Université Victor Segalen - Bordeaux 2

Année 2000 Thèse n°712

THÈSE

pour le

DOCTORAT DE L'UNIVERSITÉ DE BORDEAUX 2

Mention : Sciences Biologiques et Médicales

Option : Épidémiologie et Intervention en Santé Publique

présentée et soutenue publiquement le

14 Janvier 2000

par

Marie Noëlle BEURTON AIMAR

ARCHITECTURE D'UN SYSTÈME D'AIDE AU DIAGNOSTIC MÉDICAL : APPLICATION EN RHUMATOLOGIE INFLAMMATOIRE

Membres du Jury

Monsieur le Professeur	R. SALMI	Président
Monsieur le Professeur	M. FIESCHI	Rapporteur
Monsieur le Professeur	F. RECHENMANN	Rapporteur
Monsieur le Docteur	B. LE BLANC	Examinateur
Monsieur le Professeur	T. SCHAEVERBEKE	Examinateur
Monsieur le Professeur	R. STRANDH	Examinateur
Monsieur le Professeur	R. SALAMON	Directeur de thèse

Résumé

Les avancées actuelles de l'informatique médicale et l'introduction massive des ordinateurs dans le secteur de la santé ont modifié la nature des informations manipulées par les systèmes d'aide au diagnostic. Ils doivent être en mesure de mémoriser les informations, de rechercher une pathologie spécifique, de consulter des tableaux cliniques types et de comparer les résultats pour plusieurs pathologies. Ceci suggère la définition de modules fortement inter-connectés, partageant une base d'informations commune. Le partage des informations est possible si la base de connaissances est structurée indépendamment des traitements utilisés. Nous avons adopté une architecture orientée objet pour concevoir cette base de connaissances et obtenir un modèle de collaboration entre les différents modules de traitement.

Les problèmes à résoudre concernent la modélisation de manière uniforme des maladies qui présentent des signes pathologiques hétérogènes, l'identification de comportements génériques à l'intérieur des différentes procédures de traitement et la définition d'une interface qui garantisse la communication à travers le système. L'intégration de quatre « design patterns » dans le modèle objet issu de l'analyse du domaine, a permis de résoudre ces problèmes. `Composite` et `Etat` interviennent sur la structure des signes. `Itérateur` et `Stratégie` modélisent des tâches et des procédures génériques de diagnostic. Ces quatre patterns décrivent quatre points stratégiques d'un système d'aide au diagnostic.

Un Logiciel d'Aide au Diagnostic en Rhumatologie Inflammatoire (LADRI) a été développé. Sa base de connaissances a été élaborée en associant les connaissances d'un praticien hospitalier et des synthèses de la bibliographie du domaine.

La discussion de ces travaux porte essentiellement sur l'apport du Génie Logiciel dans des applications relevant de l'Intelligence Artificielle, et sur la réelle réutilisabilité d'une telle architecture objet.

Abstract

The improvement of medical informatics and the introduction of computers in all medical fields have modified the nature of information manipulated by diagnosis support systems. They must be able to save information, to search a specific pathology, to look forward clinical tables and to compare results for some pathologies. This suggests the definition of different modules of treatment, sharing a common data base. To share information is made possible if the knowledge data base is designed independently to the used procedures. We have adopted an object oriented design to create the knowledge data base and to obtain a model of collaboration between the treatments modules.
The problems to solve are to design in the same way pathologies which have heterogeneous signs, to identify generic behaviours inside treatment procedures, and to define interface which guarantees comunication through the system. The association of four design patterns with the object model issued from the domain analysis, have raise possible the solution of these problems. `Composite` and `State` have a part in the sign structuration. `Iterator` and `Strategy` design some generic tasks and procedures for diagnosis. These four patterns describe four strategic points of a diagnosis support system.
A software LADRI (Logiciel d'Aide au Diagnostic en Rhumatologie Inflammatoire) was developped. Knowledge data base was elaborated from an expertise of a practitionner, and summaries of domain bibliography.
The discussion of these works is essentially focused on the contribution of the Software Engineering in the fields of Artificial Intelligence and on the real reusability of such object design.

Table des matières

Introduction

Le diagnostic médical est une activité fondamentale de la médecine. Tout professionnel de la santé se doit à un moment ou à un autre d'établir une classification de son patient. Il est donc normal que les premières investigations dans le domaine du traitement automatique des informations se soient intéressées à ce problème.

Les pionniers de l'informatique médicale comme R. Ledley et L. Lusted [87], E. Shortliffe [124], F. De Dombal [39], ou F. Gremy [61] en France, ont cherché à automatiser l'élaboration d'un diagnostic. Les méthodes développées s'appuyaient sur des modèles statistiques (calcul de probabilités, théorème de Bayes) et/ou des modèles logiques issus des recherches menées en Intelligence Artificielle.

La complexité du domaine médicale et les progrès de l'informatique ont provoqué l'évolution du concept initial de *Système de Diagnostic* vers un concept plus général de *Système d'Aide à la Décision Médicale*. Ce changement de point de vue a permis d'élargir le champs de vision de cette discipline en y intégrant des paramètres comme les choix thérapeutiques, les désirs et les préférences du patient en face de ces choix, le coût des examens complémentaires, les capacités techniques ou financières du système de santé, etc.

La prise en compte de tous ces éléments a conduit *de facto* les concepteurs de ces systèmes à reconsidérer la position du diagnostic dans le système de santé et à privilégier la notion d'`aide` qu'un système informatique peut apporter à tous les décideurs. Dans cette approche, l'aide au diagnostic est devenue l'un des éléments d'un système d'information plus vaste.

Parallèlement, dans tous les domaines d'application de l'informatique (gestion ou industrie), le nombre de programmes développés et la complexité toujours croissante des traitements ont provoqué l'émergence de nombreuses méthodes d'ingénierie logicielle. Les problèmes soulevés par l'architecture et la maintenance de ces applications sont à l'origine d'une nouvelle discipline : le *Génie Logiciel*. La construction d'un ensemble d'applications destinées à évoluer dans le même environnement informatique a favorisé la mise en évidence de composants logiciels réutilisables.

Nous avons étudié ces problèmes d'architecture logicielle dans le cadre d'un système d'aide au diagnostic. Notre but n'est pas de porter un effort sur la compréhension du diagnostic en lui-même, mais de mieux cerner l'aide qu'un ordinateur peut apporter à un praticien dans sa démarche diagnostique. On cherche ainsi à identifier les tâches qui entrent dans cette démarche et que l'on peut réaliser automatiquement.

Pourquoi s'intéresser à nouveau au diagnostic médical alors que les expériences passées ont montré que les systèmes développés dans ce domaine ont globalement abouti à une impasse ?
L'inadéquation entre les systèmes réalisés et la pratique médicale peut brièvement s'analyser en trois points :

1. Pendant que l'ensemble des recherches se focalisaient sur la construction d'un diagnostic, les praticiens se détournaient de ces « *boites noires* » qui affichaient la prétention de les priver de la partie la plus intéressante de leur activité ; alors même qu'ils n'émettaient pas la requête d'obtenir une aide dans ce domaine [67]. Comme le remarque R. Salamon dans sa préface de l'ouvrage de B. Grenier [62], ces systèmes ont été perçus comme les instruments d'une intrusion de l'informatique dans le secteur de la santé et non pas comme une réponse aux véritables préoccupations

des praticiens.

2. Les modèles de raisonnement développés dans les années 60-80 s'appuyaient sur des théories statistiques complexes dont l'appropriation des résultats nécessitaient des connaissances que peu de praticiens possédaient, ou sur des méthodes basées sur l'implication logique, incapables de modéliser la complexité des situations réelles [80].
3. La mise en œuvre et l'utilisation des logiciels ainsi que la diffusion des résultats étaient tributaires d'un environnement informatique lourd, peu performant et maîtrisé par un nombre restreint de personnes.

