cohérentes à ces problèmes constitue une méthode du RàPC :

a) Représentation des connaissances.

b) Méthodes de récupération.

c) Méthodes de réutilisation.

d) Méthodes de révision.

e) Méthodes de rétention.

## 6- Conception d'un système basé sur le RàPC :

Puisque le RàPC est une méthodologie [10] il n'y a pas une implémentation en générale de ses systèmes donc la conception d'un système basé sur le RàPC a besoin de prendre quelques décisions en ce qui concerne les aspects décrits dans ci-dessous.

### 6.1. Représentation des cas [11]

Un cas est une connaissance qui représente une expérience. Il contient une leçon passée et son contexte d'utilisation. Il contient généralement :

- Le problème lequel décrit l'état où ce cas est posé.
- La solution du cas.
- Post-solution qui décrit l'état après la résolution de ce cas.

Les paires (cas, solution) peuvent être utilisées pour résoudre de nouveaux cas. Les solutions et les post-solutions sont utilisées pour évaluer les nouvelles situations. Dans le RàPC, on peut distinguer trois problèmes lorsqu'on veut représenter un cas : définir les attributs qui décrivent le cas, définir la structure pour la description du contenu du cas et proposer une organisation des cas dans la base de cas.

Dans le RàPC, il n'y a pas de représentation prédéfinie donc pour représenter un cas, il faut respecter deux mesures principales : la fonctionnalité et la facilité d'extraire les informations représentées dedans. Un cas peut prendre une de représentations suivantes : [20]

- *La représentation vecteur :* c'est la forme la plus simple ou un cas peut être représenté comme un ensemble de caractéristiques décrivant le problème (attribut- valeur) et la solution où tous les cas ont le même nombre des attributs.

- *La représentation frame ou entité :* les entités fournissent une façon intuitive pour une structuration et une représentation concise d'une connaissance ou en une seule entité, on peut combiner toutes les connaissances nécessaires d'un concept. Chaque entité a un nom et un ensemble d'attributs. Ces attributs peuvent être une valeur simple ou un pointeur vers une autre entité.

- *La représentation orientée objet :* cette représentation ne demande pas d'espace mémoire pour représenter un cas. De plus, elle permet une représentation intuitive des relations ***IS-A, HAS-A*** et ***PART-OF*.** Les cas sont représentés comme une collection d'objets ou chacun est décrit par une paire (attribut-valeur).

- *La représentation textuelle :* cette représentation est utilisée surtout lorsque toutes les connaissances sont disponibles sous forme textuelle. Elle vise à utiliser les données textuelles de façon automatique ou semi-automatique pour améliorer le processus de résolution des problèmes à travers la comparaison des cas.

- *La représentation hiérarchique :* le but de cette approche est de représenter un cas sous différents niveaux de détails avec la possibilité d'utiliser plusieurs vocabulaires.

- *La représentation prédicat :* un prédicat est une relation entre les attributs. Il consiste en une partie condition « Si » et une partie action « Alors ». Un cas peut être représenté par plusieurs prédicats. L'avantage de cette représentation est d'utiliser à la fois les règles et les faits pour représenter un cas.

### 6.2. Indexation des cas

Le problème d'indexation est un problème central et bien ciblé dans le RàPC. Il s'élève à décider de quel type d'index utiliser pour la récupération future, et comment structurer l'espace de recherche d'index. Il s'agit en fait d'un problème d'acquisition de connaissances et devrait être analysé dans le cadre de l'analyse des connaissances du domaine et de l'étape de modélisation du problème. L'indexation des cas comprend l'assignation d'index aux cas pour faciliter leur recherche. Quelques ouvrages de référence en RàPC [11] font des recommandations sur l'indexation. Les index doivent être :

- *Prédictifs* : en ce qui concerne la pertinence du cas ;
- *Reconnaissables* : dans le sens où il doit être clair pourquoi ils sont utilisés ;
- Suffisamment *Abstraits* pour permettre l'élargissement de l'utilisation future de la base de cas ;

- Suffisamment *Concrets* (discriminants) pour permettre une remémoration de cas à la fois précise et efficace.

La sélection des index des cas est faite aussi bien par des méthodes manuelles que par des méthodes automatiques. Le choix manuel des index implique de décider la raison d'être du cas par rapport aux objectifs de l'utilisateur et de décider sous quelles circonstances le cas sera utile. Dans [5], les auteurs déclarent que « *les personnes tendent à être meilleures que les algorithmes automatiques dans le choix des index* ». Néanmoins, de plus en plus de méthodes automatiques d'indexation sont proposées.

### 6.3. L'organisation de la base de cas [5]

Le stockage physique des cas est un aspect très important dans la conception d'un système à RàPC efficace. Le rangement des cas doit être organisé de façon qui permet une remémoration des cas précise et efficace. Donc, une bonne organisation de la base de cas et un bon algorithme de remémoration de cas sont ceux qui produisent le meilleur compromis entre la précision et l'efficacité de la procédure de remémoration de cas. En général, on peut distinguer trois approches principales pour l'organisation de la base de cas :

- *Une organisation plane* : c'est façon la plus simple d'organiser une base de cas, ce qui produit évidemment une structure de cas plane. L'organisation plane impose la remémoration de cas basée sur une recherche cas-par-cas dans la totalité de la base de cas. Par conséquent, pour une base de cas de taille moyenne ou grande, cette organisation produit une procédure de remémoration couteuse en temps donc un système inefficace.

- *Une organisation en clusters* : qui trouve ses origines dans le modèle de mémoire dynamique proposé par Schank, est un type d'organisation de la base de cas dans lequel les cas sont stockés dans des clusters de cas similaires. L'avantage est plutôt une remémoration simple mais elle nécessite un algorithme plus complexe pour l'ajout ou la suppression de cas.

- *Une organisation hiérarchique* : qui trouve ses origines dans le modèle « category-exemplar-memory » développé par Porter & Bareiss, est une organisation obtenue en regroupant les cas ayant des caractéristiques similaires. Chaque cas est associé à une catégorie. Différents degrés d'importance sont assignés à des caractéristiques différentes de cas dans l'appartenance d'un cas à une catégorie. Cette organisation permet une remémoration de cas précise et efficace. D'autre part, sa complexité rend la procédure d'ajout ou de suppression d'un nouveau cas encombrante.

### 6.4. La remémoration de cas [1]

La tâche « Récupérer » commence par une description (partielle) du problème et se termine lorsque la meilleure correspondance avec un cas antérieur est trouvée. Ses sous-tâches sont :

#### 6.4.1. L'identification les caractéristiques (descripteurs)

Pour identifier un problème, il peut s'agir simplement de remarquer ses descripteurs d'entrée, mais souvent - et en particulier pour les méthodes à forte intensité de savoir - une approche plus élaborée est prise, dans laquelle on tente de comprendre le problème dans son contexte. Les descripteurs inconnus peuvent être ignorés ou demandés à être expliqués par l'utilisateur. Pour comprendre un problème, il faut filtrer les descripteurs du problème pour déduire d'autres fonctionnalités de problème pertinentes, afin de vérifier si les valeurs des caractéristiques ont un sens dans le contexte. En plus des descripteurs donnés en entrée, d'autres descripteurs peuvent être déduits en utilisant un modèle de connaissance général, ou en récupérant les descriptions similaires d'un problème de la base de cas et en utilisant les caractéristiques de ce cas comme caractéristiques attendues.

#### 6.4.2. La correspondance initiale (initial match)

La tâche de trouver une bonne ressemblance est généralement divisée en deux sous-tâches : un processus d'appariement initial qui récupère un ensemble de candidats plausibles, et un processus plus élaboré de sélection du meilleur parmi ceux-ci. Cette dernière est la tâche de sélection, décrite ci-dessous. Trouver un ensemble de cas correspondants est fait en utilisant les descripteurs du problème (caractéristiques d'entrée) comme des index à la base de cas d'une manière indirecte.

#### 6.4.3. La sélection

A partir de l'ensemble des cas similaires, une meilleure correspondance est choisie. Cela peut avoir été fait au cours de la première ressemblance mais plus souvent un ensemble de cas est renvoyé à partir de cette tâche. Le meilleur cas correspondant est généralement déterminé en évaluant de plus près le degré de concordance initiale.

#### 6.4.4. Le calcul de similarité [11]

Vue la diversité des applications qui utilisent le RàPC et les différentes approches de représentation des cas, il existe plusieurs propositions pour le calcul de similarité entre deux cas (un nouveau cas et un cas mémorisé). Les sections qui suivent expliquent quelques mesures de similarités :

- *Le plus proche voisin* **:** cette approche implique l'évaluation des similarités entre les cas existants et le nouveau cas introduit en se basant sur la somme pondérée des similarités entre les caractéristiques (attributs) du cas. Le problème avec cette approche est de déterminer les poids des caractéristiques. En plus la récupération devient de plus en

plus lente lorsque la base de cas est volumineuse. Plusieurs implémentations de systèmes basés sur le RàPC utilisent cette méthode pour la récupération des cas similaire, par exemple BROADWAY pour la sélection du modèle des voitures, ANON pour l'évaluation des situations de l'échec d'un plan.

- *Induction* : les algorithmes d'induction (par exemple, ID3) déterminent quelles caractéristiques donnent les meilleurs résultats dans la discrimination des cas et génèrent une structure de type arbre de décision pour organiser les cas en mémoire. Cette approche est utile lorsqu'une fonction à un seul cas est requise en tant que solution et que ce cas dépend d'autres.

- *Induction guidée par la connaissance :* cette méthode applique les connaissances au processus d'induction en identifiant manuellement les caractéristiques des cas qui sont connues ou lesquelles affecter la caractéristique de cas primaire. Cette approche est fréquemment utilisée en conjonction avec d'autres techniques, parce que les connaissances explicatives ne sont pas toujours disponibles pour les grandes bases de cas.

- *Récupération de modèle :* semblable aux requêtes SQL-like, la récupération de modèle renvoie tous les cas qui correspondent à certains paramètres. Cette technique est souvent utilisée avant d'autres techniques, telles que le plus proche voisin, pour limiter l'espace de recherche à une section pertinente de la base de cas.

### 6.5. Réutilisation des cas [1]

La réutilisation de la solution de cas extraite dans le contexte du nouveau cas se concentre sur deux aspects :

a) Les différences entre le passé et le cas actuel.

b) Quelle partie d'un cas retrouvé peut être transférée au nouveau cas.

#### 6.5.1. Copie

Dans les tâches simples de classification, les différences sont abstraites (elles sont considérées comme non pertinentes tandis que les similitudes sont pertinentes) et la classe de solution du cas retrouvé est transférée au nouveau cas comme étant sa classe de solution. C'est un type de réutilisation trivial. Il est clair que ce genre de réutilisation est très restreint.

### 6.5.2. Adaptation [11]

Quand un cas similaire est trouvé, un système à base du RàPC doit souvent faire l'adaptation de la solution stockée dans les cas remémorés pour l'ajuster au nouveau cas. L'adaptation est basée sur les différences entre le cas passé et le nouveau cas et applique des règles pour que ces différences soient prises en considération lorsque le système donne la solution appropriée. En général il existe deux types d'adaptation dans le RàPC :

- *L'adaptation structurelle* : ou des règles d'adaptation dépendante du domaine sont établies à l'avance et appliquées directement à la solution du cas passé. Ce type d'adaptation est utilisé dans les systèmes JUDGE (système RàPC dans le domaine juridique) et CHEF (système RàPC pour la proposition de recettes de cuisine).
- *L'adaptation dérivationnelle* : elle réutilise les algorithmes et les méthodes qui ont généré la solution du problème remémoré afin de générer une solution pour le nouveau cas. Il est bien évident que l'adaptation dérivationnelle est plus difficile car elle engendre la mémorisation des algorithmes qui ont généré les solutions antérieures.

Une procédure d'adaptation complexe rend difficile la construction et la maintenance d'un système RàPC et peut aussi affecter la fiabilité du système et la confiance de l'utilisateur. Par conséquent, dans plusieurs systèmes RàPC, l'adaptation est faite par l'utilisateur. Dans [5] on reporte que : « *dans un système bien conçu, les utilisateurs ne perçoivent pas l'adaptation manuelle comme étant un aspect négatif* ».

## 6.6. Révision des cas [1]

Lorsqu'une solution de cas générée par la phase de réutilisation ne correspond pas à la solution attendue, on peut parler d'un échec. Sinon cette phase est appelée révision de cas et se compose de deux tâches :

### 6.6.1. Évaluer la solution

La tâche d'évaluation prend le résultat de l'application de la solution dans l'environnement réel. Il s'agit généralement d'une étape en dehors du système RàPC, car elle - au moins pour un système en fonctionnement normal - implique l'application d'une solution suggérée au vrai problème. Les résultats de l'application de la solution peuvent prendre un certain temps pour apparaître, selon le type d'application. Dans un système d'aide à la décision médicale, un traitement peut prendre de quelques heures jusqu'à plusieurs mois. Le nouveau cas généré reste disponible dans la base de cas pendant une période intermédiaire, mais il doit être marqué comme étant un cas non évalué. Une solution peut également être appliquée à un programme de simulation qui est capable de générer une solution correcte. Ceci est utilisé dans CHEF, où une solution (c'est-à-dire une recette de cuisine, appelée un plan) est appliquée à un modèle interne supposé être assez fort pour donner la rétroaction nécessaire pour la réparation de la solution.

### 6.6.2. Réparer le défaut

La réparation de cas implique la détection des erreurs dans la solution actuelle et la récupération ou la génération des explications pour celles-ci. Le meilleur exemple est le système CHEF, où les connaissances causales expliquent pourquoi certains objectifs du plan de solution n'ont pas été atteints. CHEF apprend les situations générales qui causeront les échecs en utilisant une technique d'apprentissage basée sur l'explication. C'est inclus dans une mémoire de défaillance qui est utilisée dans la phase de réutilisation pour prédire les éventuelles lacunes des plans. Cette forme d'apprentissage déplace la détection des erreurs d'une manière post hoc au plan de la phase d'élaboration où les erreurs peuvent être prédites, manipulées et évitées. Une deuxième tâche de la phase de révision est de réparer la solution. Cette tâche utilise les explications de défaillance pour modifier la solution de telle manière que les défaillances ne se produisent pas. Par exemple, le plan échoué du système CHEF est modifié par une réparation qui ajoute des étapes au plan qui assurera que les causes des erreurs ne se produiront pas. Le module de réparation possède une connaissance causale générale et une connaissance du domaine afin de compenser les causes d'erreurs dans le domaine. Le plan révisé peut alors être conservé directement (si la phase de révision assure son exactitude) ou il peut être évalué et réparé à nouveau.

## 6.7. Retenir un cas – Apprentissage [11]

C'est le processus d'incorporation de ce qui est utile de retenir de la nouvelle résolution de problème dans les connaissances existantes. L'apprentissage tiré du succès ou de l'échec de la solution proposée est déclenchée par le résultat de l'évaluation et la réparation éventuelle. Il s'agit de choisir l'information du cas à retenir, sous quelle forme le conserver, comment indexer le cas pour une récupération ultérieure à partir de problèmes similaires, et comment intégrer le nouveau cas dans la structure de la mémoire.

Dans le RàPC, la base de cas est mise à jour peu importe la façon dont le problème a été résolu. S'il a été résolu par l'utilisation d'un cas précédent, un nouveau cas peut être construit ou l'ancien cas peut être généralisé pour ce cas également. Si le problème a été résolu par d'autres méthodes, y compris par l'utilisateur, un nouveau cas devra être construit. En tout état de cause, une décision doit être prise sur la source de l'apprentissage. Des descripteurs pertinents de problème et des solutions aux problèmes sont des candidats évidents. Mais une explication ou une autre forme de justification de la raison pour laquelle une solution est une solution au problème peut également être marquée pour inclusion dans un nouveau cas.

## 7. Quelques projets basés sur le RàPC [11]

**CHEF :** Ce logiciel se propose de réaliser des recettes de cuisine en utilisant en se basant sur le RàPC. L'utilisateur indique au programme les aliments dont il dispose et CHEF cherche à élaborer une recette à partir de ces ingrédients. Pour ce faire, il opère comme tout système RàPC : il possède une très grande base de cas (de recettes valides) et cherche à créer, par application du cycle du RàPC, une nouvelle recette contenant les ingrédients choisis. CHEF a la particularité de tenir compte des échecs et de les sauvegarder afin d'éviter de les reproduire. Lorsque les contraintes imposées par l'utilisateur sont trop fortes et qu'aucune solution n'a été trouvée, CHEF donne une explication indiquant les raisons de l'échec.

**PERSUADER :** Ce logiciel est un outil de gestion de conflits basé sur le RàPC. Il fonctionne sur le principe de négociation/médiation. Il est capable de fournir des solutions documentées pour la résolution de problèmes de groupe. PERSUADER fait en sorte de pouvoir construire un règlement mutuellement convenu entre les différents acteurs de la dispute.

**SWALE :** Le projet SWALE cherche à fournir des explications à des situations anormales. Il apporte notamment des explications sur les causes de la mort des animaux ou des hommes. Le programme va par exemple comparer la mort inattendue d'un cheval de course très connu en pleine force de l'âge à la mort d'un cycliste due à une consommation excessive de produits dopants.