En quoi la situation actuelle a-t-elle changé les données du problème ?

Les changements les plus profonds touchent le troisième point que nous venons d'évoquer. Pour les praticiens, comme pour l'ensemble de la population, les ordinateurs font maintenant partie du quotidien. En conséquence, l'accès à l'information (au sens large) en est grandement facilité. Les promoteurs de systèmes basés sur la télématique (comme ceux qui se sont appuyés sur le minitel en France), n'auraient pu imaginer la révolution causée depuis ces cinq dernières années par le réseau Internet. La diffusion d'informations à grande vitesse, en grande quantité et quasiment sans limitation géographique a profondément bouleversé les méthodes d'échange de données dans l'ensemble du monde professionnel (privé ou public) et dans le monde éducatif (de l'école à l'université).

L'informatique a fait son apparition à tous les niveaux du système de santé. Les anciens systèmes centralisés qui étaient utilisés uniquement par un personnel spécialisé, ont été remplacés par de petits ordinateurs reliés en réseaux (intranet ou internet). Chaque unité de soins peut y introduire des modules qui lui sont propres, comme la gestion de rendez-vous ou le traitement d'une enquête épidémiologique ponctuelle. Le médecin interroge directement et librement des bases de données bibliographiques comme `Medline`. De nombreux laboratoires d'analyses transmettent directement sous format électronique leurs résultats que le praticien peut consulter « *en ligne* ». Le dossier informatisé du patient est mis en place dans les hôpitaux, par les Départements d'Informatique Médicale, et la carte `Vitale` devrait permettre aux caisses d'assurance maladie de suivre leurs assurés quel que soit le praticien ou l'établissement de soins auxquels ils s'adressent. Dans des secteurs spécialisés comme la radiologie ou la chirurgie, les innovations technologiques en matière de traitement des images ou de robotique ont profondément modifié le travail de l'ensemble des intervenants.

En ce qui concerne le deuxième point, pour les méthodes relevant de l'Intelligence Artificielle, la recherche sur la modélisation des raisonnements peut s'appuyer sur des extensions de la logique classique comme la logique multi-valuée, la logique floue ou les logiques modales. Le développement de méthodes basées sur les réseaux de neurones ouvre de nouvelles voies à la représentation de processus de classification. Les capacités accrues des ordinateurs pour l'exécution des programmes et le stockage des informations autorisent le développement d'algorithmes toujours plus complexes. L'intégration de l'enseignement de l'épidémiologie, de la statistique et de l'informatique dans certains cursus médicaux, a familiarisé les différents décideurs à la manipulation des outils relevant de ces disciplines. Ces outils sont devenus faciles d'accès grâce à des logiciels de plus en plus ergonomiques, même si cette accessibilité plus grande a révélé de nouveaux problèmes dûs à une utilisation inconsidérée et « *en aveugle* » des logiciels « *presse-boutons* » [118].

Enfin, qu'en est-il de la demande des praticiens ? Même si elle n'a pas fondamentalement évoluée, on constate que les médecins doivent aujourd'hui manipuler un nombre toujours plus grand d'informations (accroissement d'ailleurs dû dans la plupart des cas à l'informatique). Dans ce contexte, J.H. van Bemmel [134] explique que l'ordinateur peut devenir un support important dans l'étape diagnostique, mais que cela concerne uniquement la partie du diagnostic qui peut être structurée, généralisée et rendue objective. Ceci signifie que les ordinateurs ne doivent pas se substituer au médecin mais au contraire l'assister dans son travail. Les éléments du diagnostic qui traitent avec les problèmes uniques et individuels du patient ne peuvent pas être et ne doivent pas être laissés à la machine.

L'aide fournie par l'ordinateur, peut se matérialiser par une liste de suggestions faites au praticien quant à la conduite à tenir, la proposition d'un ensemble de diagnostics à envisager ou des avertissements au sujet de contre-indications thérapeutiques [110].

Systèmes d'aide au diagnostic et développements orienté objet.

Le but de ce travail est d'étudier l'architecture des systèmes d'aide au diagnostic et de définir des éléments génériques (du point de vue informatique) qui interviennent dans la modélisation de ces systèmes. Nous avons pour cela utilisé des *modèles de conception réutilisables* dénommés par leur appellation anglo-saxonne, *design patterns*. Notre modèle a été appliqué en rhumatologie inflammatoire, domaine où l'établissement d'un diagnostic repose sur un faisceau d'arguments plutôt que sur des « *gold standards* ». Dans ces conditions, ce domaine nous a fourni un cadre adéquat pour tester l'architecture d'un système dont l'objectif est d'aider les praticiens à rassembler des informations pouvant avoir plusieurs origines (littérature, cas cliniques types, expertise, etc), et à garder en mémoire une description du patient facilement comparable aux descriptions d'un groupe de pathologies possibles.

Les systèmes d'aide au diagnostic médical tels que nous les avons abordés, sont des systèmes à base de connaissances, c'est-à-dire qu'ils contiennent un modèle de représentation des données à traiter et des raisonnements qui leurs sont appliqués.

P. Degoulet et M. Fieschi [41] remarquent que les informations manipulées par de tels systèmes sont de trois types (cf figure 1) : les connaissances académiques contenues dans les publications et ouvrages scientifiques, l'expérience acquise dans l'exercice médical, et l'observation clinique rapportée dans le dossier patient (au sens large, incluant les différents documents d'examens complémentaires). Bien que ces trois types d'informations soient très fortement liés par le langage médical, leur prise en compte dans un système à base de connaissances nécessite des efforts de structuration importants [84].

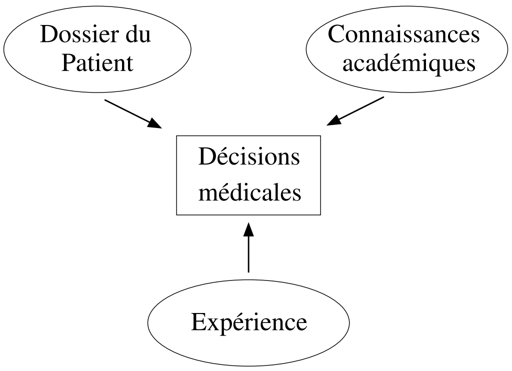

FIG. 1: les connaissances intervenant dans la décision

Les méthodes d'ingénierie logicielle décrivent plusieurs étapes pour la réalisation d'un système informatique. La première étape est une étape d'analyse de l'ensemble des données et des traitements manipulés par le système. La deuxième étape est celle de la conception d'un modèle informatique pour la représentation de ces données et traitements. Les deux dernières étapes concernent l'implémentation du logiciel et sa validation.

Contrairement à ce que prônent les méthodes d'analyse conçues pour des systèmes traditionnels, il n'est pas possible de baser le travail d'analyse d'un système à base de connaissances sur un cahier des charges ou un modèle des besoins. En effet, dans ces systèmes ce ne sont pas des besoins fonctionnels que l'on analyse mais un problème [46]. La différence essentielle vient de ce qu'un problème est en général mal compris et pauvrement formulé [63]. Il est donc indispensable d'utiliser des méthodes qui supportent de commencer la conception et même un début d'implémentation alors que le modèle de représentation des connaissances est incomplet et que les tâches à accomplir sont parfois imprécises. Les méthodes orientées objet appartiennent à cette catégorie. Ainsi, les phases d'analyse et de conception sont rassemblées dans le même cadre alors que généralement ces deux étapes sont considérées comme indépendantes dans des méthodes traditionnelles (Merise par exemple). Hodgson explique que ces deux phases se recoupent souvent :

> *"Le processus de développement de systèmes est un processus de compréhension, d'invention et de réalisation par lequel le domaine du problème est saisi, appréhendé sous forme de phénomènes, de concepts, d'entités, d'activités, de rôles et*

> *d'assertion. Il s'agit de compréhension et cela correspond entièrement à l'analyse. Cependant, comprendre le domaine d'un problème implique simultanément d'appréhender la structure, les composants, les modèles de calcul et autres constructions mentales qui prennent en compte le domaine des solutions possibles. Cette activité inventive correspond au processus de conception*" [71].

Le seul fait d'énoncer que l'approche objet s'intéresse à la *"composition"* du système en termes d'objets, ou qu'elle se centre sur *"ce que manipule"* le système plutôt que sur *"ce qu'il fait"* conduit à mêler plus ou moins intimement analyse et conception. Ceci apparaît très nettement dans la pratique de l'approche objet sur les projets : déterminer la vue statique des objets du système s'apparente à l'analyse, détailler la vue dynamique et le comportement de ces mêmes objets conduit rapidement à franchir la frontière de la conception car spécifier de manière détaillée le comportement d'un objet amène à se poser la question du *"comment"* de sa mise en œuvre, et donc de sa conception.

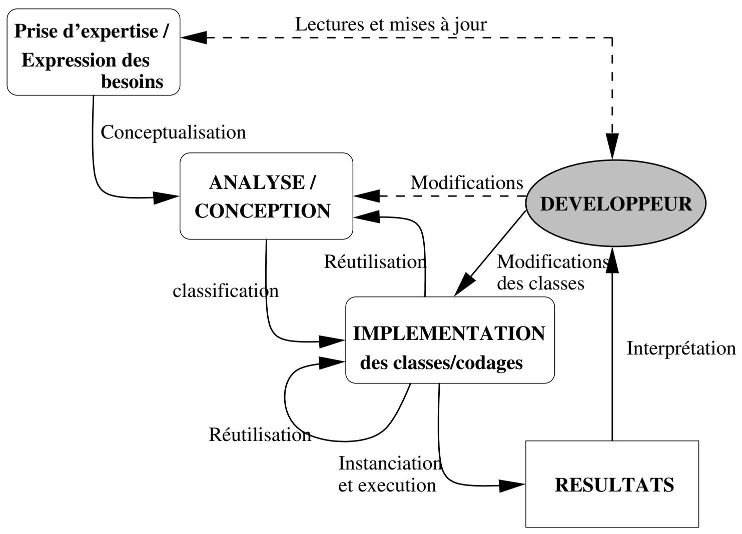

FIG. 2: les étapes du développement orienté objet

La figure 2 montre que le développement d'une application dans un environnement objet s'effectue par des cycles imbriqués : constructions de classes/créations d'objets et exécution [83].

Ce mémoire présente notre travail non pas dans la chronologie des étapes de développement d'un logiciel d'aide au diagnostic mais suivant un découpage thématique de ce développement : analyse, conception et réalisation.

La modélisation d'un système d'aide au diagnostic.

Le premier chapitre est centré sur l'analyse du domaine étudié. Il commence par une présentation des données et des traitements du domaine, ainsi que des différents modes de représentation utilisés dans les systèmes informatiques. A cette occasion, nous évoquons quelques réalisations qui ont marqué l'histoire des systèmes d'aide au diagnostic médical. Puis vient notre travail d'analyse des éléments impliqués dans la réalisation du système d'aide au diagnostic en rhumatologie inflammatoire.
Cette analyse nous a conduit à définir un ensemble d'objets qui vont constituer le squelette d'un système d'aide au diagnostic. Ces objets modélisent aussi bien des connaissances que des traitements. Les deux points essentiels concernent d'une part la définition d'une description générique qui s'applique au patient et aux maladies, et d'autre part l'organisation du système qui va permettre la cohabitation de deux méthodes de raisonnement partageant la même base de connaissances. La première méthode repose sur l'expertise d'un praticien hospitalier. Cette méthode a donné lieu à l'écriture d'un système de score. La seconde méthode permet de comparer la description d'un patient à des listes de critères diagnostiques issues de la littérature médicale.
Nous terminons ce chapitre avec la présentation du modèle objet du système que nous avons élaboré.

Des éléments pour la conception.

Dans le deuxième chapitre, un aspect particulier de la démarche de conception orientée objet est abordé, faisant appel aux *design patterns*. Ces objets définissent des structures ou des tâches génériques, et facilitent la conception d'applications modulaires. Nous avons identifié quatre *design patterns* décrits dans l'ouvrage de Gamma *et al.* [54] modélisant chacun un des éléments stratégiques du système d'aide au diagnostic. Le pattern

`Composite` encapsule la complexité des signes pathologiques. Le pattern `Etat` sauvegarde les informations sur *la présence* ou *l'absence* des signes indépendamment du traitement mis en œuvre. Le pattern `Itérateur` définit une tâche commune à toutes les procédures diagnostiques et garantit un accès uniforme aux données quelles que soient leur structure ou la méthode de raisonnement activée. Le pattern `Stratégie` définit une famille de procédures de traitement qui s'applique aussi bien dans le cadre du système de score que dans celui de la méthode des listes de critères.

Le Logiciel d'Aide au Diagnostic en Rhumatologie Inflammatoire.

Le modèle objet du système d'aide au diagnostic a été utilisé pour réaliser un logiciel : LADRI (Logiciel d'Aide au Diagnostic en Rhumatologie Inflammatoire). Le troisième chapitre contient les détails de la mise en œuvre de l'application. Il s'agit d'une part du recueil des descriptions de 13 pathologies auprès d'un praticien spécialiste de ce domaine et d'autre part des procédures de calcul attachées aux deux modes de traitement. Nous présentons ensuite un algorithme qui permet l'agrégation des résultats générés par ces procédures, afin de proposer au praticien un ou plusieurs diagnostics possibles. LADRI a été testé sur 107 dossiers de patients soignés au Centre Hospitalier Universitaire de Bordeaux (dans le service du Professeur Dehais) et au Centre Hospitalier de Libourne (dans le service du Docteur Verhnes).

Génie Logiciel et Informatique Médicale.

Le dernier chapitre conclue sur les éléments qui caractérisent un système d'aide au diagnostic, par rapport aux autres systèmes informatiques ou aux autres activités médicales. Il contient sur une réflexion à propos de l'introduction du Génie Logiciel dans des domaines relevant de l'Intelligence Artificielle.

Chapitre 1

Analyse des données et des traitements

> *Je crois que la question originale "Les machines peuvent-elles penser ? " a trop peu de sens pour mériter une discussion. Néanmoins, je crois qu'à la fin du siècle l'usage, les mots et l'éducation de l'opinion générale auront tant changé que l'on pourra parler de machines pensantes sans s'attendre à être contredit.*
>
> A. Turing, *Computer Machinery and Intelligence (1950).*

Ce chapitre est consacré aux problèmes soulevés par la modélisation des systèmes d'aide au diagnostic et à l'explication du modèle objet que nous avons conçu pour une application en rhumatologie inflammatoire. Pour cette application, nous avons choisi d'intégrer au système deux méthodes de raisonnement ayant chacune leur propre traitement mais partageant la même base de connaissances. Notre analyse avait donc pour objectif de décrire une architecture globale du système qui supporte la cohabitation de plusieurs méthodes de traitement, et une structuration des données qui « *autorise* » un tel partage.

Ce chapitre débute par la présentation du contexte médical dans lequel a été développée notre application et des différents éléments que nous avons pris en compte pour cette

application.
Dans la deuxième section, nous présentons quelques étapes majeures de l'histoire de la représentation des connaissances en Intelligence Artificielle. Le but de cet exposé est de montrer les solutions apportées aux problèmes soulevés par la prise en compte simultanée de la représentation des connaissances et de la modélisation du raisonnement au sein d'un même système. Ces solutions passent par la logique formelle, les systèmes de production, les réseaux sémantiques et l'approche objet. Nous présentons également quelques systèmes d'aide au diagnostic médical qui ont marqué l'histoire de l'informatique médicale.
Dans la troisième section, nous expliquons comment nous avons mené l'analyse de notre domaine à l'aide des méthodes orientées objet. Cette analyse aboutit à un modèle objet du système d'aide au diagnostic.
Le chapitre se termine par l'explication des différents modules qui composent le modèle résultant de cette phase d'analyse. Un modèle est toujours une approximation du réel (c'est son intérêt) et le choix des éléments retenus pour cette approximation dépend du problème posé [80]. Pour répondre à notre problématique, nous avons donc défini le modèle d'un système qui sache structurer une description, de patient ou de maladie, pour la rendre accessible à des méthodes de traitements hétérogènes.