## 8. Les plateformes de développement

Le succès des applications basées sur le RàPC a poussé plusieurs groupes de recherche à développer des plateformes qui permettent de développer des applications qui utilisent le principe du RàPC. Il s'agit le plus souvent de plateformes génériques dans lesquelles tout est à redéfinir. La plateforme implémente en général le cycle du RàPC et fournit quelques fonctionnalités de base. Le développeur peut ajouter les spécificités de son application (par exemple de nouvelles fonctions de calcul de similarité). Les sections qui suivent décrivent deux plateformes développées dans des laboratoires universitaires et qui sont proposées en « open-source ».

### 8.1. myCBR Workbrench

Le projet myCBR a été développé par un groupe de chercheurs du « German Reasearch Center for Artificial Intelligence » (DFKI) [5]. La motivation derrière l'implémentation était le besoin d'un outil compact et facile à utiliser pour la construction d'applications prototypes basées sur le RàPC pour l'enseignement, la recherche et de petits projets industriels. myCBR Workbrench fournit une interface graphique simple pour modéliser une variété de foncions de mesure de similarité et pour évaluer la qualité de la remémoration. myCBR Workbench s'installe comme un « plugin » dans l'environnement de développement « Protégé »[1].

[1] http://www.mycbr-project.net

myCBR Workbrench a été utilisé dans plusieurs projets européens qui rentrent dans le cadre du mégaprojet ESPRIT. Depuis 2007, cette plateforme a fait l'objet de dizaines de publications et plus d'une dizaine de thèses de doctorat.

### 8.2. COLIBRI Studio [5]

Le projet COLIBRI est un produit du groupe de recherche GAIA[2] (*Group of Artificial Intelligence Applications*) de l'université Complutense de Madrid, Espagne. COLIBRI est une plateforme pour le développement de logiciels basés sur le RàPC. Le but du projet COLIBRI est de fournir l'infrastructure nécessaire pour développer de nouveaux systèmes basés sur le RàPC ainsi que les composants associés. COLIBRI est conçu pour supporter le développement d'une large collection de systèmes basés sur le RàPC : système de RàPC standards, systèmes de RàPC guidés par les connaissances, systèmes de RàPC de conseil, et les applications de RàPC distribuées. Il intègre également des outils d'évaluation, de maintenance et de visualisation de la base de cas. Plusieurs composants de COLIBRI ont été développés par des groupes de recherche externes au groupe GAIA.

jCOLIBRI Studio est une version orientée objet de COLIBRI développée en Java et qui s'installe comme un « plug-in » sous l'environnement de programmation Eclipse. La plateforme jCOLIBRI est réutilisable et extensible dans le sens où le développeur peut facilement réutiliser le code source et éventuellement faire des extensions de classes afin de construire un nouveau système.

La plateforme jCOLIBRI Studio a également été utilisée dans des projets d'envergure et on compte quelques dizaines de publications reliées à cette plateforme. Contrairement à myCBR qui a une orientation pédagogique, jCOLIBRI est beaucoup plus orienté vers l'apprentissage automatique.

## 9. Conclusion

Ce chapitre avait pour objet de définir les principes du RàPC dans le contexte de l'IA. Un historique a permis de tracer l'avènement du RàPC comme paradigme de résolution de problèmes. Beaucoup de recherches ont permis aux concepts du RàPC de se développer et des systèmes basés sur le RàPC ont vu le jour. Certains sont utilisés dans des projets de grande envergure.

L'application du RàPC touche des domaines divers, mais dans le cadre de ce projet on s'intéresse à l'utilisation du RàPC dans les systèmes d'aide à la décision médicale. En effet, la nature de résolution de problème en médecine, le diagnostic, ressemble beaucoup au principe du RàPC. Ceci a donné lieu à plusieurs projets qui s'attaquent à une variété d'applications médicales en utilisant le RàPC. Le but du chapitre suivant est justement de montrer l'importance de l'utilisation du RàPC dans des applications d'aide à la décision médicale.

[2] https://gaia.fdi.ucm.es/research/colibri

**Chapitre 02**

# Application du RàPC dans le Domaine Médical

## 1- Introduction

Les cliniciens ou bien les médecins ont une expérience acquise à travers de nombreuses années d'exercice. À titre d'exemple, lorsqu'un clinicien expérimenté est confronté à un nouveau problème (symptômes qui ne sont pas familiers), le clinicien pourrait commencer à analyser l'ensemble de la situation et essayer de diagnostiquer en s'appuyant sur les connaissances, mais aussi en profitant des situations antérieures déjà diagnostiquées correctement. Cette tâche peut prendre beaucoup de temps et peut conduire à ne pas trouver un bon diagnostic. Les situations antérieures constituent une aide considérable dans le travail quotidien des praticiens de la santé. De plus, il est courant, dans le cas de manque d'expérience, de demander à des collègues leurs avis. Le fonctionnement du processus de diagnostic médical ressemble beaucoup au schéma du cycle du RàPC présenté au chapitre précédent. D'une manière plus générale, ce mode de fonctionnement, basé sur l'expérience est retrouvé dans pratiquement tous les domaines de diagnostic (médical, industriel, ...).

Cette similitude a poussé plusieurs groupes de recherche à s'appuyer sur le RàPC dans la conception de systèmes d'aide à la décision médicale. Il est clair que les travaux présentés dans ce chapitre ne sont que des échantillons, et qu'en réalité l'utilisation du RàPC dans ce domaine ne cesse de gagner du terrain.

## 2- Le RàPC comme un système d'aide à la décision pout le diagnostic du Cancer [14]

L'objectif de ce travail de recherche était de développer un système basé sur le RàPC pour l'identification des patients atteints de différents types de cancer en se basant sur les « microarrays ».

### 2.1. Microarray d'ADN

Un microarray est également connu sous le nom de puce d'ADN ou de bio chip est une collection de taches d'ADN microscopique attaché à une surface solides. Les scientifiques utilisent des microarrays d'ADN pour mesurer les niveaux d'un grand nombre de gènes simultanément. Ou pour génotyper de multiples régions d'un génome. Il existe différentes sortes de microarrays comme CGH tableaux (Comparative Génome Hybridation), et les réseaux d'expression. La technologie des microarrays est un élément critique pour l'analyse génomique et permet l'étude de la caractérisation moléculaire de l'expression de l'ARN. L'analyse par microarray a permis de caractériser les mécanismes moléculaires qui causent Plusieurs cancers. En mettant l'accent sur la leucémie, l'analyse des microarrays a facilité l'identification de Certains caractéristiques de gènes dans les différentes variantes de la leucémie. Les spécialistes

du cancer remarquent l'importance de l'identification des gènes associés à chaque type de cancer afin d'établir les traitements les plus efficaces pour les patients.

### 2.2. Le RàPC pour la classification des informations du Microarray

Le système RàPC développé reçoit des données de l'analyse des puces comme il est responsable de classer les individus selon les données existantes. Le RàPC utilise les cas passés pour résoudre de nouveaux problèmes. Les cas vont passer sous le cycle du RàPC, Les algorithmes sélectionnés pour la remémoration des cas devraient rechercher dans la base des cas et sélectionnez le problème et la solution qui correspondent le plus à la nouvelle situation. La phase de réutilisation commence, ou les solutions récupérées et regroupées. Après, la phase de révision ou l'expert fait une révision pour la solution proposé. Finalement la phase rétention qui permet au système d'apprendre à travers les expériences passées dans les phases précédentes et une mise à jour pour la base des cas. La figure 2.1 montre un diagramme des techniques appliquées aux différentes étapes du cycle du RàPC.

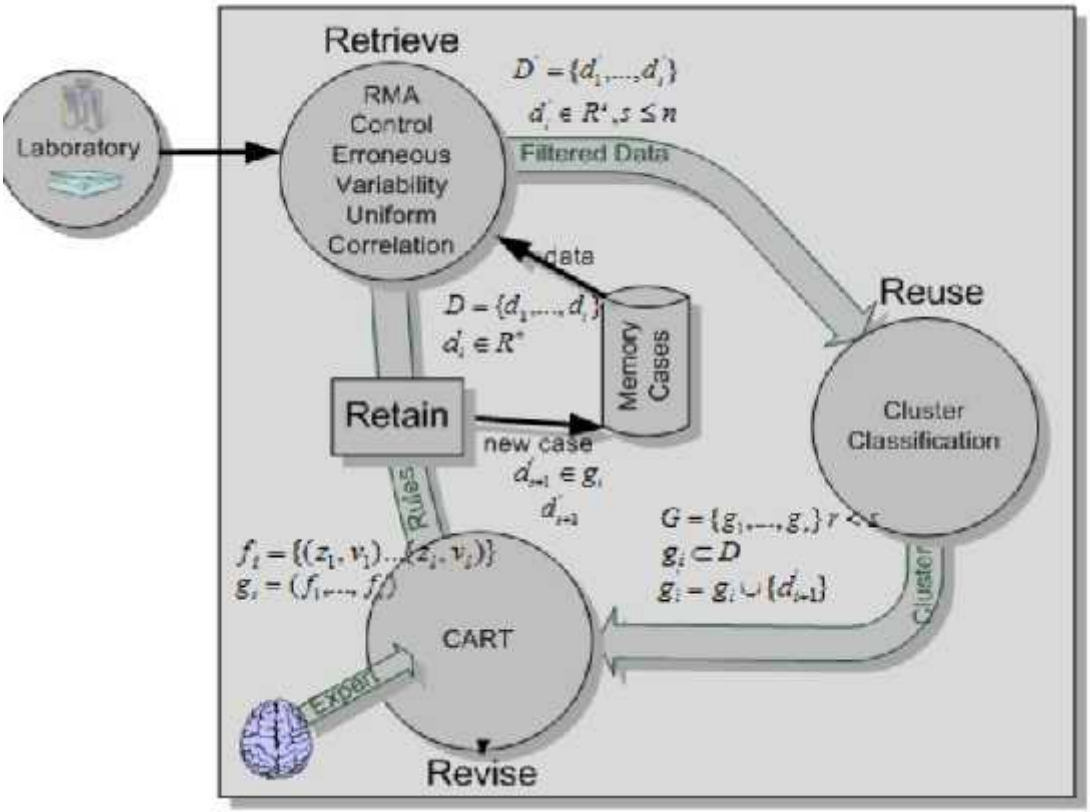


Figure 2.1 : la représentation des différentes étapes du Système RàPC développé [14]

### 2.3. Récupérer

En principe, seuls les cas similaires au problème actuel sont récupérés souvent à cause de leur performance. Tout d'abord, un prétraitement des données est effectué à l'aide de l'algorithme RMA qu'est fréquemment utilisé pour le prétraitement des données d'Affymetrix Microarray.

### 2.4. Réutilisation

Une fois les sondes filtrées et normalisées, elles produisent un ensemble de valeurs. L'étape suivante consiste à effectuer le regroupement des individus en fonction de leur proximité en fonction de leurs sondes. Étant donné qu'il n'est pas possible d'établir un nombre fixe de groupes dans les phases initiales, en tenant compte de la nécessité pour le système d'initier le regroupement sans classification, une technique de classification non supervisée a été utilisée. Puisque dans ce cas le nombre de clusters est inconnu. Ils ont décidé d'utiliser une variation du réseau de neurone SOINN, appelé ESOINN.

### 2.5. Réviser et retenir

Comme la figue 3 montre la révision est faite à l'aide d'une intervention d'un expert qui détermine la correction des groups assignés par le système RàPC. Dans l'étape de révision. Les informations sont discrétisé en cinq niveaux [0,0.25, 0.5, 0.75, 1] et l'extraction de connaissance en utilisant l'algorithme CART. Enfin, l'expert assigne chaque individu à son propre group.

Néanmoins, le système fournit une révision temporelle automatique pour les cas remémorés. En la phase retenir le système calcule le taux des cas bien classés parmi les cas retenus pour le problème courant. Si le taux d'une classe est supérieur de seuil, le système détermine que la classe est bien classé et les deux : cas et connaissance obtenue vont être stocké dans la base des cas. Cette décision doit être confirme par l'expert.

### 2.6. Résultat d'application du système

Le système a été utilisé pour identifier les variables important pour chaque variété de cancer de sang, réduire la dimension des informations originales et classifier les patients. Il été appliqué sur une base de donnée contient 212 patients adultes avec 5 type de leucémie. La classification des variables après le filtrage était similaire par rapport à la classification faite par les experts de laboratoire.

## 3- Le RàPC dans le diagnostic du stress [18]

Pour la majorité des gens, les investigations médicales ont identifié que la température du doigt (en Anglais finger temperature FT) à une forte corrélation avec le stress. Le système se compose d'une thermistance, détectant la température du doigt. Une phase d'étalonnage contribue à établir un profil de stress de l'individu. Il est utilisé comme protocole standard dans les cliniques afin de prendre les mesures. Le protocole comprend différentes conditions. En 6 étapes, ils sont comme suit : ligne de base, respiration profonde, violence verbale, relaxation, stress mathématique, et se détendre. Les étapes de la tension du diagnostic à l'aide de la température du doigt (FT) et le RàPC est illustré à la figure 2.2.

### 3.1. L'extraction des attributs du signal du FT

L'extraction des attributs a une importance majeure pour donner une classification exacte [7]. Pour déterminer les caractéristiques importantes, le système utilise 15 minutes (temps, température) dans 3600 échantillons, conjointement avec d'autres composés numériques (c'est à-dire l'âge, la température ambiante, heures depuis le dernier repas, etc.) et des mesures symboliques (sexe, nourriture et boissons prises, sommeil la nuit, etc.). Selon une discussion plus approfondie avec les cliniciens, la dérivée de chaque étape de la mesure FT (à partir de la phase d'étalonnage) est utilisée pour introduire un "degré de changements" comme une indication de changement de FT. Une valeur d'angle faible, par exemple zéro ou proche de zéro, indique qu'il n'y a pas de changement ou une température de doigt stable. Une haute valeur d'angle positive indique une augmentation de FT, tandis qu'une valeur négative angle, par exemple -20, indique une chute de FT. Le signal total est divisé en 12 parties chacune avec une minute d'intervalle de temps. Un cas est formulé avec 19 caractéristiques au total dont 17 caractéristiques sont extraites du signal de capteur (température de départ, température de fin, température minimale, température maximale et différence entre plafond et plancher) et 2 sont les caractéristiques relatives à la personne (sexe, heures depuis le dernier repas). Ce nouveau cas formulé est alors donné en entrée dans le cadre d'un cycle RàPC du système pour aider au diagnostic de stress [18].

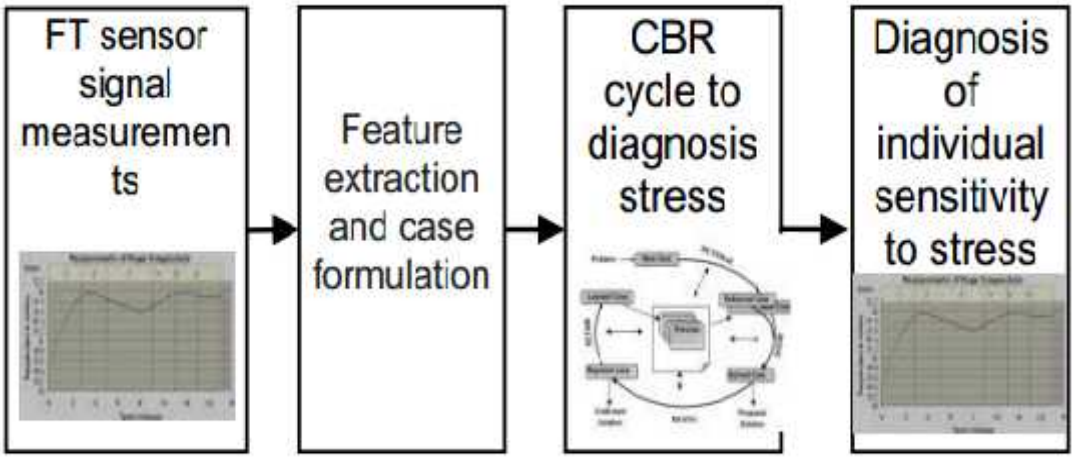


Figure 2.2 : les étapes dans le diagnostic du stress en utilisant FT [18]

### 3.2. Remémoration du cas [19]

La remémoration de cas est la phase la plus importante de cycle du RàPC surtout dans le domaine médical car l'absence d'un cas peut donner une décision informelle. L'exactitude du système de diagnostic dépend du stockage des cas et la remémoration des cas similaire. La similarité entre l'ancien cas ($S_f$) et le nouveau cas ($C_f$) est calculée en utilisant un algorithme de plus proche voisin. Cet algorithme donne une liste ordonnée contenant les cas les plus similaires selon l'équation :

$$\textbf{Similarity(C,S)} = \sum_{f=1}^{n} w_f \ * \ \textbf{sim}(C_f, S_f). \ [18]$$

Où **C** est le nouveau cas, **S** est le cas mémorisé, **w** est le poids défini par un expert, **n** est le nombre des attributs pour chaque cas, **f** est l'index d'individu et **sim($C_f$,$S_f$)** est la similarité locale pour l'attribut **f** de cas **C** et **S**.