1.1 Le domaine d'application

La « *clinique* » est le point de départ obligé du processus diagnostique [62]. La quantité d'informations à gérer, la difficulté à formaliser les données issues d'un examen clinique et le nombre de pathologies à prendre en compte pour modéliser le raisonnement du praticien sont autant de problèmes à résoudre lorsqu'il s'agit de développer un système informatique d'aide au diagnostic [77]. L'élément primordial en médecine clinique est la consultation, elle est le point de départ du recueil des données concernant le patient. C'est ce point de vue que nous avons voulu restituer. Le système se place autant que possible (autant qu'il est possible à un programme informatique de le faire) dans la situation du médecin qui prend en charge un nouveau patient et évalue son état.

La rhumatologie étudie les pathologies touchant l'appareil locomoteur. Elle associe des pathologies dites mécaniques et des pathologies de type inflammatoire où les manifestations articulaires sont souvent associées à des atteintes plus générales (cœur, poumon, ...). De façon réciproque, la plupart des pathologies « dites systémiques » présentent des manifestations touchant l'appareil locomoteur. C'est cette branche de la rhumatologie (la rhumatologie inflammatoire) qui nous a intéressés car il s'agit d'une activité clinique difficile à modéliser. Nous avons sélectionné un groupe de 13 maladies les plus fréquemment observées dans un service de rhumatologie afin d'étudier la mise en œuvre d'un système d'aide au diagnostic de ces pathologies.

1.1.1 Les descriptions de maladies

Pour obtenir les descriptions des pathologies, nous avons utilisé principalement deux sources d'information : l'expertise d'un praticien hospitalier et la littérature du domaine.

Le rhumatologue nous a fourni 13 descriptions qui peuvent être placées sur un pied d'égalité ; c.-à-d. que les signes présentés par son patient, seront évalués de façon comparable pour chaque pathologie dont ils pourraient être une manifestation. Ces descriptions ont été utilisées comme point de départ pour la structuration du système.

Dans un second temps, des définitions académiques ont été ajoutées. Cela a permis de vérifier que le modèle issu de l'expertise permettait d'intégrer des données issues d'une autre source d'information.
Ces définitions sont publiées dans les revues scientifiques ou dans le cadre de recommandations des associations professionnelles comme l'ACR (American College of Rheumatology) ou la SFR (Société Française de Rhumatologie). Ces informations prennent généralement la forme de listes de critères de classification qui permettent de diagnostiquer ou de rejeter une pathologie donnée. Ces critères diagnostiques sont issus des résultats d'enquêtes épidémiologiques. Ils sont assortis de valeur de probabilités résultant des calculs statistiques réalisés au cours de ces enquêtes.

Toutes les pathologies ne font pas l'objet de telles publications et de plus il n'existe pas en rhumatologie inflammatoire de « *gold standards* », le raisonnement diagnostique s'effectue donc toujours dans un contexte d'incertitude.

Il existe d'autres sources d'informations telles que la présentation de cas cliniques types dans les ouvrages d'apprentissage (par exemple les annales de l'internat) ou encore les résultats d'autres enquêtes épidémiologiques communiqués dans les congrès ou publiés dans les revues scientifiques. Ces éléments ont été consultés lors d'une phase préliminaire de prise de contact avec le domaine et ont servi à nous familiariser avec le vocabulaire du domaine. Ils n'ont pas été par la suite intégrés au système.
Les listes de critères reprennent les résultats des enquêtes lorsque ceux-ci ont été largement acceptés par la communauté scientifique. Les études de cas cliniques type relèvent plutôt du domaine du « raisonnement par cas ». Nous n'avons pas utilisé celui-ci pour concevoir le système car cette méthodologie n'a pas vocation à servir de modèle de structuration. Par définition, les études de cas privilégient l'aspect spécifique de chaque exemple. Bien que non structurantes pour notre domaine, ces études n'en demeurent pas moins informatives et pourraient faire l'objet d'un nouveau module de traitement.

1.1.2 La description du patient

L'essentiel des informations recueillies pendant la consultation a pour origine l'interrogatoire du patient et l'observation de son état par le médecin (examen clinique), auxquels s'ajoutent d'éventuels résultats d'analyses biologiques ou examens complémentaires. A chaque consultation, ces informations viennent enrichir le dossier du patient.

Il est donc intéressant dans le cadre d'un système d'aide au diagnostic de générer une description du patient correspondant au modèle adopté pour la description des pathologies. Cette description du patient peut-être obtenue par saisie directe du médecin lors de la consultation ou bien à partir de l'analyse du dossier du patient.

1.1.3 Le système d'aide au diagnostic

Chaque type de définition des pathologies (expertise ou liste de critères), est associé à un traitement qui permet de l'utiliser au mieux : prise en compte de toutes les pathologies (expertise) ou recherche de la validation d'un tableau spécifique (liste de critères). Un praticien qui dispose de ces deux types d'information est capable de les manipuler tour à tour, ou encore d'en favoriser l'un plutôt que l'autre suivant que le patient présentera peu ou prou de signes très symptomatiques d'une pathologie. De même, un système d'aide au diagnostic doit pouvoir prendre en compte des données de types différents et adapter son comportement en fonction des informations recueillies. Il doit intégrer et utiliser des informations hétérogènes puis inférer des méthodes différentes suivant le type de ces informations. Ce caractère hétérogène permet de construire des systèmes plus efficaces et plus compétents [37].
Mais l'objectif n'est pas de faire simplement coexister différents modules de traitements les uns à coté des autres. Ces modules doivent collaborer et partager la même base de connaissances [16]. Cette volonté conduit à rechercher la définition d'un concept de diagnostic suffisamment générique pour répondre aux exigences de plusieurs méthodes de raisonnement.

1.2 La représentation des connaissances

Nous venons de voir qu'un système d'aide au diagnostic doit prendre en compte des éléments divers : les connaissances sur les pathologies, celles du médecin ou plus généralement de la médecine, et les informations relatives au patient. La représentation des connaissances dans les systèmes informatiques fait l'objet de nombreux débats. Un système informatique contient-il des informations ou des connaissances ? Ce qui différencie une information d'une connaissance est la capacité à agir, à adapter sa réponse à une situation ponctuelle, situation qui ne serait pas obligatoirement (et totalement) définie par avance, c'est-à-dire au moment de la construction du système [80]. Un livre ne *connaît* pas les informations qu'il contient puisqu'il ne sait pas les utiliser et bien que ces informations puissent faire l'objet d'un traitement formel, elles ne permettent pas d'accéder au *sens* qui est seul accessible à l'esprit humain [9]. Toutefois, le passage de l'état passif des informations à l'état actif des connaissances peut s'effectuer par palier. L'information passive type est la base de données bibliographique, l'utilisation de cette base par un moteur de recherche est une première activation de ces informations. Des activations plus élaborées se retrouvent dans la déduction de nouveaux faits ou la découverte de stratégie de résolution optimale, comme dans les programmes de jeux, la conception de robots ....