### 3.3. Réviser et retenir

Quand le cas plus similaire est trouvé, un expert prend la décision selon les informations données par le système et si le cas est nouveau et intéressant il décide de l'ajouter à la base de cas.

## 4- RàPC pour le traitement du stress [18]

Le principe de base dans un système de biofeedback est qu'un patient puisse clairement voir l'évolution de son état à travers une courbe par exemple. Dans le contexte du traitement du stress ceci s'appelle la rétroaction. La rétroaction peut entraîner le corps et l'esprit de manière meilleure avec une réponse biologique. Des discussions avec cliniciens ont montré que la plupart des applications de biofeedback comprennent trois phases illustrées dans la figure 2.3.

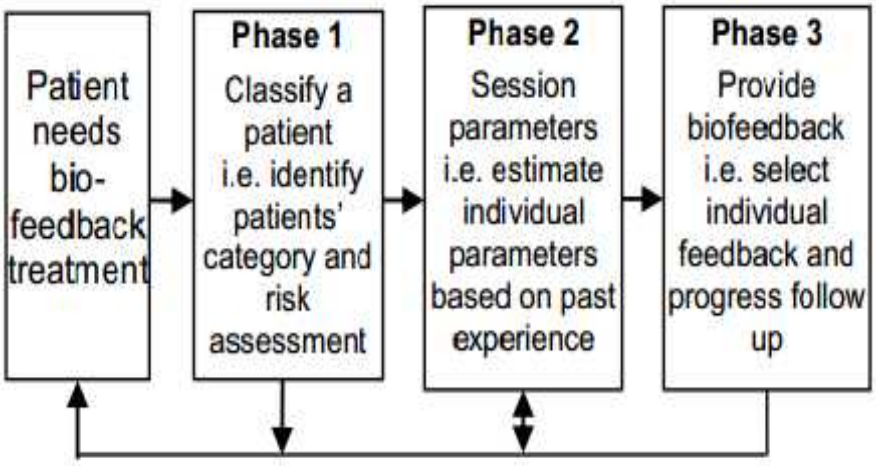


Figure 2.3 : Architecture générale (à trois phases) du système de biofeedback [18]

Analyser et classer un patient et faire une évaluation de risque, déterminer les niveaux et les paramètres individuels, adapter et commencer la formation de biofeedback. Si le clinicien n'utilise que les lectures des capteurs affichés. La classification est fortement basée sur l'expérience. Dans ce cycle représenté à la figure 2.4.

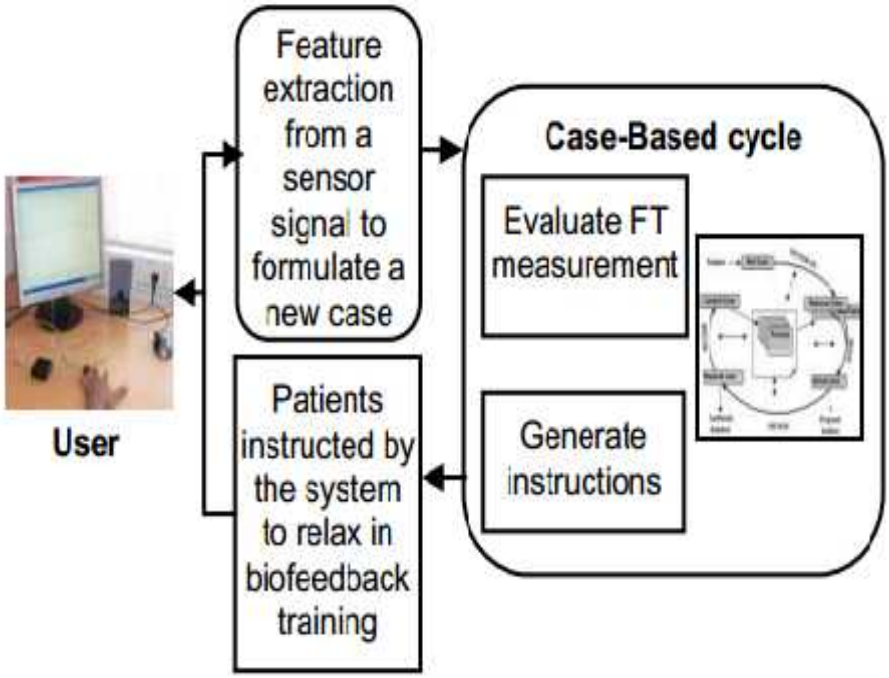


Figure 2.4 : les étapes de cycle de traitement biofeedback [18]

Un utilisateur connecte un capteur à son doigt et peut voir les changements de FT pendant plusieurs instructions dans la formation de relaxation. Les mesures FT sont effectuées en temps réel et toutes les 2 minutes le système évalue la dernière mesure de 2 minutes et si nécessaire et génère des instructions pour le patient. Un cycle du RàPC est appliqué à la formation de biofeedback dans la gestion du stress. Ce temps d'entraînement est souple, ce qui signifie qu'un patient peut choisir la durée de formation entre 6 minutes (comme minimum) à 20 minutes (maximum).

Le système génère des commentaires avec les suggestions appropriées toutes les 2 minutes si nécessaire. Ainsi, pour chaque individu, le biofeedback est formulé avec un vecteur de caractéristiques selon un signal biomédical (c'est-à-dire avec une mesure FT de 2 minutes) dans la partie conditionnelle et une suggestion de relaxation comme une solution.

### 4.1. La remémoration [18]

Un nouveau cas de biofeedback est comparé aux cas précédemment résolus en appliquant une correspondance basée sur un algorithme de similarité floue (fuzzy similarity) et affiche le résultat en retour. Ou chaque surface d'ensemble (m1, m2 et om) fuzzy (floue) est calculé. L'équation de similarité est définie [6] :

$$\mathbf{sim\ (C_f, S_f) = s_f(m1,m2) = max(om/m1, om/m2)}\ [6]$$

Où $S_f$ est un cas mémorisé, $C_f$ est le nouveau cas, m1, m2 et om sont les surfacee de chaque ensemble flou. Ici La rétroaction est définie par une paire, c'est-à-dire qu'elle présente une évaluation de la mesure FT et une recommandation pour le prochain entraînement. Cette rétroaction générée est ensuite proposée comme solution.

### 4.2. Réviser et Retenir

Le clinicien par la suite examine les cas proposés et prend une décision finale pour le traitement du patient. Le système est considéré comme une seconde opinion pour améliorer l'aspect physique et psychologique du patient.

## 5- Conclusion

A travers les systèmes présentés dans ce chapitre, il est clair que l'application du RàPC dans l'aide à la décision médicale gagne beaucoup de terrain. La diversité des applications montre aussi que la méthodologie du RàPC est solide et peut répondre dans des modèles de problèmes très variés.

Les applications présentées dans ce chapitre sont loin d'être représentatives des variétés de projets qui ont apporté des solutions à divers domaines d'application en médecine. Il est clair que plus une méthodologie gagne des champs d'applications plus les chercheurs vont l'adopter et tenter d'élargir son champ d'application.

Le but de ce chapitre était justement d'arriver à montrer le succès de l'application du RàPC dans les systèmes d'aide à la décision médicale. L'objectif derrière cela est de proposer un système basé sur le RàPC pour l'amélioration de la prise en charge de l'enfant victime d'un accident de la route.

**Chapitre 03**

# La Plateforme de Développement « jCOLIBRI »

## 1- Introduction

Grace au succès des premières applications basées sur le RàPC, cette méthodologie s'est rapidement imposée comme sous domaine de l'intelligence artificielle. Le cycle du RàPC a été formalisé et ses principales étapes ont constitué un champ de recherche intense. Cet élan de recherche et d'applications basées sur le RàPC a poussé quelques laboratoires à développer des plateformes qui permettent un développement rapide de nouvelles applications qui s'appuient sur cette méthodologie.

Les plateformes existantes sont génériques : elles implémentent le cycle de base du RàPC ainsi que quelques fonctions simples de calcul de similarité. Si la plateforme est exploitée en boite blanche, le développeur peut ajouter des fonctions plus adaptées à son application. Le projet « COLIBRI » fait partie de cet effort et propose une plateforme qui supporte le RàPC et qui est distribuée en licence libre.

## 2- jCOLIBRI

En 2002, Belen Diaz-Agudo a développé une architecture sous le nom COLIBRI « l'acronyme de **C**ases and **O**ntology **L**ibraries **I**ntegration for **B**uilding **R**easoning **I**nfrastructures » [9]. jCOLIBRI est une plateforme réalisée et entretenue par le laboratoire de recherche GAIA de l'université Compultence de Madrid. Elle est développée en langage de programmation Java et destinée aux développeurs dans le domaine du RàPC, lesquels peuvent définir et implémenter des modules supplémentaires. jCOLIBRI est disponible en Open Source [2]. Il existe deux versions majeures du Framework jCOLIBRI [13] :

- **jCOLIBRI version 1** : cette version est une boite noire et elle est recommandée aux non-développeurs qui veulent créer des systèmes RàPC sans programmer. Elle inclut une interface graphique qui guide l'utilisateur durant l'implémentation du système.

- **jCOLIBRI version 2.1** : cette version est destinée aux développeurs Java comme une boite blanche pour inclure jCOLIBRI dans des applications basées sur le RàPC. Les sections qui suivent concernent la version jCOLIBRI 2.1.

## 3- L'architecture de jCOLIBRI [13]

L'architecture de la version 2.1 de jCOLIBRI est complètement une boite blanche où l'idée d'arrière est de permettre aux développeurs d'intégrer jCOLIBRI dans des applications qui utilisent le RàPC. La figure 3.1 ci-dessous donne une vue globale de cette architecture. Elle est destinée aux développeurs et fournit une architecture capable d'intégrer les différentes techniques du RàPC. Cette couche fournit le code de base de conception d'un système basé sur le RàPC qui est extensible et réutilisable par des programmeurs. jCOLIBRI offre un ensemble de fonctions et de classes Java de base. De plus, les programmeurs peuvent inclure de nouvelles classes et fonctions pour couvrir des besoins particuliers.

Les cas dans cette version sont représentés sous forme de Java Beans. La persistance du cas est gérée en utilisant le package Hibernate. Hibernate est une implémentation de Java Data Objects (JDO) où les Java Beans seront stockés automatiquement dans une base de données relationnelle ; soit dans une ou plusieurs tables. Java Beans et Hibernate sont des technologies noyaux dans la plateforme Java 2 Entreprise Edition.

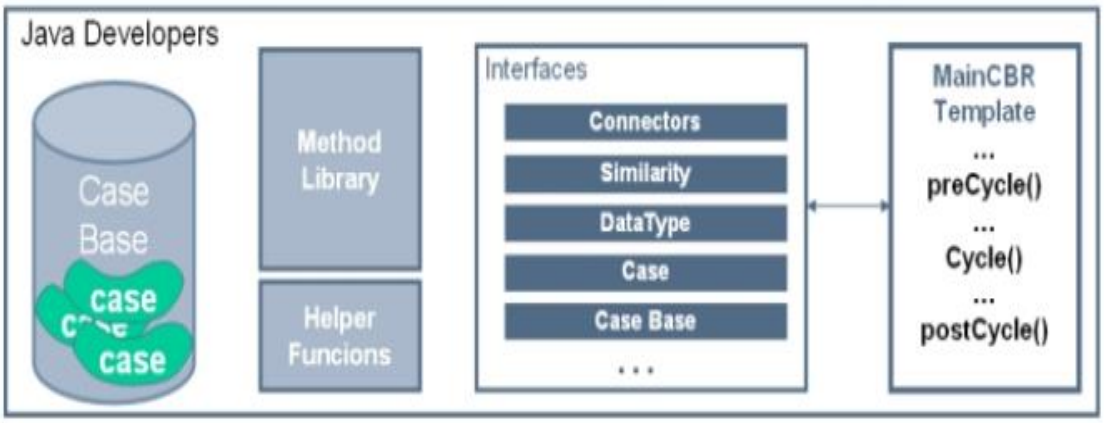


Figure 3.1 : L'architecture de jCOLIBRI

## 4- Les interfaces principales de jCOLIBRI

Pour bien implémenter un cycle du RàPC et de façon efficace les fondateurs de jCOLIBRI mettent à la disposition des développeurs des interfaces qui doivent être respectées afin de concevoir un nouveau système basé sur le RàPC. Ces interfaces représentent la hiérarchie d'un système à RàPC. Elles représentent le noyau du système. Les sections suivantes présentent les principales interfaces.

### 4.1. Interface StandardCBRApplication

Cette interface est considérée comme l'interface de configuration localisée dans le package « cbraplications ». Chaque nouvelle application doit implémenter cette interface qui définit les méthodes standards d'une application à base du RàPC. On peut distinguer quatre méthodes :

- *configure()* : méthode de configuration de l'application où on doit introduire les classes gérant l'application comme : les connecteurs à la base de données, la structure de la base de données, la structure de la base des cas et même définir les fenêtres principales de l'application.
- *precycle()* : initialise l'application ; dans cette étape les cas vont être lus et organisés dans la base de cas. Elle retourne la base de données avec les cas dedans. Le « precycle » est exécuté une seule fois.
- *cycle(CBRQuery query)* : exécute les quatre étapes du cycle du RàPC pour la requête *query* en utilisant des fonctions définies par l'utilisateur. Le cycle est exécuté plusieurs fois selon les taches voulues par l'utilisateur.
- *postCycle()* : cette fonction représente la fin de l'application du cycle du RàPC. Cette fonction ferme les connecteurs à la base de données.

En cas d'échec d'une fonction une exception d'exécution « ExecutionException » est envoyée. Cette exception est bien définie dans la classe « ExecutionException » qui hérite de la classe « java.lang.Exception ». Cette classe crée un nouveau ExecutionException avec un message spécifiant en détail l'exception en utilisant la méthode « ExecutionException (String msg) ». En plus, elle donne la cause de l'exception à travers la méthode « ExecutionException (Trowable th) ».

### 4.2. Interface CBRCaseBase

Elle définit les méthodes que chaque base de cas doit implémenter pour une meilleure utilisation dans le cadre de ce Framework. La base de cas représente l'organisation des cas en mémoire. Elle contient les méthodes décrites ci-dessous :

- *init(Connector connector)* : cette méthode initialise la base de cas à travers la réception de connecteur « connector ». En cas d'échec cette fonction retourne une exception de type « InitializingException ».
- *Close()* : elle vide la base de cas en mémoire.
- *Collection<CBRCase> getCases :* retourne les cas disponibles dans la base de cas.
- *Collection<CBRCase> getCases(CaseBaseFilter filter) :* retourne les cas selon un critère.
- *learnCases(Collection<CBRCase> cases)* : ajoute une collection de nouveaux objets de type CBRCase à la base des cas.

- *forgetCases(Collection<CBRCase> cases) :* supprime des cas de la base de cas.

### 4.3. Interface CaseComponent

L'interface qui définit les composants ou bien les attributs d'un cas. Les cas seront identifiés en instanciant cette interface. Ces composants comme déjà noté sont des Java Beans. Elle retourne un attribut unique pour chaque cas.