Dans le monde médical, il serait très certainement malvenu de prétendre réaliser un système informatique qui génère de nouvelles définitions de pathologies ou découvre chez un patient des symptômes non encore identifiés. Les médecins rejetteront toujours les systèmes qui afficheraient de telles prétentions et les patients potentiels que nous sommes s'en féliciteront. Mais il existe une place entre un système d'interrogation de bases de données bibliographiques et un générateur automatique de diagnostic. L'objectif est de réaliser des systèmes qui puissent intégrer différents types d'informations, les restituer en fonction d'une situation ponctuelle, faire des suggestions à l'utilisateur ... ce que P. Levine et J. Ch. Pomerol ont appelé les Systèmes Interactifs d'Aide à la Décision [90].

Nous pouvons identifier deux tâches essentielles dans la conception d'un système d'aide à la décision. D'une part celle qui consiste à modéliser les données afin de les intégrer au système ; cette tâche comprend l'étape du choix des connaissances que l'on va représenter. D'autre part, la conception d'un ou plusieurs modules de manipulation des données.

Ces deux tâches ont occupé au fil de l'histoire de l'Intelligence Artificielle des places très différentes. Nous allons parcourir les différentes étapes de cette histoire en montrant à chaque fois quels aspects ont été privilégiés. Des éléments généraux sur les concepts abordés peuvent être trouvés dans [130], [131] et [64] des références plus détaillées sont précisées en fonction des thèmes présentés.

1.2.1 L'automatisation du raisonnement grâce à la logique mathématique

Le premier problème auquel se sont intéressés les pionniers de l'Intelligence Artificielle fut l'automatisation du raisonnement. Ils disposaient pour cela d'un outil, la logique formelle, issue des travaux de mathématiciens du début du XX^e siècle[1], Gottlob Frege [52] et Bertrand Russel [115]. Grâce à la logique formelle, il est possible de démontrer des théorèmes généraux en appliquant simplement des règles typographiques à des ensembles de symboles préalablement définis. Au milieu des années 50, A. Newell et H. Simon ont intégré cette logique dans un des premiers programmes d'Intelligence Artificielle, *Logic Theorist* un démonstrateur de théorèmes, parfois considéré comme le premier car il fût le premier à être réellement implémenté sur une machine [36].

La logique formelle est bien adaptée à la représentation des concepts mathématiques. Toutefois, son utilisation suppose que l'on dispose d'une connaissance complète et stable sur le problème à modéliser. Cette contrainte est rarement satisfaite lorsqu'il s'agit de modéliser les connaissances du monde vivant. D'autre part dans le cas d'une application complexe de grande taille, le formalisme utilisé est impossible à appréhender par l'esprit humain [80]. Enfin, l'efficacité avec laquelle la déduction logique peut-être simulée est rarement satisfaisante [64].

> *Les premiers travaux étaient axés sur des tâches requérant peu de savoir spécialisé. Ce n'était pas que les gens ne mesurait pas l'importance des connaissances mais les machines existant à l'époque limitaient de fait la taille des bases de données qu'ils pouvaient construire [102].*

De nombreux travaux basés sur la logique formelle ont continué et continuent toujours à faire progresser la recherche sur le raisonnement en informatique, elle est par exemple à la base du lambda-calcul ou des systèmes de réécriture. Mais dès que les contraintes matérielles se sont assouplies, les chercheurs ont désiré intégrer plus largement les connaissances dans les programmes informatiques afin d'aborder d'autres classes de problèmes,

[1] Ils reprenaient en cela les premières études d'Aristote sur la forme des inférences qu'utilisent les mathématiciens pour faire leurs démonstrations.

plus proches des problèmes qui concernent les êtres humains.

1.2.2 Un premier pas vers la prise en compte des connaissances : les systèmes de production

La traduction des connaissances en règles empiriques ou heuristiques est un processus ancien. Un papyrus égyptien datant du XVII^{e}siècle avant JC présente un protocole formel de diagnostic et de traitement de 48 cas de blessures à la tête. Ce protocole est rédigé sous la forme, symptômes - diagnostic - traitement. A des règles comme : « ***si** vous examinez un homme présentant ces symptômes, **alors** vous pourrez dire de lui, il s'agit de telle blessure* », sont associés le pronostic et le traitement recommandé [25].

Est-ce un hasard si cet ancêtre des systèmes experts concerne la médecine ? Répondre à cette question ne serait que spéculations gratuites mais il est à noter que les problèmes posés par ce domaine ont très vite intéressés les chercheurs en Intelligence Artificielle. C'est en 1972 que commencèrent au sein de l'équipe de Feigenbaum[2], les travaux d'un étudiant nommé E. Shortliffe sur le premier système expert à base de règles en médecine : MYCIN [124]. Cette application traite le diagnostic des maladies infectieuses du sang.

Extrait de MYCIN [26], règle 006 :

```
If    The culture was taken from a steril source AND
      It is definite that the identity of the organism is one of :
      staphylococcus-coag-neg, corynebacterium-non-diphtheriae,
      bacillus-subtilis.
Then  There is strongly suggestive evidence (.8)
      that the organism is a contaminant.
```

FIG. 1.1: Mycin

[2] Feigenbaum et son équipe sont à l'origine du premier système expert à base de règles : DENDRAL qui cherchait à identifier la structure de molécules organiques complexes [49].

Non seulement Mycin intégrait un ensemble de règles diagnostiques correspondant aux connaissances du médecin mais il contenait également des mécanismes qui permettaient de décider quand et comment appliquer une de ces règles. MYCIN fut le premier système expert affichant une séparation claire entre les règles et le mécanisme d'inférence. Ce moteur d'inférence fut même extrait de MYCIN, sous le nom d'EMYCIN, dans le but d'être utilisé avec d'autres bases de connaissances [136].

✔ Les points forts des systèmes à base de règles

Les concepteurs des systèmes à base de règles ont marqué un profond changement dans les méthodes de développement des systèmes informatisés. Ils ont montré qu'il était possible de s'abstraire d'une partie du « jargon » informatique et de se rapprocher du langage humain pour introduire et mémoriser les données dans le système. La séparation des données (règles et base de faits) du programme informatique est apparue comme une possibilité de ne pas toujours avoir besoin de recourir à un informaticien pour maintenir et mettre à jour le système. Il semblait qu'un non-informaticien - un expert en médecine par exemple - pouvait contrôler et valider la façon dont son savoir était décrit.
De plus, lorsqu'il s'agit de modéliser des connaissances humaines sur la manière de résoudre un problème, la modélisation à partir d'une démarche heuristique est parfois plus aisée à formuler que sous la forme d'une règle de calcul [90]. La capacité des systèmes informatisés à traiter de problèmes réels est largement conditionnée par leur capacité à communiquer avec ceux qui l'alimente de leur savoir ou l'utilise.

Pendant deux décennies, les systèmes à base de règles ont servi de modèle à un grand nombre de réalisations ; pour ne citer que des exemples en médecine : RENOIR [69] et AI-RHEUM [11] deux systèmes d'aide au diagnostic en rhumatologie, SPHINX pour le diagnostic du diabète [23], ONCOCIN [123] qui s'intéressait aux protocoles de chimiothérapie, SENEX [113] pour le diagnostic en cancérologie, INTERNIST [95] pour la médecine interne, AntibES [85] pour la prescription des antibiotiques ...
Quelques 20 ans plus tard, P. Le Beux qualifiait MYCIN (avec une pointe d'ironie) de

« *système le plus cité bien qu'il n'ait jamais été utilisé en routine* »[84]. Ce système a fait référence parce qu'il était le premier système expert en médecine mais aussi parce qu'avec MYCIN, Shortliffe a apporté la preuve qu'un système de production peut être décrit séparément des connaissances qu'il contient. A partir de ce principe, vont se développer un grand nombre de « *coquilles* » (*shells*) dont la vocation était de servir de générateurs de systèmes experts et ainsi de décharger les concepteurs de l'écriture des moteurs d'inférence. C'est le cas d'EXPERT [141] qui fut utilisé entre autre dans AI/RHEUM ou de MILORD le moteur de RENOIR. L'ouvrage « Readings in Medical Artificial Intelligence » [31] recense un grand nombre de systèmes experts en médecine élaborés dans les années 70-80. L'intérêt pour ces outils a décru au début des années 90 lorsque les systèmes à base de règles ont montré leurs propres limites. Mais cette recherche dans l'écriture d'applications modulaires deviendra un des thèmes majeurs d'une discipline informatique qui émergera au même moment : le Génie Logiciel.

✔ Les limitations des systèmes à base de règles

L'extension de la capacité de stockage des machines a favorisé l'émergence de grandes bases de règles. Des problèmes sont alors apparus lorsqu'il s'est agit de modifier, d'ajouter, de retirer des règles ou des faits. Pour faire évoluer le système, la théorie veut qu'il suffise d'ajouter ou de modifier une connaissance [124] [65]. Dans la pratique les règles se trouvent si inextricablement reliées les unes aux autres qu'il devient difficile, sinon impossible, de maîtriser leurs interactions [37].
Quant à la démarche couramment pratiquée dans les années 80, consistant à « extraire » les règles d'un spécialiste et à les placer dans une « *coquille* » de systèmes experts, Clancey [29] a montré qu'elle fournit une vision réductrice des connaissances du spécialiste et de sa façon de résoudre un problème. Dans de nombreux domaines, il est impossible d'obtenir des spécialistes l'ensemble des règles pertinentes à la solution d'un problème donné [98]. D'autre part, la plupart des systèmes ont intégré des méthodes de pondération des faits ou des règles pour gérer le caractère incertain des connaissances. Ces méthodes sont ba-

sées sur les statistiques, les fonctions de croyance, la logique des possibilités ou encore la logique floue. Chacun de ces modes d'évaluation a ses propres contraintes : complétude, exhaustivité des règles pour les statistiques, ordre d'application des règles pour les opérateurs *Min*, *Max* de la logique des possibilités [3]. Tout ajout ou modification de règle suppose la modification (ou au minimum la vérification) des calculs liés à l'évaluation.
Ces problèmes de maintenance semblaient être mineurs au début car les premiers systèmes avaient un durée de vie trop courte pour être concernés. Ils se sont révélés pourtant des handicaps majeurs pour une véritable utilisation en routine des systèmes experts.

Même si les règles codent la connaissance, la modélisation sous forme d'un système de production consiste avant tout à formaliser l'utilisation de cette connaissance. Comme la logique formelle, les systèmes de production permettent d'automatiser certaines formes de raisonnement mais fournissent peu d'éléments sur la manière de modéliser les objets de cette connaissance.

Parallèlement aux recherches sur les systèmes de production, ont commencé à émerger des modèles de représentation sémantique des données se voulant plus proches du modèle cognitif humain. Ces modèles (réseaux sémantiques, objet ...) vont faire l'objet des sections suivantes. Notons déjà que les systèmes à base de règles vont par la suite intégrer des modules de structuration des données directement issus de ces modèles. La gestion de ces modules se fait alors par des méta-règles [106]. Chaque règle appartient à un ou plusieurs ensembles. La sémantique des éléments d'une règle ne s'interprète plus uniquement en terme de *prémisse-conclusion* mais aussi en terme de module d'appartenance. Le mécanisme des règles de production n'est plus alors utilisé que pour gérer une inférence automatique.

[3] On trouvera dans [80] une preuve de la modification du résultat suivant l'ordre de traitement des règles de MYCIN.

1.2.3 L'accent mis sur l'analyse sémantique des données : les réseaux sémantiques

A partir de recherches menées en psychologie cognitive, Quillian [109] puis Brachman [24] et Sowa [126] ont décrit un mode de structuration des connaissances axé sur une représentation sémantique des données. L'objectif n'est plus seulement d'écrire des algorithmes pour traiter les données mais de fournir un modèle qui permette de représenter les objets sur lesquels portent les connaissances. La méthode consiste à identifier les entités du domaine puis à les classifier et les organiser dans un réseau sémantique.

Un réseau sémantique est un graphe étiqueté dans lequel les nœuds représentent des objets ou des concepts et les arcs étiquetés des relations entre ces concepts. Différents types de liens peuvent unir les entités. Ce sont essentiellement les liens « *sorte de* » ou « *partie de* » suivant qu'il existe un lien d'héritage ou de composition entre les entités. On trouve aussi des liens plus axés sur la description du domaine, par exemple la forme, la couleur, l'appartenance etc. Ainsi, un réseau définit une organisation hiérarchique des données (le graphe est orienté) ou taxinomie de concepts. Dans le cas d'un lien d'héritage, les propriétés qui caractérisent un nœud sont automatiquement héritées par ses descendants. L'héritage est un moyen économique de raisonnement permettant notamment à un concept particulier de posséder automatiquement les propriétés des concepts plus généraux dont il est issu [64].

La figure 1.2 montre une partie du réseau sémantique d'un système expert en rhumatologie [108] basé sur la définition des termes médicaux du système HELP [82]. Le dossier du patient possède 4 composants essentiels (liens « `partie de` ») : les signes cliniques, les antécédents du patient, le diagnostic retenu et les actions thérapeutiques prescrites. Des liens d'héritage (liens « `sorte de` ») unissent les différentes classes génériques de médicaments et les molécules spécifiques auxquelles elles appartiennent.

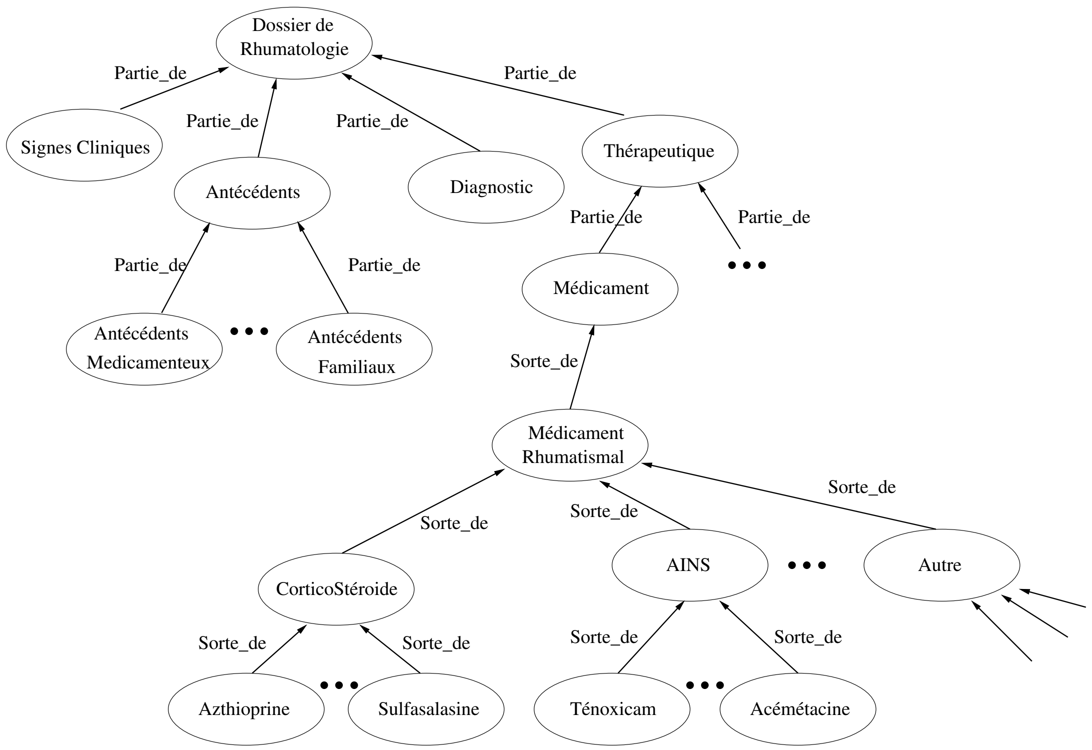

FIG. 1.2: structure d'un réseau sémantique pour le dossier patient en rhumatologie.