### 4.4. Interface Connector

L'interface-Connector déclare les méthodes requises pour l'accès aux cas stockés dans une structure qui garantit leur persistance et qui dans ce cas est une base de données. jCOLIBRI gère les cas en base de données et l'organisation en mémoire définie par l'interface CBRCaseBase. Elle définit les fonctions suivantes :

- *initFromXMLfile(java.net.URL file)* : initialise les connecteurs en utilisant des documents XML. Ces documents XML représentent la configuration de package Hibernate.
- *Close()* : réinitialise les ressource utilisées par les connecteurs et éteindre le service.
- *storeCases(Collection<CBRCase> cases)* : stocke les classes de représentation des cas « les Java Beans » et retourne une liste de cas.
- *DeleteCases(Collection<CBRCase> cases)* : supprime la liste des cas de la mémoire.
- *Collection<CBRCase> retrieveAllCases()* : retourne tous les cas.
- *Collection<CBRCase> retrieveSomeCases(caseBaseFilter filter)* : retourne des cas selon un critère

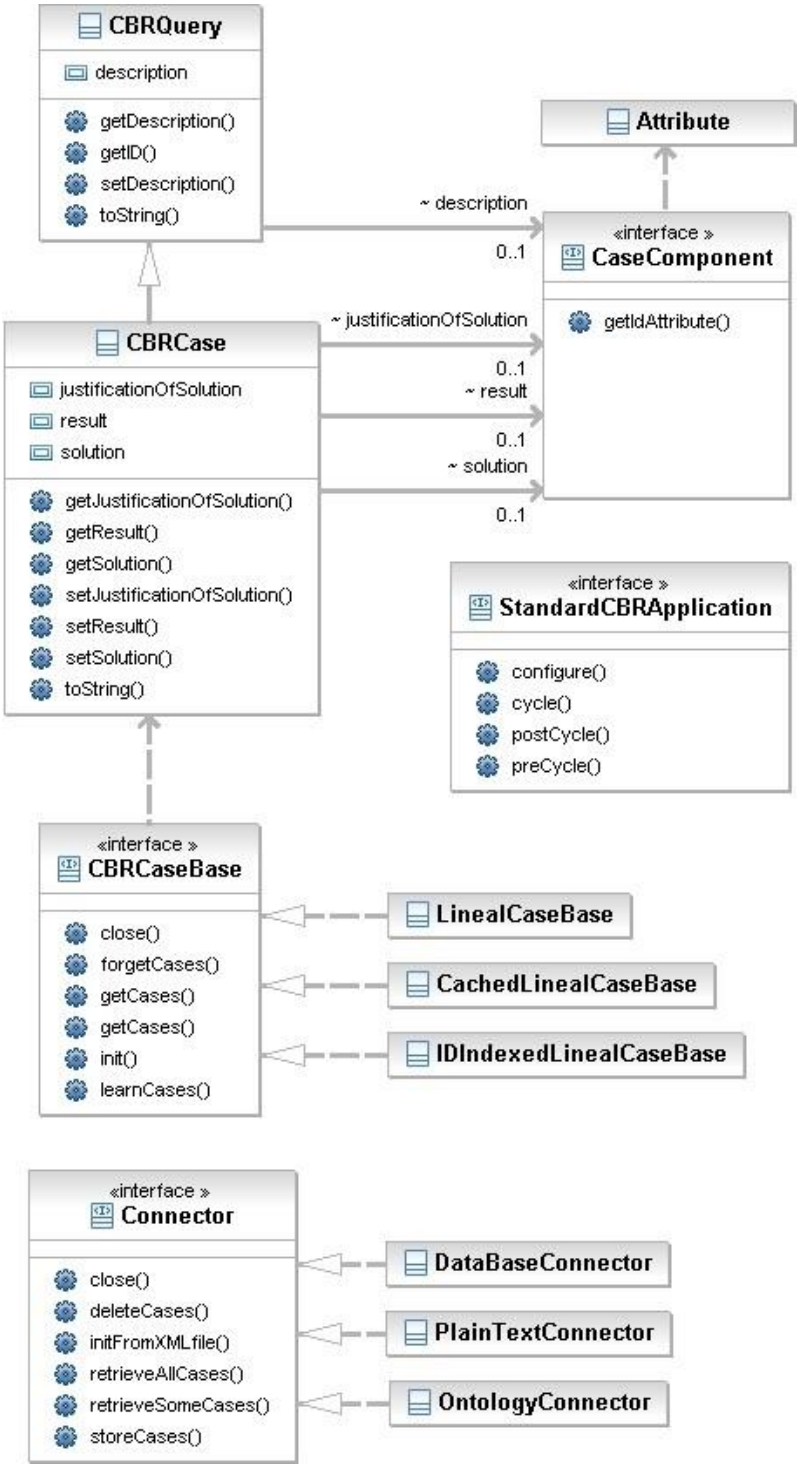


Figure 3.2 : diagramme UML avec les éléments principaux ode jCOLIBRI 2.1

## 5- L'organisation de la base de cas

La base de cas de l'application est stockée dans une base de données. jCOLIBRI gère n'importe quel Data Base Manager « DBM » grâce à l'utilisation du middleware Hibernate. Hibernate[1] est un outil puissant avec une performance très élevée pour la persistance objet/relationnelle et les services de requête. Il utilise les documents XML pour la configuration de ce dernier. A travers cet outil jCOLIBRI permet l'utilisation de n'importe quelle DBM comme Oracle, MySQL, etc.

## 6- Structure d'un cas [12]

Un cas dans JCOLIBRI est constitué de quatre éléments :

- La description du cas.
- La solution
- Le résultat d'application de la solution.
- Justification de la solution.

Cette structure est basée sur les principaux travaux du domaine de RàPC (Kolodner, 1993, Althoff et al. 1995). Elle est définie dans les classes CBRQuery, CBRCase et CaseComponent.

## 7- Les Processus du cycle du RàPC

Il faut noter que jCOLIBRI fournit juste les interfaces expliquées ci-dessus comme une hiérarchie de base des fonctions qui exécutent les quatre étapes du cycle. Ces fonctions doivent être implémentées par les développeurs Java car elles dépendent du domaine. jCOLIBRI donne quelques exemples sur les quatre processus détaillé ci-dessous :

### 7.1. La phase de remémoration

C'est phase la plus importante et la plus coûteuse dans la résolution d'un problème cible en utilisant le RàPC. Elle permet de récupérer, à partir de la base de cas, les cas similaires au nouveau cas. La solution du cas sélectionné (parmi les cas remémorés) sera utilisée pour résoudre le nouveau problème. Certains éléments critiques peuvent influencer le fonctionnement de cette phase [2] :

- Une base de cas volumineuse, dans ce cas le temps de récupération des cas similaires peut devenir très lent et inacceptable.
- Des cas contenant un grand nombre d'attributs. Lorsque les attributs sont nombreux, la performance de la remémoration devient mauvaise vu le besoin de calcul de distance entre les attributs du nouveau cas et les cas dans la base de cas.

[1] www.hibernate.org

### 7.1.1. Les fonctions de similarité globales

La fonction de remémoration la plus utilisée est le plus proche voisin [12]. Elle est utilisée comme une fonction principale de similarité globale entre les deux cas c.-à-d. la requête introduite au système et le cas existant [13]. Comme elle très populaire et on peut trouver dans nombreux systèmes à RàPC. De plus, dans jCOLIBRI on trouve deux autres fonctions de remémoration comme « ExpertClerkMedianScoring » et « FilteringRetrieval ».

### 7.1.2. Les fonctions de similarité locales

jCOLIBRI met à la disposition des développeurs des fonctions de similarité locales comme des fonctions de base pour les utiliser pour calculer la similarité entre les attributs de deux cas. De plus, chaque fonction est utilisée pour un type de donnée diffèrent (nombre, chaine de caractère, texte etc.). Ces fonctions sont localisées dans le package « similarity » et elles sont décrites comme suit :

- Fonctions de similarité pour les ontologies : ces fonctions sont utilisées pour calculer la similarité en cas d'ontologie. Donc si on veut développer un système en utilisant l'ontologie, on utilise une des fonctions suivantes : OntCosine, OntDeep, OntDeepBasic, OntDetail.
- Fonctions de similarité pour les textes : ces fonctions sont dédiées au type texte ; on peut citer : CosineCoefficient, DiceCoefficient, JaccardCoefficient, LuceneTextSimilarity, OverlapCoefficient.
- Fonctions de similarité entre les nombres : Elles sont appliquées pour calculer la distance entre les nombre comme : Threshold, Interval,
- Fonctions de similarité entre les chaines de caractères comme : Table, MaxString, EqualsStringIgnoreCase.
- Equal : C'est une fonction utilisable pour tous typee de données puisqu'elle est simple, elle vérifie si les deux attributs sont égaux ou pas.

### 7.1.3. La sélection des cas similaires

Après la saisie d'une requête ou bien un nouveau cas, la phase de remémoration commence en calculant la distance (similarité) entre les attributs des deux cas (stocké et nouveau) en utilisant l'une de fonctions décrites ci-dessus comme une fonction de similarité locale pour chaque attribut du cas.

Les développeurs peuvent définir des poids pour chaque attribut selon son importance pour la sélection des cas similaires, par exemple, l'algorithme K-PPV est calculé comme une moyenne simple s'il n'y a pas de poids pour les attributs ou bien pondéré sinon selon la fonction « Average () » qui donne une valeur dans l'intervalle [0...1]. Les cas similaires sont retournés en fonction de leurs similarités. Lorsque les cas sont renvoyés, ils vont être sélectionnés. La

sélection des cas est définie dans la classs « SelectCases ». L'expert peut choisir une fonction définie dans cette classe selon ses besoins :

- *Collection<CBRCase> selectAll(Collection<RetrievalResult> cases)* : elle sélectionne tous les cas dans une liste ordonnée du plus proche cas au dernier cas.
- *Collection<CBRCase> selectTopK (Collection<RetrievalResult> cases, int k)* : sélectionne des cas selon un k donné par l'expert, k représente le nombre des cas à retourner.

### 7.1.4. Définir une nouvelle fonction de similarité globale

Le développeur peut définir ses propres fonctions de calcul de similarité en implémentant l'interface « GlobalSimilarityFunction ». Cette interface définit les fonctions de similarité globales qui sont appliquées pour calculer la similarité entre le nouveau cas et le cas existant. Lorsqu'elle est implémentée, elle génère la fonction *compute* (CaseComponent *componentOfCase*, CaseComponent *componentOfQuery*, CBRCase *_case*, CBRQuery *_query*, NNConfig *numSimConfig*) où :

- *componentOfCase :* le composant du cas existant.
- *componentOfQuery :* le composant du nouveau cas ou la requête.
- *_case :* le cas qui doit être comparé avec la requête.
- *_query :* la requête introduite.
- *numSimConfig :* la configuration de la fonction de similarité.

Cette fonction renvoie une valeur entre 0 et 1. Lorsque cette valeur est plus proche de 0 cela veut dire que ce cas est similaire au nouveau cas.

### 7.1.5. Définir une nouvelle fonction de similarité locale

Pour définir une fonction de similarité locale entre les attributs des cas, le développeur doit implémenter l'interface « LocalSimilarityFunction ». Cette interface définit la structure de fonction de similarité. Elle génère la fonction *compute*(Object *caseObject*, Object *queryObject*). Cette fonction calcule la distance entre deux objets Java où :

- *caseObject* : l'objet de cas existant.
- *queryObject* : l'objet de nouvelle requête.

Cette fonction renvoie une valeur de similarité entre 0 et 1. En plus, cette interface génère une autre fonction « *isApplicable*(Object caseObject, Object queryObject) » qui indique si cette fonction « compute » est applicable pour les deux objets sinon elle renvoie une exception de type « NoApplicableSimilarityFunctionException » qui veut dire que cette fonction n'est pas applicable pour ce type de donnée.

### 7.2. Exemple de remémoration

Dans le Framework jCOLIBRI on trouve un exemple explique les différentes étapes du cycle du RàPC qui est « Travel Recommender ». Comme son nom l'indique, il s'agit d'une application qui permet d'aider un utilisateur qui veut planifier un voyage (des vacances). Dans cet exemple, un cas est formulé avec 9 attributs (Id, HolidayType , Price, NumberOfPersons, Region , Transportation, Duration, Season, Accommodation, Hotel). Ou Id est l'identifiant unique de chaque cas, la description de cas est expliquée via les 7 attributs (HolidayType, NumberOfPersons, Region, Transportation, Duration, Season, Accommodation) et les deux attributs restants (Price, Hotel) représentent la solution. Ces cas sont stockés dans une base de données relationnelle.

L'utilisateur introduit son nouveau cas (requête) dans la fenêtre de requête où il doit saisir les valeurs des attributs de description. Ensuite, l'utilisateur configure ses propres fonctions de similarité pour la phase de remémoration des cas similaire dans la fenêtre de configuration de similarité (Figure 3.3).

L'utilisateur peut également préciser l'entier k qui définit le nombre de cas à sélectionner dans la phase de remémoration. Cet exemple utilise la fonction plus proche voisin (k-ppv) comme une fonction de similarité globale. Cette fonction renvoie les **k** cas proches du nouveau cas.

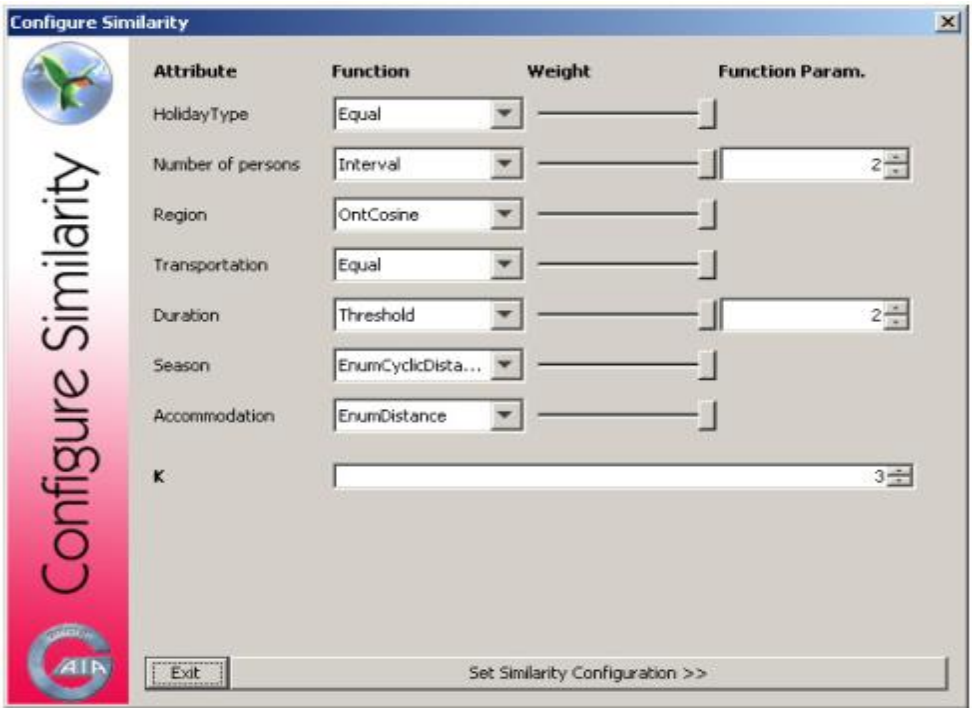


Figure 3.3 : Fenêtre de configuration des mesures de similarité

### 7.3. La phase réutiliser [13]

Dans phase réutiliser ou bien la phase d'adaptation, il s'agit d'adapter la solution des cas récupérés selon les exigences du nouveau cas. Elle dépend du domaine de l'application. jCOLIBRI ne définit pas des fonctions d'adaptation particulières mais il est ouvert aux développeurs qui doivent définir leurs propres fonctions d'adaptation. Lorsque les solutions des cas remémorés sont adaptées, le système à RàPC les propose comme des solutions pour le nouveau problème, ensuite l'utilisateur du système ou l'expert du domaine révise ces solutions.

### 7.4 La phase réviser

Dans cette étape, la solution proposée par le système est testée et évaluée selon son application par l'expert (qui peut être un simple utilisateur). La solution proposée est réparée en cas d'échec. Cette phase dépend toujours du domaine et elle change selon l'application.

### 7.5 la phase retenir

L'étape retenir enregistre les cas jugés utiles par l'expert dans la base de cas pour des utilisations futures. Cette phase représente l'apprentissage automatique dans les systèmes basés sur le RàPC. jCOLIBRI inclut la classe « StoreCaseMethod » pour enregistrer des nouveaux cas dans la base de cas. Les cas ajoutés vont être stockés dans la base de données.

## 8- Configuration de Hibernate [21]

Hibernate est un outil de mapping objet/relationnel pour le monde Java. Hibernate permet aux développeurs d'écrire plus facilement des applications dont les données survivent au processus de demande. En tant que cadre « Object/Relational Mapping » (ORM), Hibernate s'intéresse à la persistance des données, tel qu'elle s'applique aux bases de données relationnelles. Il permet également de faire le lien en Java entre la représentation objet des données (Java Beans) et la représentation relationnelle basé sur un schéma SQL à travers des fichiers de configuration.

Une instance de « *net.sf.hibernate.cfg.Configuration* » représente un ensemble de mapping des classes Java d'une application vers la base de données SQL. La Configuration est utilisée pour construire un objet (immuable) « SessionFactory ». Les mappings sont constitués d'un ensemble de fichiers de mapping XML.

Il est possible d'obtenir une instance de « Configuration » en l'instanciant directement. Voici un exemple de configuration d'une source de données et d'un mapping composé de deux fichiers de configuration XML (qui se trouvent dans le classpath) :

```
Confuguration cfg = new Confuguration()
    .addFile("fichier1.hbm.xml ")
    .addFile("fichier2.hbm.xml ") ;
```

### 8.1. Obtenir une « SessionFactory »

Quand tous les mappings ont été parsés par la Configuration, l'application doit obtenir une fabrique d'instances de Session. Cette fabrique est supposée être partagée par tous les threads de l'application :

SessionFactory sessions = cfg.buildSessionFactory() ;

Cependant, Hibernate permet à l'application enquestion (l'application qui utilise Hibernate) d'instancier plus d'une SessionFactory. C'est utile si plusieurs bases de données sont utilisées.

### 8.2. Connexions JDBC fournie par Hibernate

Alternativement, il est possible de laisser la SessionFactory se charger de l'ouverture des connexions. La SessionFactory doit recevoir les propriétés de connexions JDBC de l'une des manières suivantes :

- Passer une instance de « java.util.Properties » àn « Configuration.setProperties() ».
- Placer « Hibernate.properties » dans un répertoire racine du « classpath ».
- Positionner les propriétés « System » en utilisant « java -Dproperty=value ».

Tous les noms et sémantiques des propriétés d'Hibernate sont définies dans le « javadoc » de la classe « net.sf.hibernate.cfg.Environment ». Hibernate obtiendra des connexions (et les mettra dans un pool) en utilisant « java.sql.DriverManager » si les paramètres sont positionnés de la manière suivante :

| **Nom de la propriété** | **Fonction** |
|---|---|
| hibernate.connection.driver_class | *Classe du driver jdbc* |
| hibernate.connection.url | *URL jdbc* |
| hibernate.connection.username | *utilisateur de la base de données* |
| hibernate.connection.password | *mot de passe de la base de données* |
| hibernate.connection.pool_size | *nombre maximum de connexions dans le pool* |

Tableau 3.1 : Propriétés JDBC d'Hibernate

### 8.3. Fichier de configuration XML de Hibernate

Une approche alternative est de spécifier toute la configuration dans un fichier nommé « hibernate.cfg.xml ». Ce fichier peut être utilisé à la place du fichier « hibernate.properties », voire même peut servir à surcharger les propriétés si les deux fichiers sont présents. Le fichier de configuration XML doit par défaut être placé à la racine du CLASSPATH. En voici un exemple :

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration
DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>

        <!-- Database connection settings -->

        <property name="connection.driver_class">org.hsqldb.jdbcDriver</property>

        <property
name="connection.url">jdbc:hsqldb:hsql://localhost/hopital</property>

        <property name="connection.username">sa</property>

             <property name="connection.password"></property>

        <!-- JDBC connection pool (use the built-in) -->

        <property name="connection.pool_size">2</property>

        <!-- SQL dialect -->

        <property name="dialect">org.hibernate.dialect.HSQLDialect</property>

        <!-- Enable Hibernate's automatic session context management -->

        <property name="current_session_context_class">thread</property>

        <!-- Disable the second-level cache  -->

        <property
name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>

        <!-- Echo all executed SQL to stdout -->

        <property name="show_sql">true</property>

    </session-factory>

</hibernate-configuration>
```

### 8.4. Fichier de mapping

Un fichier de mapping (voir l'exemple au-dessous) associe des propriétés Java de la classe persistante (un java Bean) à des propriétés d'un SGBD relationnel. Un java Bean représente un évènement qui a un identifiant unique où :

- Elément <class> : permet d'associer une classe Java à une table de la base de données.
- Attributs de l'élément <class> : « name » est le nom de la classe Java, « table » est le nom de la table correspondante
- Elément <id> : Identifiant unique pour les évènements.
- Attributs de l'élément <id> : « name » est nom de l'attribut dans la classe Java, « column » est le nom de la propriété dans la table SQL
- Elément <property> : déclare une propriété de la classe au sens JavaBean.
- Attributs de l'élément <property> : « name » est le nom de la propriété, « column » est le nom de la colonne mappée.

```
<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping
DTD//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping default-lazy="false">