En médecine, de tels réseaux sont souvent utilisés pour organiser des thésaurus. UMLS (Unified Medical Language System), le projet de la National Library of Medicine (Bethesda, USA) intègrent les thésaurus les plus utilisés comme le MeSH (Medical Subject Headings), SNOMED ou COSTAR en rattachant chaque terme de ces thésaurus à une catégorie de concepts dans un réseau sémantique [51].

✔ Limitations des réseaux sémantiques

La représentation graphique des données est facile à visualiser et à modifier par l'expert et le concepteur (l'informaticien ou le cogniticien), mais le résultat est difficile à valider avec des procédure formelles [80]. D'autre part, des informations telles que la définition de comportements spécifiques à un type d'objet, ne sont pas représentables dans un réseau sémantique.

En ce qui concerne la phase d'analyse du domaine étudié, cette méthode ne fournit aucune

aide à l'identification des objets qui doivent être modélisés ; le choix des entités qui seront effectivement représentées par des objets dans le système informatique reste totalement empirique.
Enfin, nous verrons par la suite que l'organisation des données dans un programme ne correspond pas toujours à l'organisation mise en place par un esprit humain. La gestion informatisée des données peut nécessiter la définition d'objets spécifiques, qui ne seront pas identifiés par une analyse sémantique.

Les réseaux sémantiques ont été une des premières tentatives de faire entrer dans un ordinateur une certaine « *compréhension* » des données. Cette organisation à partir du typage des objets et d'une structure hiérarchique a conduit à une nouvelle génération de programmes d'Intelligence Artificielle. Cette voie sera poursuivie et développée au travers de la plupart des techniques actuelles de modélisation.

1.2.4 La modélisation du raisonnement en médecine

Nous allons maintenant parler plus particulièrement des systèmes d'aide au diagnostic médical.
Dans ce domaine, la modélisation des connaissances a souvent été appréhendée soit par des systèmes statistiques, soit par des heuristiques pour tenter de reproduire le raisonnement du praticien. Nous allons en premier lieu examiner les particularités des systèmes statistiques puis nous développerons celles des systèmes utilisant une démarche heuristique.

✔ Les systèmes statistiques

Grâce aux outils informatiques, des calculs de plus en plus complexes basés sur des modèles probabilistes ont pu être utilisés pour découvrir les liens qui unissent un ensemble des signes pathologiques à une maladie.
Les modèles probabilistes permettent de quantifier l'incertitude et de représenter les re-

lations dynamiques qui existent entre des suites d'événements aléatoires. Il existe deux grandes familles de théories probabilistes. La théorie dite classique et la théorie Bayésienne.
Soit un événement X, la probabilité $P(X)$ de la survenue de cet événement est définie par une loi (Bernoulli, Normale ... suivant la nature de l'événement) qui détermine la distribution de $P(X)$. Cette distribution dépend d'un vecteur de paramètres θ qui est généralement inconnu.

- Dans la théorie classique, on considère que ces paramètres possèdent une valeur fixe qu'il sera possible de découvrir grâce à des tests et à la recherche d'estimateurs qui s'approcheront autant que possible de la valeur de ces paramètres. Les estimateurs peuvent être par exemple, de simples moyennes entre le nombre d'événements réalisés et le nombre d'expériences tentées.
- Dans la théorie Bayésienne la valeur des paramètres est modélisée par des variables aléatoires avec une distribution de probabilités qui représente la connaissance ou les incertitudes posées sur les paramètres. On admet une connaissance a priori qui va permettre de construire un estimateur de cette fonction de distribution et donner une valeur a posteriori de la distribution de probabilité.

Utilisation du Théorème de Bayes pour le diagnostic médical [117] :
Soit D un diagnostic et S un signe, on peut écrire :

$$P(D/S) = P(D).\frac{P(S/D)}{P(S)}$$

- $P(D/S)$ est la probabilité que D soit présent si le signe S est constaté chez le malade ; on l'appelle probabilité a posteriori de D.
- $P(D)$ est la probabilité de D si l'on a aucune information. C'est donc la fréquence de la maladie D dans l'ensemble de la population (ou encore le taux de prévalence de D) ; on l'appelle *probabilité a priori* de D.
- $\frac{P(S/D)}{P(S)}$ est la fréquence du signe S dans la maladie D relative à sa fréquence dans l'ensemble de la population (ayant ou non la maladie D). On l'appelle *valeur diagnostique* de S pour D (qui sera d'autant meilleure que S sera fréquent dans D par rapport à sa présence dans les autres maladies).

FIG. 1.3: théorème de Bayes

✔ Les problèmes de l'utilisation des probabilités conditionnelles.

Si on a k maladies et p symptômes booléens et pertinents, on a 2^{p+k} situations possibles, $2^k - 1$ probabilités a priori et $(2^p - 1)2^k$ probabilités conditionnelles.
Le modèle peut être simplifié si les maladies sont exclusives : k-1 probabilités a priori et $(2^p - 1)k$ probabilités conditionnelles.
Si l'on a indépendance des symptômes, on peut écrire $P(S_2/D_i \ et \ S_1) = P(S_2/D_i)$, le nombre de probabilités conditionnelles à estimer devient $p * k$.

Ledley et Lusted [87] ont défini le diagnostic médical à partir de trois étapes :

1. déterminer l'ensemble des diagnostics possibles,
2. calculer et ordonner ces diagnostics en fonction des observations réalisées sur le patient,
3. choisir les actions les plus adaptées pour guérir (ou soulager) le patient.

Le première étape est obtenue en recherchant à partir d'un ensemble de signes S, tous les diagnostics D possibles.
La deuxième utilise le théorème de Bayes. La probabilité $P(D/S)$ désigne alors l'état des connaissances connaissances médicales que l'on possède sur la maladie D. Cette connaissance à un caractère stable et universel pour tous les patients alors que la probabilité $P(D)$ est dépendante du contexte géographique, des influences saisonnières, etc.
Le choix des actions se fait par la maximisation de la fonction d'utilité : minimisation des risques, maximisation des préférences du patient etc. De nombreux systèmes (le premier d'entre eux fut celui de De Dombal [39]) ont été réalisés à partir de ce modèle ou de ses extensions comme les réseaux bayésiens ou les réseaux de croyance [88] [12] [96] [78].

Si cette définition du diagnostic en trois étapes reste un modèle tout à fait valide à l'heure actuelle [50], les limitations du modèle tiennent surtout à l'hypothèse qui est faite que l'ensemble des observations est obtenu en une seule étape. Cette limitation a provoqué un rejet de l'approche statistique de la part des chercheurs en Intelligence Artificielle qui se sont tournés vers une modélisation symbolique des connaissances à manipuler.

✔ La modélisation d'une démarche heuristique

Clancey [30] propose de résoudre le problème du diagnostic médical en utilisant une méthode de classification heuristique. Ainsi, établir un diagnostic c'est réussir à associer un ensemble de données (les observations réalisées sur le patient) à un ensemble de solutions (les diagnostics ou les actions à mener). Pour réussir cette adéquation, on définit au préalable une ou plusieurs classes abstraites de données, l'espace des problèmes à traiter, et une ou plusieurs classes de solutions à ces problèmes.