<class name="jcolibri.test.test1.TraumaDescription" table="maladie">

        <id name="Id" column="Id"><generator class="native"/></id>

                <property name="nom" column="nom"/>

                <property name="Age" column="Age"/>

</class>

</hibernate-mapping>
```

## 9- Conclusion

L'utilisation d'une plateforme telle que jCOLIBRI destinée aux développeurs nécessite une connaissance approfondie de sa structure et des principales fonctionnalités. En effet, sans cette connaissance il est impossible d'ajouter de nouvelles fonctionnalités ou de modifier des fonctionnalités existantes. Malgré la documentation fournie avec la plateforme jCOLIBRI, cette exploration a été très fastidieuse. Beaucoup d'aspects nécessaires à l'implémentation n'ont pu être compris qu'à travers un examen minutieux du code Java fourni dans le Framework.

Après cette exploration, il devient maintenant possible d'implémenter une nouvelle application en utilisant la plateforme jCOLIBRI. C'est l'objectif du chapitre suivant.

# Chapitre 04

# Application du RàPC pour un Système d'Aide à la Décision Médicale

## 1-Introduction

Après la compréhension de la méthodologie du RàPC et l'étude et l'analyse du Framework de développement jCOLIBRI, notre objectif est d'appliquer la méthodologie du RàPC pour la conception d'un système d'aide à la décision pour la prise en charge d'un enfant victime d'un accident de la route. Ce projet est fait en collaboration avec de l'établissement hospitalier spécialisé de Canastel-Oran- (EHS). Ce chapitre est consacré à la modélisation d'une solution de prise en charge d'une enfant victime d'un accident de la route. Cette solution est bien évidemment basée sur le RàPC et utilisera la plateforme jCOLIBRI. Une attention particulière sera accordée à la phase de remémoration car elle constitue le goulot d'étranglement de tout système basé sur le RàPC.

## 2- Cas d'étude : L'enfant victime d'un accident de la route

Les accidents de circulation chez les enfants constituent un problème de santé publique majeur en Algérie et ont des conséquences sanitaires très importantes. De plus, l'impact est direct physique et psychologique sur ceux qui y sont directement touchés, s'ils ne sont pas morts. La prise en charge aux urgences doit pouvoir répondre à plusieurs besoins de la victime : traiter de façon approprié et rapide les traumatismes afin d'augmenter la probabilité de survie et/ou de minimiser les troubles fonctionnels, mais également tenter de réduire au maximum la douleur et les souffrances physiques et/ou psychologiques. Il est bien évident que la rapidité de la prise en charge des victimes des accidents de la route est très importante. Les enfants victimes d'accidents de la route dans l'ouest Algérien sont souvent évacuées vers l'urgence pédiatrique d'EHS de Canastel soit directement à partir du site de l'accident directement ou bien à travers un service d'urgence d'un autre hôpital.

### 2.1. La procédure actuelle de prise en charge de l'enfant victime d'un accident

#### 2.1.1. SAMU (Service d'aide médicale d'urgence)

Comme il est internationalement connu « il est interdit de toucher un accidenté quel que soit les circonstances » car les actions faites par les non-expert peuvent provoquer des lésions en plus de ceux provoqué par l'accident. C'est la raison derrière la création de groupes d'intervention mobiles pour la prise en charge victimes d'un accident de la circulation. La prise en charge commence déjà sur le lieu même de l'accident avec un transport sous assistance et surveillance médicale.

Une fois sur place, l'équipe soignante (médecin, infirmier et brancardier) délivre les soins d'urgence vitale, évalue la gravité des lésions et organise le transport médicalisé du blessé vers la structure hospitalière la plus proche ou la plus apte à gérer les soins d'un blessé par accident

de la route. Elle doit donc évaluer la gravité lésionnelle, les circonstances accidentelles, mesurer les différents paramètres physiologiques appréciant le retentissement des lésions traumatiques (Score de Glasgow, pression artérielle, fréquence cardiaque, fréquence respiratoire, température corporelle) et prendre les décisions urgentes qui s'imposent.

Par exemple, l'équipe médicale d'urgence doit être capable d'intuber et de ventiler le patient victime d'un traumatisme crânien grave ayant des troubles de conscience, d'exsuffler un pneumothorax suffocant, de restaurer l'hémodynamique d'un patient en défaillance cardio-vasculaire (remplissage, éventuelle transfusion d'urgence, arrêt des hémorragies extériorisées). Ensuite, elle conditionnera le blessé de manière à ce que le transport puisse se faire sous surveillance continue et en toute sécurité (matelas coquille, collier cervical, scope, surveillance de la pression artérielle et du $CO_2$ expiré si le patient est intubé). La rapidité de la prise en charge immédiate est fondamentale et en aucun cas le transport vers le centre hospitalier ne doit être retardé.

### 2.1.2. Service d'urgence

Dans le langage courant, les **urgences** sont le service d'un hôpital qui s'occupe de recevoir les malades et les blessés qui se présentent d'eux-mêmes, ou qui sont amenés par les services de secours (SAMU, pompiers, etc.). Dans ce service, selon le docteur responsable de l'EHS, quand les urgentistes (les médecins de l'urgence) reçoivent une victime, si elle est amenée par le SAMU, ils consultent la fiche de ces derniers pour évaluer l'état de l'enfant, surtout le score de Glasgow (détaillé par la suite) Ils vont également prendre d'autres mesures importantes pour décider la suite de la prise en charge. Si l'enfant n'a pas été amené par un SAMU, les médecins vont prendre les mesures eux-mêmes. Quand les médecins décident que l'état est grave la victime est renvoyée au service de réanimation.

### 2.1.3. Service de réanimation

Le service de réanimation est le lieu où l'on prend en charge les malades les plus gravement atteints, ceux dont le pronostic vital est en jeu parce qu'il y a défaillance d'un ou plusieurs organes. Dès lors, se met en marche une machine très lourde, de haute technicité. Il s'agit tout à la fois de faire le diagnostic de la ou des maladies en cause, de maintenir les fonctions vitales, d'assurer le confort du malade, d'éviter les complications (infections nosocomiales…).

### 2.1.4. Médecin

C'est l'acteur le plus important dans la prise en charge de l'enfant victime d'un accident. Le médecin représente le décideur, sa décision représente la thérapie adéquate.

La **pédiatrie** est une branche spécialisée de la médecine qui étudie le développement psychomoteur et physiologique normal de l'enfant, ainsi que toute la pathologie qui y a trait (maladies infantiles), de la naissance à la période post pubertaire où il devient adulte ; c'est la médecine des enfants. L'enfant étant défini en droit comme tout sujet âgé de moins de 16 ans. Le médecin spécialisé en pédiatrie s'appelle le **pédiatre**.

# 3- Les mesures importantes dans la prise en charge

## 3.1. Le Score de Glasgow

Le Score de Glasgow, décrit par Jennett et Teasdale en 1974, est défini par l'échelle de Glasgow (Glasgow Coma Scale, GCS). Il permet l'évaluation de l'état de conscience à un instant donné et de suivre l'évolution. Ce test est reproductible d'un examinateur à l'autre. Il peut être utilisé même par des examinateurs non médecins quelle que soit leurs expériences, tels que les secouristes ou les infirmiers. Ce score est mesuré toutes les 45 minutes pour suivre l'état de l'enfant victime d'un accident [17]. Ce Score représente un entier entre 3 et 15. D'après les discussions avec les responsables du service de réanimation de l'EHS, dès que le score est inférieur à 10 on considère que l'enfant est en état grave et doit être intubé.

| **Echelle de Glasgow** | | |
|---|---|---|
| Ouverture des yeux | Réaction | Score |
| | Spontanée | 4 |
| | À la demande | 3 |
| | À la douleur | 2 |
| | Aucune | 1 |
| | | |
| Réponse Verbale | Orienté | 5 |
| | Confuse | 4 |
| | Inappropriée | 3 |
| | Incompréhensible | 2 |
| | Aucune | 1 |
| | | |
| Réponse motrice | Obéit aux ordres | 6 |
| | Localise la douleur | 5 |
| | Inadaptée | 4 |
| | Flexion à la douleur | 3 |
| | Extension à la douleur | 2 |
| | Aucune | 1 |
| | | Total ../15 |

Tableau 4.1 : Score de Glasgow

### 3.2. Hémodynamique

La prise en charge médicale initiale dans ces circonstances doit avoir comme objectifs le maintien d'une bonne hémodynamique [3]. Il y'a plusieurs paramètres qui permettent au médecin de décider si l'enfant est dans l'état normal, hypertension (HTA) ou en état de choc. Ces facteurs sont :

#### 3.2.1. Pression artérielle systolique (PAS) [3]

La pression artérielle, ou pression artérielle systolique, correspond à la pression du sang dans les artères de la circulation systémique (circulation principale). On parle aussi de tension artérielle (ou simplement de tension en raccourci) car cette pression est aussi la force exercée par le sang sur la paroi des artères, ce qui les tend : *Stricto sensu*, la *tension* dans la paroi de l'artère résulte directement de la pression.

| ALERTE | AGE/Mois | SEXE | PAS |
|---|---|---|---|
| Etat normal | < 7 j | Masculin | 63-83 |
| Etat normal | 7-30 j | Masculin | 71-93 |
| Etat normal | 1-6 mois | Masculin | 80-106 |
| Etat normal | [7-12[ | Masculin | 80-110 |
| Etat normal | [12-24[ | Masculin | 82-106 |
| Etat normal | [24-36[ | Masculin | 84-106 |
| Etat normal | [36-48[ | Masculin | 81-107 |
| Etat normal | [48-60[ | Masculin | 79-103 |
| Etat normal | [84-96[ | Masculin | 88-108 |
| Etat normal | [120-132[ | Masculin | 92-112 |
| Etat normal | [180-192[ | Masculin | 101-125 |
| Etat normal | < 7 j | Féminin | 63-81 |
| Etat normal | 7-30 j | Féminin | 70-94 |
| Etat normal | 1-6 mois | Féminin | 80-104 |
| Etat normal | [7-12[ | Féminin | 80-110 |
| Etat normal | [12-24[ | Féminin | 80-106 |
| Etat normal | [24-36[ | Féminin | 84-106 |
| Etat normal | [36-48[ | Féminin | 80-106 |
| Etat normal | [48-60[ | Féminin | 78-104 |
| Etat normal | [84-96[ | Féminin | 86-106 |
| Etat normal | [120-132[ | Féminin | 91-113 |
| Etat normal | [180-192[ | Féminin | 97-119 |

Tableau 4.2 : Pression Artérielle dans l'état normal

### 3.2.2. Fréquence cardiaque (FC)

La fréquence cardiaque ou le rythme cardiaque peut être mesurée par le pouls. Le rythme cardiaque est le nombre de pulsations (battements) cardiaques par minute. Il doit, théoriquement, être de 70 pulsations/min chez un homme au repos et être plus élevé lors de la pratique d'un sport ou pendant la digestion. S'il apparaît trop élevé, on parle de tachycardie. S'il apparaît trop bas on parle de bradycardie. Le rythme cardiaque est un outil de diagnostic indispensable pour les troubles cardiaques, notamment en cas d'arythmie.

| ALERTE | AGE/ Mois | SEXE | FC |
|---|---|---|---|
| Etat normal | Naissance | Masculin | 115-165 |
| Etat normal | 1-3 J | Masculin | 105-127 |
| Etat normal | 4-7 J | Masculin | 108-134 |
| Etat normal | 8-14 J | Masculin | 130-152 |
| Etat normal | < 1 mois | Masculin | 108-152 |
| Etat normal | [1-12[ | Masculin | 117-153 |
| Etat normal | [12-24[ | Masculin | 89-121 |
| Etat normal | [24-36[ | Masculin | 81-105 |
| Etat normal | [36-48[ | Masculin | 78-96 |
| Etat normal | [48-60[ | Masculin | 76-92 |
| Etat normal | [84-96[ | Masculin | 67-83 |
| Etat normal | [120-132[ | Masculin | 60-74 |
| Etat normal | [180-192[ | Masculin | 53-69 |
| Etat normal | Naissance | Féminin | 115-165 |
| Etat normal | 1-3 J | Féminin | 105-127 |
| Etat normal | 4-7 J | Féminin | 108-134 |
| Etat normal | 8-14 J | Féminin | 130-152 |
| Etat normal | < 1 mois | Féminin | 108-052 |
| Etat normal | [1-12[ | Féminin | 105-147 |
| Etat normal | [12-24[ | Féminin | 87-121 |
| Etat normal | [24-36[ | Féminin | 84-102 |
| Etat normal | [36-48[ | Féminin | 80-98 |
| Etat normal | [48-60[ | Féminin | 76-91 |
| Etat normal | [84-96[ | Féminin | 68-84 |
| Etat normal | [120-132[ | Féminin | 61-77 |
| Etat normal | [180-192[ | Féminin | 57-73 |

Tableau 4.3 : Fréquence Cardiaque dans l'état normal

### 3.3. Respiration

#### 3.3.1. Les voies aériennes supérieures

Représente l'état respiratoire au moment d'arrivé de l'enfant à la salle d'urgence soit il est en liberté normale ou bien arrivé intubé.

#### 3.3.2. Fréquence respiratoire (FR)

La fréquence respiratoire est la quantité de cycles respiratoires se déroulant chez un individu en une minute. Un cycle respiratoire comprend une inspiration (air entrant) et une expiration (air sortant). La fréquence respiratoire se réduit naturellement avec l'âge. Celle des nouveau-nés (en moyenne 50 cycles par minute) est ainsi beaucoup plus rapide que celle d'un adulte (en moyenne 16 cycles par minute). De nombreux facteurs peuvent aussi influer sur la fréquence respiratoire : pratique d'une activité physique, prise de certains médicaments, maladie touchant le système respiratoire, etc.

| ALERTE | AGE/ Mois | SEXE | FR |
|---|---|---|---|
| Etat normal | Prématuré | Masculin | 40-90 |
| Etat normal | Nouveau-né | Masculin | 30-80 |
| Etat normal | [1-7[ | Masculin | 30-54 |
| Etat normal | [7-12[ | Masculin | 23-39 |
| Etat normal | [12-24[ | Masculin | 22-30 |
| Etat normal | [24-36[ | Masculin | 21-29 |
| Etat normal | [36-48[ | Masculin | 21-27 |
| Etat normal | [48-60[ | Masculin | 21-25 |
| Etat normal | [84-96[ | Masculin | 17-23 |
| Etat normal | [120-132[ | Masculin | 17-21 |
| Etat normal | [180-192[ | Masculin | 14-20 |
| Etat normal | Prématuré | Féminin | 40-90 |
| Etat normal | Nouveau-né | Féminin | 30-80 |
| Etat normal | [1-7[ | Féminin | 30-54 |
| Etat normal | [7-12[ | Féminin | 24-36 |
| Etat normal | [12-24[ | Féminin | 23-31 |
| Etat normal | [24-36[ | Féminin | 22-28 |
| Etat normal | [36-48[ | Féminin | 21-27 |
| Etat normal | [48-60[ | Féminin | 20-24 |
| Etat normal | [84-96[ | Féminin | 18-22 |
| Etat normal | [120-132[ | Féminin | 16-22 |
| Etat normal | [180-192[ | Féminin | 15-21 |

Tableau 4.4 : Fréquence Respiratoire dans l'état normal

### 3.4. Bilan des lésions

Représente le siège des lésions dans le corps d'enfant : abdomen, bassin, reins et voies urinaires et l'existence d'hématomes ou des plaies.

## 4- Modélisation de la solution

Après la collecte des informations nécessaires auprès du service de réanimation de l'EHS de Canastel et les discussions avec les médecins, il devient maintenant possible de passer à une modélisation d'une solution dans le cadre de système d'aide de décision pour la prise en charge d'un enfant victime d'un accident de la route. Cette solution sera bien entendu basée sur l'utilisation du RàPC.

### 4.1. La représentation du cas

Puisqu'il n'y a pas une représentation prédéfinie du cas pour les systèmes basés sur le RàPC, nous avons opté, dans cette première solution, pour une représentation simple sous forme de vecteur. Compte tenu des informations recueillies et des dossiers mis à notre disposition, le cas est composé de 65 attributs répartis en quatre catégories : problème, solution, résultat et pertinence.

#### 4.1.1. Partie problème

- **Informations sur l'enfant :** représentent les informations d'enfant qui sont Nom, Prénom, sexe, Age, Poids, Numéro du téléphone, Wilaya de l'enfant.
- **Informations sur l'accident :** les informations tirées de l'accident : lieu de l'accident, heure de l'accident, position de l'enfant dans la voiture (avant/arrière) s'il était dedans, port de la ceinture de sécurité, type de véhicule, la vitesse du véhicule, durée écoulée entre le lieu de l'accident et l'arrivée au service d'urgence, le mode de ramassage de l'enfant et le Score de Glasgow au lieu de l'accident et à la salle d'urgence.
- **Informations concernant Respiration** : Fréquence respiratoire (FR), Voies aérienne supérieure.
- **Informations concernant L'Hémodynamique** : PAS, PAD, FC.
- **Informations concernant les lésions :** précisions données par le médecin après un premier examen de l'enfant dont le but est de localiser les lésions et plaies : abdomen, membres et bassin, reins et les voies urinaires et l'examen de la face et l'emplacement de l'hématome et de la plaie.

### 4.1.2. Partie solution

Dans la partie solution, le médecin du service nous a conseillé d'utiliser les attributs : immobilisation du rachis, $O_2$ thérapie, intubation de l'enfant, perfusion, transfusion, immobilisation, et l'indication opératoire qui représente la décision du médecin si cet enfant doit être conduit vers le bloc opératoire ou pas.

## 4.2. Construction de la base de cas

La base de cas représente le cœur des systèmes basés sur le RàPC. Dans le cadre de ce projet, la base de cas a été construite en collaboration avec l'établissement hospitalier de Canastel –Oran-. Il faut savoir que le service en question mémorise les données qui concernent tous les patients (enfants) passés par ce service. Dans ce projet, nous nous sommes intéressés uniquement des dossiers d'enfants victimes d'un accident de la route. Et comme il s'agit d'une première étude dans ce domaine, nous avons sélectionné les dossiers des patients (enfants) qui présentent un traumatisme crânien.

Dans le service concerné, les données sont mémorisées sous format papier et reportées par ordre chronologique dans un registre d'archive. Les données à caractère médical sont protégées par le secret médical et tout accès doit être règlementé. Nous avions donc eu beaucoup de difficultés à recueillir des données de départ afin de construire une première base de cas qui permet de faire les premiers essais.

Les données collectées ont été reportées dans un tableau afin de constituer un fichier des données sous format CSV (Comma Separated Values). Puisque jCOLIBRI utilise les fichiers « .txt » pour générer la base du cas du système, il a fallu ajouter une fonctionnalité (à jCOLIBRI) qui permet de filtrer les données à partir d'un fichier sous format « .csv ».

## 4.3. Mesures de similarités Locales

Puisque la phase de remémoration est la phase la plus importante dans un système basé sur le RàPC, un premier objectif était de trouver des mesures de similarité qui donnent des résultats efficaces car le Framework jCOLIBRI fournit uniquement des fonctions de mesures de similarité simples. Pour notre système nous avons choisi quatre fonctions de similarité locale c.-à-d. entre les attributs des cas, puis nous avons implémenté ces fonctions dans la plateforme. Ces fonctions sont :

- **La distance Jaro-Winkler** : fonction de similarité basée sur l'analyse des caractères et qui a été développée au bureau du recensement des États-Unis. Jaro a introduit une mesure de similarité qui porte désormais son nom et qui se définit comme suit [16] :

$$Jaro(c,d) = pc\,\frac{com}{m} + pd\,\frac{com}{n} + ptr\,\frac{com - tr}{com}$$

*pc :* est le poids associé aux caractères présents dans *c* mais pas dans *d* ;

*pd :* est le poids associé aux caractères présents dans *d* mais pas dans *c* ;

*ptr :* est le poids associé aux caractères transposés ;

*tr :* est le nombre de transpositions de caractères ;

*com :* est le nombre de caractères communs à *c* et à *d*.

Deux caractères sont considérés comme étant en commun s'ils sont identiques et si leur position dans leurs chaînes respectives différent au maximum de $\max(m,n)/2-1$ positions. Le nombre de transpositions est égal à la moitié (arrondie à l'entier inférieur) du nombre de caractères communs qui ne sont pas dans le même ordre dans les deux chaînes.

- **La Distance MetricLCS [15]** : cette distance définit la similarité entre deux séquences comme étant le nombre maximum des caractères d'une séquence qui peuvent être *pairés* (*matched* en anglais) avec celles de l'autre séquence. Cela revient à calculer la longueur de la plus longue sous-séquence commune (PLSC). . On utilise pour ce cas particulier un système de pointage qui n'attribue aucun point pour les caractères *pairés* et où on pénalise chacune des opérations d'édition nécessaires pour passer d'une chaîne de caractères à une autre.

- **La Distance Euclidienne** : c'est la distance euclidienne classique entre deux points. Afin de fournir un résultat compris entre 0 et 1, nous avons procédé à la normalisation de cette fonction selon la formule :

$$D(x,y)=\frac{\sqrt{x^2-y^2}}{x+y}$$

- **Equal** : une fonction simple pour les attributs qui ont des valeurs fixes, si deux attributs sont égaux elle retourne 0, sinon elle retourne 1.

Les mesures des similarités décrites ci-dessus sont utilisées pour calculer la similarité entre les attributs du cas. Le tableau qui suit associe pour chaque attribut sa propre mesure de similarité

| Attributs | Mesure de similarité |
|---|---|
| Nom | Jaro Winkler |
| Age | Euclidienne |
| Poids | |
| Sexe | Equal |
| Emplacement d'enfant dans la voiture | |
| Vehicule | |
| Dure | Euclidienne |
| Ramasseur | Equal |
| Score de Glascow au lieu d'accident | Euclidienne |
| Score de Glascow à la salle d'urgence | |
| Voie aérienne supérieure | Equal |
| Pression artérielle | Euclidienne |
| Fréquence cardiaque | |
| Fréquence respiratoire | |
| Lésion Abdomen | MetricLCS |
| Reines et voie urinaires | |
| Examen de la face | |
| Hématome | |
| Plaie | |

Tableau 4.5 : Association des mesures de similarité aux attributs.

### 4.4. La Mesure de similarité globale

Une fois les mesures de similarité entre attributs calculées, la similarité globale calcule la distance entre deux cas. Elle utilise une fonction de calcul de similarité globale. Dans le cadre de ce projet, nous utilisons une mesure de similarité globale basée sur la technique du plus proche voisin puisqu'elle est la plus utilisée dans les systèmes basés sur le RàPC [10]. Dans cet algorithme, la similarité entre le nouveau cas et le cas antérieur est calculée à travers la sommation de similarités entre les attributs (deux à deux) auxquels sont associés des poids. Cette similarité est normalisée pour donner une valeur entre 0 et 1 : où 0 représente la similarité totale. Pour notre système, l'algorithme est défini selon l'équation :

$$Sim(C,S) = \sum_{f=1}^{n} w_f * sim(C_f,S_f)/sum(w_f) \text{ [9]}$$

**C** est le nouveau cas, **S** est le cas mémorisé, **w** est le poids défini par un expert, **n** est le nombre d'attributs pour chaque cas, **f** est l'index de l'attribut et **sim($C_f$,$S_f$)** est la similarité locale pour l'attribut **f**.

## 5- Approche proposée

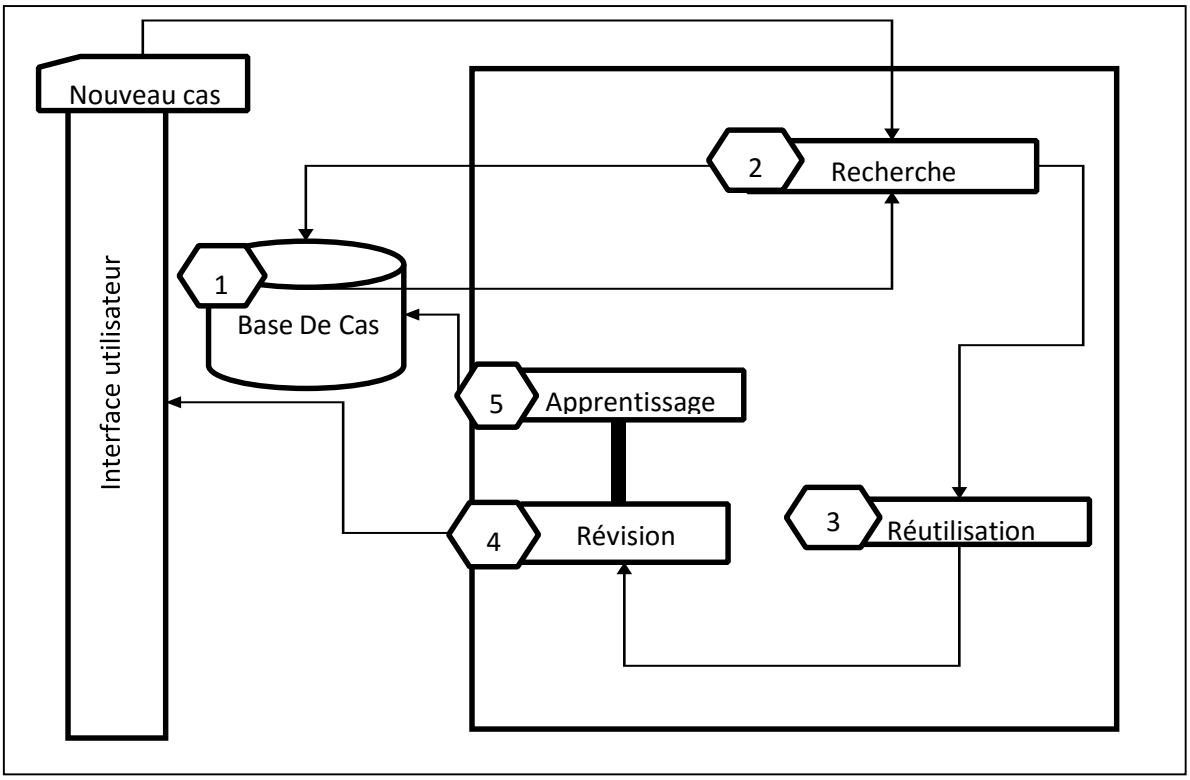


Figure 4.1 : L'approche proposée

1- La base de cas remplie par des cas réels issus des données fournies par les dossiers des enfants victimes d'accidents de la route. Ces dossiers ont été consultés sur place, au service de réanimation pédiatrique de l'EHS de Canastel.

2- La remémoration se fait dans la plateforme jCOLIBRI. Quand un nouveau cas arrive les cas similaires sont sélectionnés en se basant sur le calcul de similarité

3- Les cas similaires seront adaptés au nouveau problème pour trouver la solution adéquate ; ou plutôt la thérapie pour la prise en charge de l'enfant victime d'un accident de la route.

4- La révision consiste à valider la solution adaptée par l'expert qui est le médecin.

5- Si la solution générée n'existe pas dans la base de cas et elle est considérée comme un succès dans l'état de révision, ce nouveau cas est ajouté à la base de cas ce qui permet au système d'apprendre.

## 6- Le Schéma explicatif

La figure au-dessous donne une représentation graphique des interactions entre les différents acteurs. Dans cette section nous allons présenter les étapes décrit ci-dessous qui donne une vue globale de notre application :

1- Dans le site d'accident le ramasseur du SAMU remplit la fiche de renseignement sur le site d'accident concernant l'enfant accidenté.

2- Le SAMU envoie la fiche de renseignement au service d'urgence pour commencer la prise en charge de l'enfant accidenté.

3- Dans la salle d'urgence, le médecin consulte la fiche de renseignement et saisit les données dans l'application que nous proposons.

4- Le système remémore les cas similaires et propose une solution de cas antérieure jugé similaire.

5- Le médecin décide si la solution est appropriée pour le nouveau cas si oui, il l'applique directement sinon, il opère quelques modifications.

6- En utilisant le système et la décision de médecin l'enfant est envoyé au service de réanimation pédiatrique pour le traitement en se basant sur la solution proposée par le système (et éventuellement modifiée par le médecin).

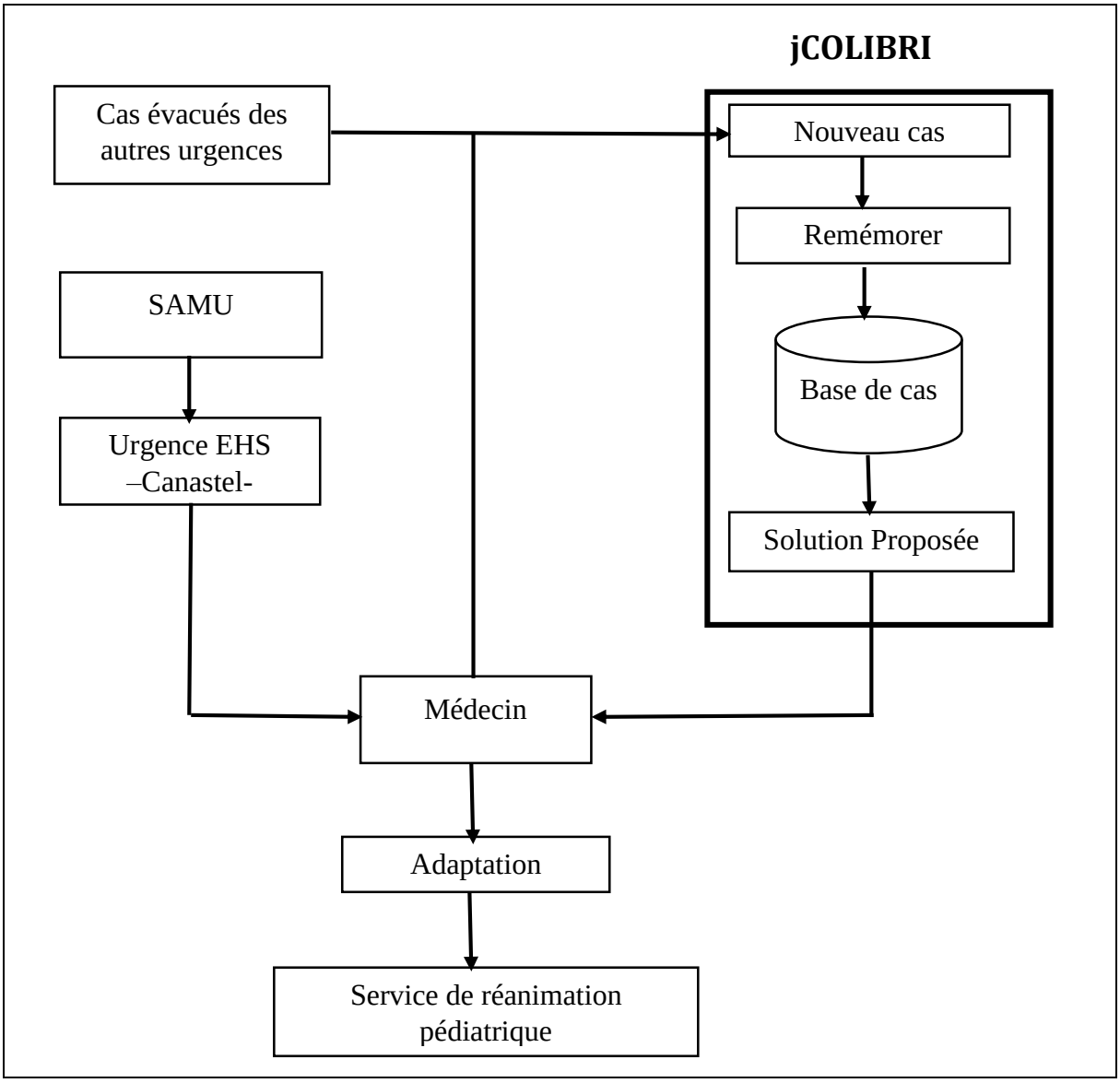


Figure 4.2 : Le schéma explicatif

## 7- Conclusion

Les systèmes d'aide à la décision jouent un rôle très important dans tous les domaines, en particulier lorsque le temps de prise de décision devient critique. Il est bien évident que dans l'application que nous traitons, le temps est primordial puisqu'il s'agit le plus souvent de cas très graves qui exigent une prise de décision rapide. La solution proposée s'appuie sur des données réelles fournies par des médecins de l'EHS de Canstel. Ces données ont servi pour la construction de la base de cas de départ.

Il est bien évident que cette solution ne peut être adoptée que lorsqu'elle montre son utilité dans l'aide à la décision pour l'expert. Il faut évidemment faire des expériences et les comparer avec des situations réelles afin de « valider » l'application. Le chapitre suivant montre quelques expériences menées en utilisant cette application.

# Chapitre 05

## Description de La Solution Mise en Œuvre

## 1- Introduction

Après la réalisation du système d'aide à la décision pour la prise en charge d'un enfant victime d'un accident de la route, nous nous intéressons à l'évaluation des résultats fournis par ce dernier. Pour cette raison, dans ce chapitre, nous allons commencer par un test de validation et de comparaison entre notre système et le système d'origine fourni par jCOLIBRI. Ensuite, une description générale des interfaces principales qui permettent l'exploitation de l'application.

## 2- Test de Validation

JCOLIBRI fournit l'exemple « Travel Recommender » sous deux versions. La première version est une boite noire où l'on se restreint à saisir des requêtes à travers une interface et recevoir les réponses du système. La seconde version est une boite blanche destinée aux développeurs et dans laquelle tout est modifiable pourvu que la structure générale du système reste fidèle à l'architecture de jCOLIBRI décrite au chapitre 3. Cet exemple propose des voyages basés sur une requête saisie par un client. Afin de tester la fonctionnalité et la cohérence des mesures de similarité que nous avons ajoutées au système. Il s'agissait ensuite de lancer la même requête sur le système d'origine (de base) et ensuite sur le système que nous avons modifié. Le tableau suivant représente chaque attribut (de l'exemple « TravelRecommender ») avec la mesure de similarité proposée dans jCOLIBRI et la mesure de similarité « test » que nous avons proposé :

| Attributs | Mesure de similarité initiale | Mesure de similarité du test |
|---|---|---|
| HolidayType | Equal | Jaro Winkler |
| Number of persons | Threshold | Euclidienne |
| Region | OntCosine | OntCosine |
| Transportation | Equal | Jaro Winkler |
| Duration | Interval | Euclidienne |
| Seasons | EnumDistance | EnumDistance |
| Accomodation | EnumDistance | EnumDistance |

Tableau 5.1 : Table de comparaison entre les mesures de similarité

## 2.1. La requête d'entrée

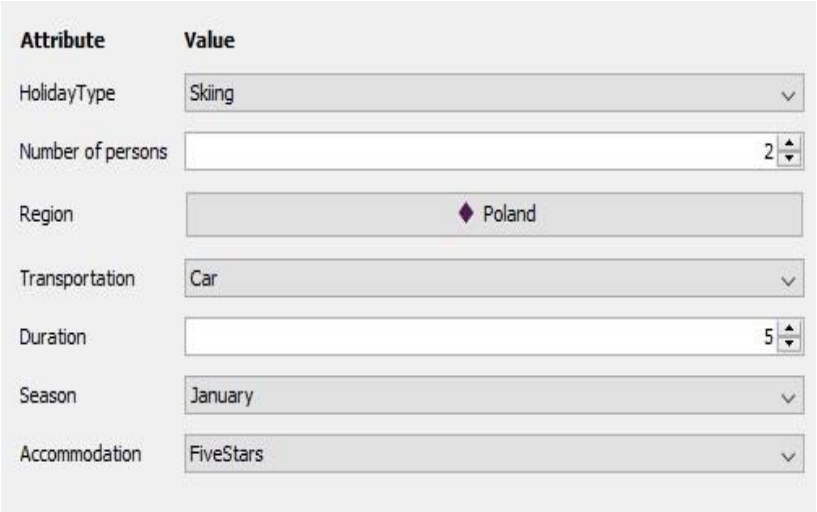


Figure 5.1 : La requête d'entrée pour les deux exemples

## 2.2. Résultats pour l'exemple (boite noire)

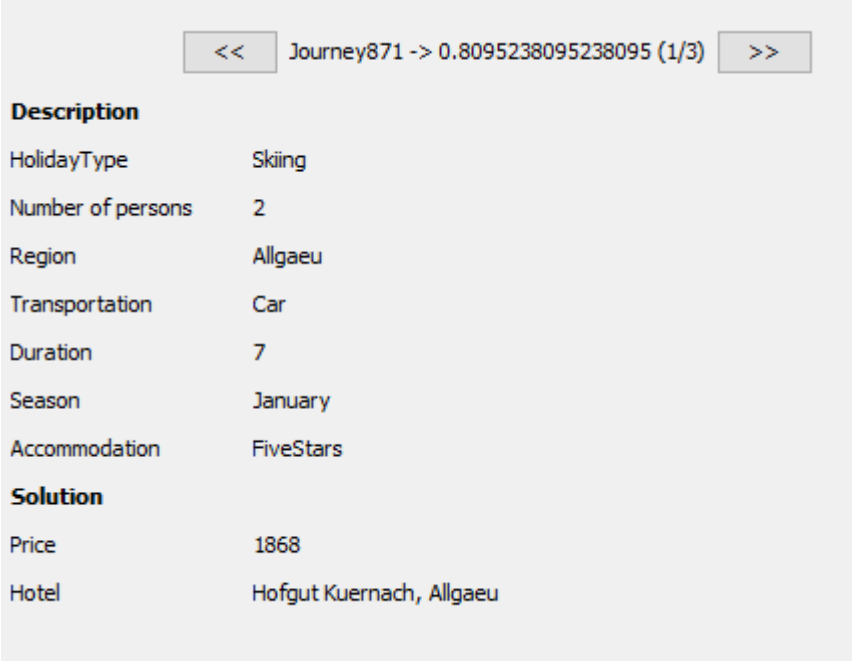


Figure 5.2 : résultat de la requête par la boite noire

La figure 5.2 représente le résultat de la requête ; c'est-à-dire le cas remémoré. Elle comporte une description du cas et la valeur de la mesure de similarité correspondante (en haut de la figure).

### 2.3. Résultat de la boite blanche modifiée


Figure 5.3 : résultat de la requête par la boite blanche

### 2.4. Explication des résultats

Pour l'exemple de jCOLIBRI, la similarité de classification prise est celle qui est proche de 1 car leur mesure de similarité retourne1 si deux attributs sont égaux. Dans notre application la similarité est une mesure de distance, donc lorsque deux attributs sont égaux elle est égale à 0. Ceci explique les valeurs différentes des mesures de similarité.Néanmoins, on peut remarquer que les deux systèmes retourne la même solution ( ou bien remémore le même cas). Nous avons mené plusieurs expériences de ce type et nous avons remarqué des résultats pareils.

## 3- Présentation du Système

Dans notre système d'aide à la décision nous nous intéressons à l'étape de remémoration car elle représente le cœur des systèmes basés sur le RàPC. En ce qui concerne l'étape d'adaptation de la solution, cette étape est laissée aux experts du domaine car il n'y a pas de règlesprédéfinies et le domaine médicale est un domaine délicat ou une simple faut geste risque de nuire à la santé d'un patient. D'autre part, une adaptation manuelle n'est pas considérée comme un aspect négatif. De plus, dans [9], les auteurs déclarent : « quand le domaine de la connaissance n'est pas vraimenttrès clair,une adaptation automatique est difficile à développer ou bien déconseillée ».

## 4- Les Interfaces graphiques

Les interfaces représentent le moyen de communication et d'exploitation de toute application. Après le lancement du système, une interface graphique contenant le cycle du RàPC est lancée. En même temps, et en arrière-plan, les fonctions principales de chargement de la base de cas et de configuration sont lancées ou même temps. Une fois le chargement terminé, le système se présente sous forme d'une interface simple qui permet de lancer une requête. Cette requête représente évidemment le cas d'un nouvel enfant victime d'un accident de la route.

### 4.1. L'interface Nouveau cas

Dans cette interface le médecin (pédiatre) ou bien un agent de SAMU saisit les informations concernant la victime et les autres informations décrites dans le chapitre précédent en ce qui concerne les attributs de nouveau cas.

En plus, nous avons fourni la possibilité au médecin de définir le nombre de cas à retenir. Cette dernière possibilité nous a obligé de modifier la fonction « *Cycle* »de l'interface principale de jCOLIBRI« *StandardCBRApplication* »en ajoutant un paramètre pour permettre aux expert » de choisir le nombre de cas à remémorer pour donner un décision efficace qui peut par la suite sauver une vie d'une victime ou au moins réduire les conséquence graves.

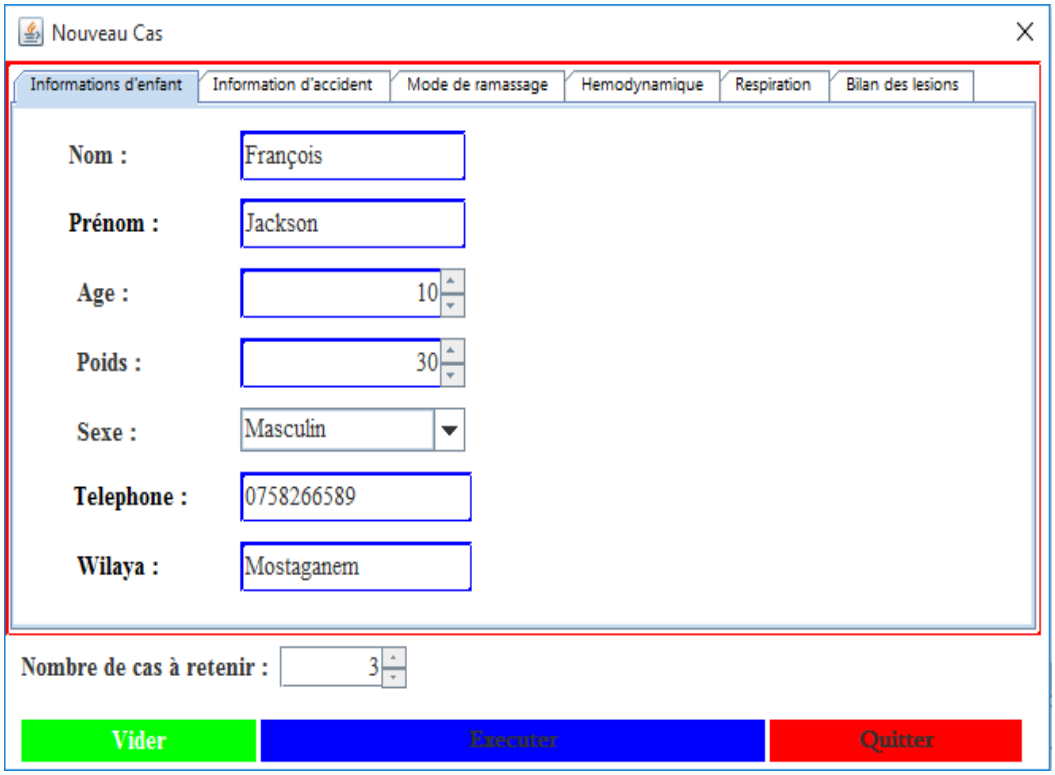


Figure 5.4 : Interface pour entrer le nouveau cas

### 4.2. L'interface du résultat

Le rôle de cette interface est d'afficher à l'utilisateur les meilleurs cas résultants de la remémoration. Ces cas sont affichés dans un ordre décroissant de similarité. L'utilisateur (médecin du service d'urgence pédiatrique), peut consulter les cas un par un et ensuite choisir le cas qu'il pense le plus approprié. Après ce choix, l'utilisateur peut apporter des modifications (ajustements) à la solution proposée. Cette solution avec la requête représentent un cas que le système vient de résoudre.

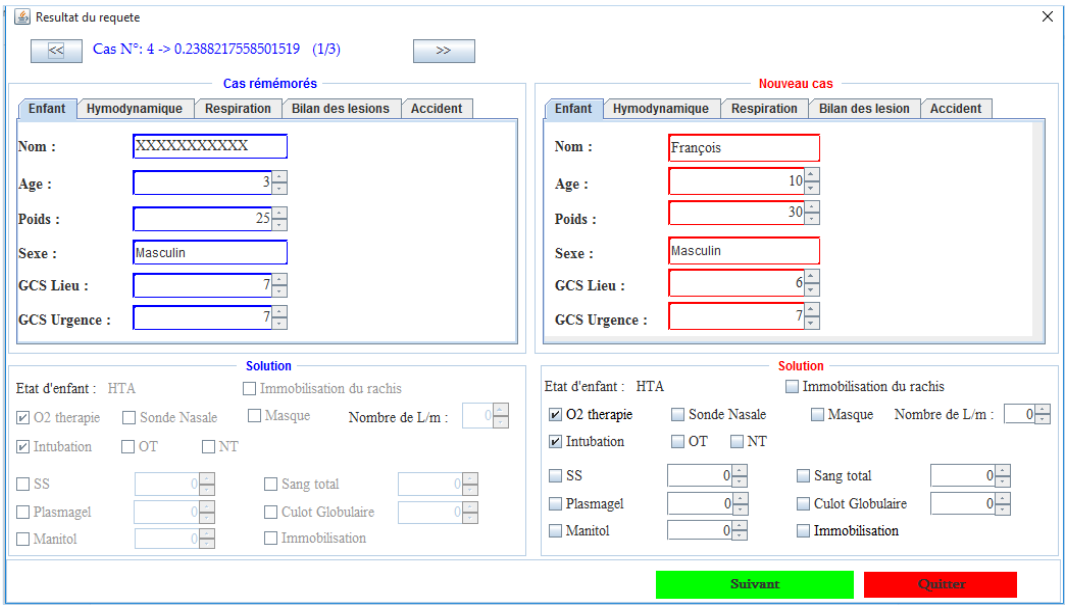


Figure 5.5 : L'interface du résultat

### 4.3. L'interface de Rétention

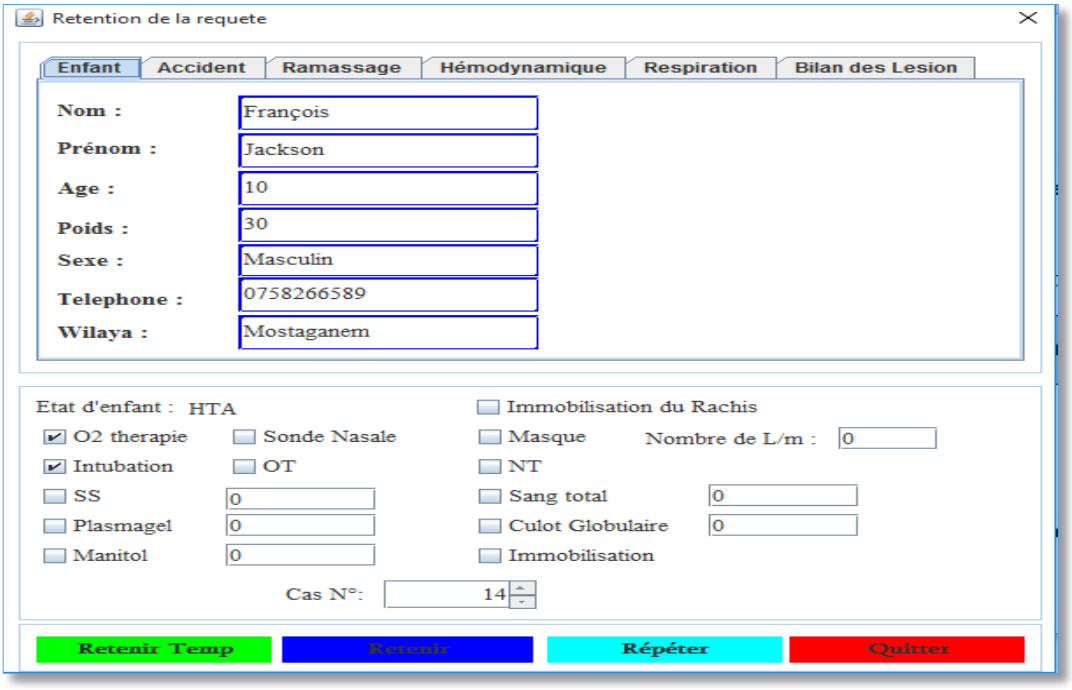


Figure 5.6 : L'interface de rétention

Une fois la solution ajustée (ou adaptée) par le médecin, l'interface de rétention donne la possibilité de retenir ce nouveau cas ou bien de le laisser dans une base de cas temporaire que nous avons ajouté dans notre application. Si le cas est retenu, il est intégré dans la base de cas (apprentissage). Le médecin peut décider de se donner du temps supplémentaire avant de décider de la rétention de ce nouveau cas. De plus, cette même interface donne la possibilité de saisir des informations sur le patients (qui n'ont pas déjà été saisies), par exemple son état de sortie. Par la suite, le médecin peut, de temps en temps, consulter la base de cas temporaire afin de compléter des cas et/ou décider de retenir quelques cas.

## 5- L'application de consultation des cas

C'est une application qui ne fait pas partie du cycle du RàPC. Nous avons décidé de l'ajouter afin de permettre une gestion plus facile de la base de cas. Cette application permet d'afficher, un par un, les cas présents dans la base de cas, ou bien les cas qui sont encore dans la base de cas temporaire.

### 5.1. L'interface de consultation de la base de cas temporaire

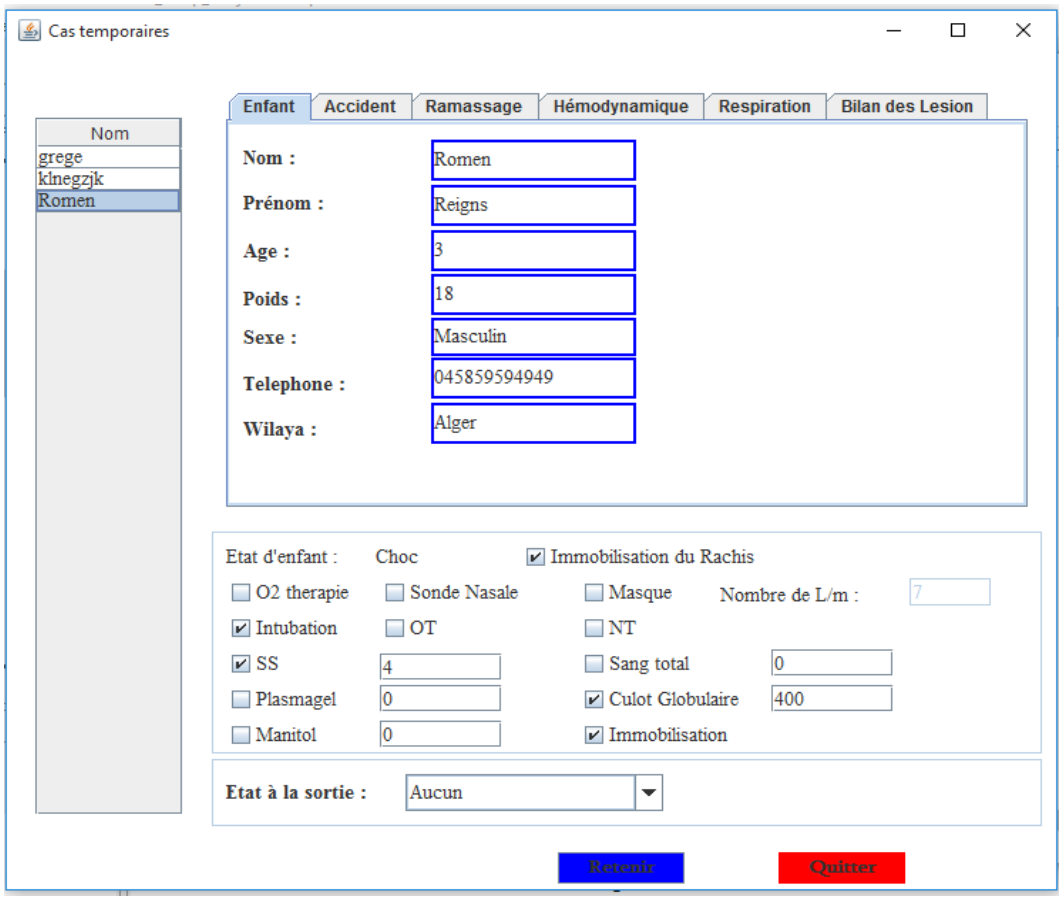


Figure 5.7 : L'interface de consultation de la base de cas temporaire

### 5.2. Consultation des cas de la base

Cette interface montre les cas dans la base de cas si l'expert décide de les voir mais dans cette interface il n'y a aucune possibilité de modification.

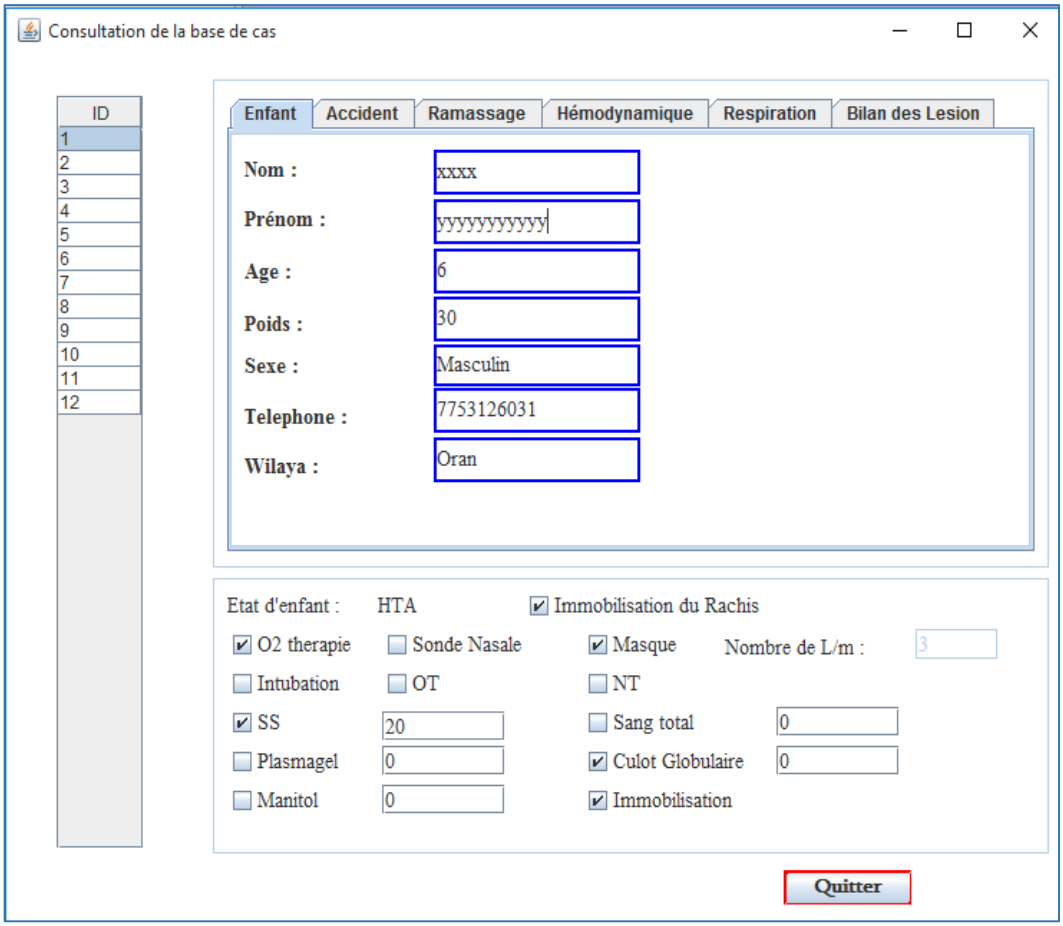


Figure 5.8 : L'interface de consultation de la base de cas

## 6- Conclusion

Le but de ce chapitre était de présenter le système d'aide à la décision médicale ou nous avons présenté les interfaces principales de ce système. Ce système est conçu pour aider les médecins pour une prise en charge rapide d'un enfant victime d'un accident de la route. Ce projet est fait en collaboration avec l'établissement hospitalier spécialisé de Canastel –Oran-. Le point fort de ce système est qu'il est basé sur une base de cas avec des cas réels qui sont évalués par les experts de l'établissement.

# Conclusion Générale

Depuis l'apparition des premiers ordinateurs les chercheurs se sont lancés dans des projets dont l'objectif était la conception d'une machine pensante, ou une machine qui reproduit le raisonnement humain. Les premiers travaux ont donné lieu au concept d'intelligence artificielle. Dans ce cadre il y a eu un développement massif des systèmes à base de règles, ou systèmes experts. Ces systèmes sont basés sur la déduction.

Des travaux de recherche menés durant les années 80 ont montré que les humains avaient des modes de raisonnement variés et que le raisonnement déductif n'était qu'une facette. Un autre mode très utilisé par les humains est le raisonnement inductif ou raisonnement par analogie. Le RàPC rentre dans ce cadre et cette méthodologie et a pu proposer des solutions dans des domaines d'application très variés.

Les systèmes d'aide à la décision médicale font partie des domaines dans lesquels beaucoup de travaux ont proposé des solutions basées sur le RàPC. Le but de ce projet en premier lieu était de comprendre la méthodologie du RàPC puisque elle ne faisait pas partie du cursus de Master. Ensuite, et en se basant sur cette méthodologie, de concevoir et mettre en œuvre une application d'aide à la décision médicale pour la prise en charge de l'enfant victime d'un accident de la route.

Le système proposé est réalisé en utilisant la plateforme jCOLIBRI ; une plateforme qui implémente le cycle du RàPC et qui constitue un cadre aux développeurs d'applications basées sur le RàPC. La mise en œuvre de cette application a nécessité une exploration en profondeur et dans les fins détails de la plateforme jCOLIBRI. En effet, sans cette exploration, il n'est pas possible de modifier, ou d'ajouter, du code dans la plateforme.

Cette solution a été montée à partir de données réelles fournies par des médecins de l'établissement hospitalier spécialisé (EHS) de Canastel, Oran. Ceci a nécessité des rencontres avec ces médecins afin de comprendre la procédure de prise en charge de l'enfant victime d'un accident de la route. Ensuite, une analyse de quelques dossiers fournis a été nécessaire afin de décider comment les cas seront représentés. Il était difficile de décider, à partir de dossiers fournis sous format papier, où est le problème et où est la solution.

Au bout de ce projet, une solution a été mise en œuvre et testée sur des cas réels. On ne peut pas encore parler de validation car le nombre de cas fournis par l'EHS est réduit, et de plus, il faut donner le temps à une solution, avant de pouvoir faire un jugement juste. Dans un premier temps, l'acquisition de nouvelles données (nouveaux dossiers) permettra d'accroitre l'expérience emmagasinée dans la base de cas, ce qui permettra de rendre le système plus efficace. Ensuite, la solution doit être utilisée pendant une période de test au bout de laquelle des lacunes peuvent être décelées.

Ce projet a constitué une opportunité immense d'apprentissage. Tout d'abord, sur le plan théorique, la compréhension du RàPC ainsi que tous les aspects liés à cette méthodologie ont engendré beaucoup de lectures et de consultation de documents de recherche récents. Ensuite, la maitrise de la plateforme utilisée a aussi nécessité une multitude de lectures et de tests pratiques afin de relever les principales classes et méthodes où le développeur peut opérer sans casser la philosophie de la plateforme ni le cycle du RàPC pris en charge par cette dernière.

Ce projet ne constitue pas une fin en soi, mais ouvre la voie vers une plus grande collaboration entre l'université et les milieux hospitaliers. Le but étant d'améliorer la qualité de service offert par ces milieux et permettre une meilleure prise en charge de la santé publique. D'un autre côté, l'utilisation effective de cette solution prendra toute sa grandeur lors du développement d'applications mobiles qui permettent de communiquer directement avec le service d'urgence. En effet, ceci permettra de recevoir les informations sur l'accident bien avant l'arrivée de l'enfant ce qui accroitra les chances de survie, mais aussi diminuera les éventuelles séquelles.

Sur un autre plan, il reste du travail dans le cœur même du système. En effet, ce projet n'a pas pris en considération l'adaptation des solutions et s'est contenté d'une adaptation manuelle faite par l'expert. Il est légitime de penser à une adaptation automatique, ou semi-automatique, une fois la solution déployée. L'idée est d'enregistrer les modifications faites par l'expert sur la solution proposée en vue de dégager des règles qui capitalisent cette expertise. Ceci ne peut se faire que lors d'une utilisation sur une période suffisamment longue.

Un autre aspect à prendre en charge dans les travaux à venir est la gestion de la base de cas. Dans ce projet, aucun module de gestion n'a été proposé mais la structure du cas a été pensée dans ce sens. En plus du problème et sa solution, la structure du cas (dans notre application) intègre « le résultat » de l'application de la solution ainsi que la « pertinence » du cas. Ces deux aspects peuvent être pris en considération pour proposer une gestion périodique de la base de cas pour éviter qu'elle devienne trop volumineuse.

# Bibliographies


1. A. Aamodt, E. Plaza, (1994); Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7: 1, pp. 39-59.

2. A. Bottrighi, G. Leonardi, S. Montani, L. Portinale Extending the JColibri open source architecture for managing high-dimensional data and large case bases, Universit´a del Piemonte Orientale "A. Avogadro", Alessandria (Italy).

3. B. Benhaoua, (2016) ; Conception d'un module d'aide à la décision basé sur le RàPC pour le service de réanimation pédiatrique du CHU d'Oran, Thèse de Master, Université d'Oran 1 ahmed Benbella.

4. F. Fürst. L'histoire de l'IA. Support de cours, Université de Picardie, Département d'Informatique ; www.u-picardie.fr/~furst/docs/5-Crise_IA.php.

5. F. Henni, (2015); Composition Dynamique de Services Web par Apprentissage Artificiel. Thèse de Doctorat Es-Sciences en Informatique. Université d'Oran 1, Ahmed Benbella. Chap 4 : Le Raisonnement à Partir de Cas.

6. G. Dvir, G. Langholz, M. Schneider, (1999) ; Matching attributes in a fuzzy case based reasoning. Fuzzy Information Processing Society, pp. 33–36.

7. G. Salton, C Buckley,(1988); Term weighting approaches in automatic text retrieval, Information Processing and Management, Volume 24, Issue 5, p513-23.

8. H. Prade; L'intelligence Artificielle, mais enfin de quoi s'agit-il ? Institut de Recherche en Informatique de Toulouse, Les livrets du Service Culture UPS n°3.

9. I**.** Nasrullah, A. M. Hassan, (2006); Evaluation of JCOLIBRI, Rapport de these de master, Malardalen University, Vasagatan 44, 72123 Vasteras, Sweden.

10. I. Watson, (1999); Case-based reasoning is a methodology not a technology. Knowledge-Based Systems 12 .p 303–308.

11. I. Watson, F. Marir, (1994); Case-Based Reasoning: A Review. Cambridge University Press, The Knowledge Engineering Review, vol. 9, issue 4, pp. 355-381.

12. J. Antonio Recio García,(2008); jCOLIBRI: A multi-level platform for building and generating CBR systems. these de doctorat. Universidad Complutense de Madrid.

13. J. A.Recio-Garcìa, B. Dìaz-Agudo, P. Gonzalez-Calero, (2008). JCOLIBRI2 Tutorial. Group for artificial intelligence applications. Universidad Compultense de Madrid

14. J. F. De Paz, S. Rodríguez, J. M. Corchado; Case-based reasoning as a decision support system for cancer diagnosis: A case study. Departamento de Informática y Automática. Universidad de Salamanca Plaza de la Merced, Salamanca, España.

15. J. Philippe Raymond, (2005) ; Comparaison de fonctions de similarité entre chaînes de caractères dans une tâche de détection de doublons approximatifs ;These de Master.

16. K. Dreßler , A. Ngonga Ngomo ; On the Efficient Execution of Bounded Jaro-Winkler Distances. AKSW Research Group, University of Leipzig, Germany.

17. M. Gazzah ; le score de Glasgow e-Formation médecine d'urgence - www.efurgences.net

18. M. Uddin Ahmed, S. Begum, P. Funk, (2012); Case Studies on the Clinical Applications using Case-Based Reasoning. Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 3–10.

19. M. Uddin Ahmed, S. Begum, P. Funk, N Xiong, B Von Schéele, (2008). Case-based Reasoning for Diagnosis of Stress using Enhanced Cosine and Fuzzy Similarity. Transactions on Case-Based Reasoning for Multimedia Data, Journal Vol.1, No 1; 3-19.

20. Shaker H. El-Sappagh, M. Elmogy, (2015); Case Based Reasoning: Case Representation Methodologies, IJACSA International Journal of Advanced Computer Science and Applications, Vol. 6, No. 11,pp 192-208.

**Webographie**

21. – community Documentation, chapter 03 :configuration ;www.hibernate.org, consulté le 01/05/2017.