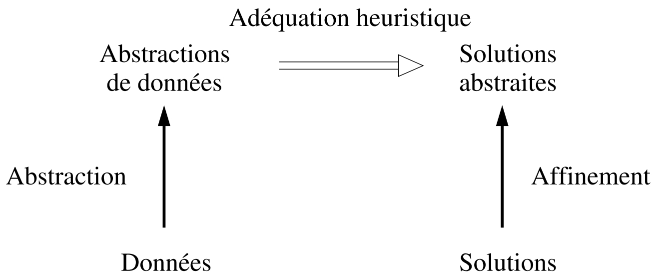

FIG. 1.4: raisonnement heuristique

La figure 1.4 montre comment on obtient la solution adéquate à un problème donné par affinement de l'espace des solutions. La figure 1.5 montre une inférence extraite de Mycin. L'observation réalisée sur un patient (un taux de *globules blancs* inférieur à $2500/mm^3$) permet par étapes successives d'affecter ce patient à la classe *Sujet à risque d'une infection Gram négative* pour proposer finalement une *infection E. Coli.*

Le système expert Internist [107] est un des exemples de système utilisant ce type de démarche heuristique appliquée au diagnostic de plusieurs pathologies. Sa base de connaissances contient 3500 signes permettant de diagnostiquer 500 maladies différentes. Reggia [111], Reiter [112], et De Kleer et Williams [40] ont modélisé les bases de connaissances

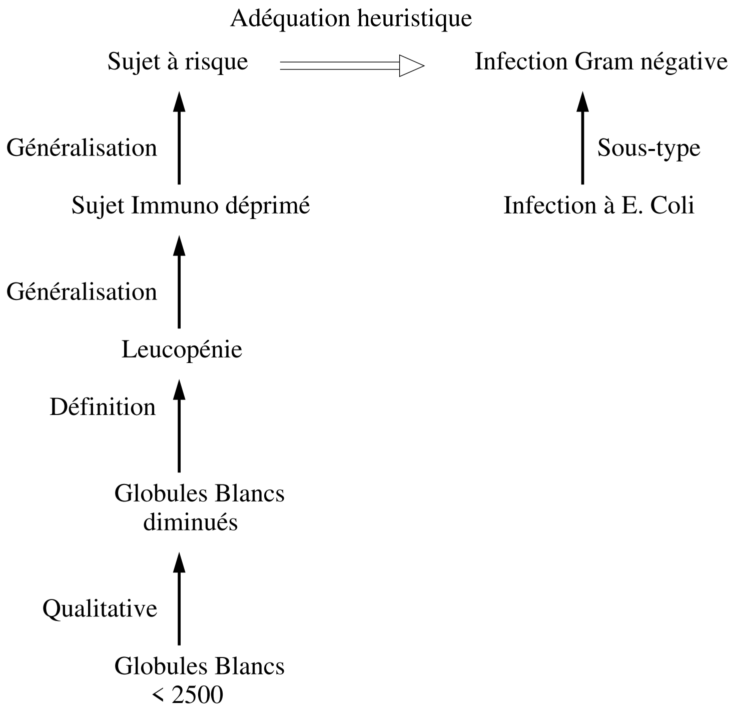

FIG. 1.5: application du raisonnement heuristique dans Mycin

à partir de graphes bi-parties de maladies (exclusives les unes par rapport aux autres) auxquels sont reliés les signes (également exclusifs) qui peuvent en être les causes.

Le résultat de ces différents travaux a été de constater que le diagnostic médical n'est pas une tâche atomique mais plutôt le résultat d'une combinaison de procédures qui sont modélisables soit par un calcul mathématique, l'évaluation des différents diagnostics possibles, soit par la construction d'un raisonnement symbolique. Szolovits et Pauker [128] ont montré comment l'association du raisonnement probabiliste et de la structuration des connaissances par classification des signes et des maladies pouvaient apporter une solution au problème du diagnostic médical.

Toutefois, l'ensemble de ces méthodes ne fournit que peu d'aide dans le choix des connaissances à modéliser et dans le choix des structures les mieux adapter pour les formaliser. P. Degoulet, M. Fieschi et G. Chatellier [42] ont proposé de prendre la structuration des connaissances comme point de départ de la conception d'un système d'aide à la décision. Les méthodes *objet* peuvent être utilisées pour concevoir l'architecture de tels systèmes [53].

1.2.5 Intégration des connaissances et des traitements dans la même structure : l'approche objet

Le terme « objet » a émergé presque simultanément et indépendamment dans divers domaines de la science des ordinateurs au début des années 70, pour référencer des notions qui furent inventées pour contrôler la complexité des logiciels. Les objets représentaient des composants d'un système ou des unités modulaires de représentation des connaissances [20]. Un des points essentiels de la théorie objet est la recherche d'un modèle abstrait des données du système. Shankar [122] a fait ressortir l'importance de l'abstraction de données pour maîtriser la complexité :

> « *la nature des abstractions qui peuvent être réalisées par l'utilisation des procédures est bien adaptée à la description des opérations abstraites mais ne convient pas à la description des objets abstraits. Ceci est un inconvénient sérieux car dans de nombreuses applications la complexité des données à manipuler contribue grandement à la complexité générale du problème* ».

Les approches traditionnelles en informatique procèdent à un découpage algorithmique (les procédures) basé sur les tâches à accomplir par le système. Les méthodes objet se placent sur un autre plan et consistent à identifier les objets du domaine et leurs comportements [66].

✔ Les concepts fondateurs de la notion d'objet

Le premier concept est celui de l'**abstraction**. Cette mise en avant de l'abstraction lie la théorie objet à la théorie des types abstraits de Hoare [70] qui définit un type de donnée abstrait comme un modèle (une structure), actionné par plusieurs opérations. Un objet, comme un type abstrait, est défini en fonction de ce qu'il fait et non par la manière dont il le fait. Définir un objet consiste à mettre l'accent sur les caractéristiques essentielles d'une chose, selon le point de vue d'un observateur. Cela conduit à la définition de modèle commun à plusieurs objets, la **classe**. On parle donc de classe d'objets, les éléments de la classe étant appelés les instances. Le terme d'objet, employé dans son sens le plus général, désigne à la fois les classes et les instances.

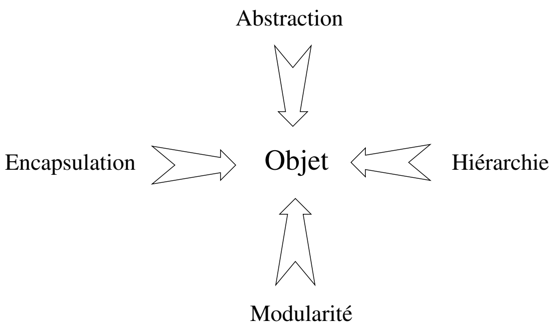

FIG. 1.6: les concepts de la notion d'objet.

Des liens **hiérarchiques** unissent les objets qui partagent un ensemble de propriétés, qu'il s'agisse de valeurs (les attributs) ou de comportements (les méthodes). L'**abstraction** met l'accent sur le comportement observable d'une chose et cache l'implémentation qui donne accès à ce comportement. On parlera d'**encapsulation** pour signifier qu'aucune partie d'un système complexe ne doit dépendre des détails internes d'une autre partie [75] ; l'encapsulation se traduit le plus souvent par un masquage de l'information. Enfin, le dernier concept « *fort* » est la **modularité**, c.-à-d. l'acte de partitionner un système en composants individuels dans le but de réduire sensiblement sa complexité [100] mais aussi de produire des composants réutilisables.

1.2.6 Les systèmes dits de seconde génération

Nous allons dans un premier temps faire quelques rappels sur les développements de la « philosophie » objet en Intelligence Artificielle. Nous ne discuterons pas le fait de savoir à qui revient la paternité des méthodes objet, mais nous nous contenterons de constater qu'elles ont pris leur essor parallèlement en Intelligence Artificielle et en ingénierie logicielle, aboutissant à des résultats sensiblement différents dans ces deux domaines.

Le recentrage amorcé au début des années 80 vers le niveau cognitif provoque l'éclosion de méthodes fortement imprégnées de sciences humaines comme KOD, basée sur des travaux de linguistique et d'anthropologie de C. Vogel [139]. Les concepteurs de systèmes se concentrent sur l'analyse des tâches cognitives, l'explicitation de modèles des domaines traités et l'identification des méthodes de résolution mises en œuvre. A. Newell [101] parle de niveau cognitif en opposition au niveau symbolique. Ses travaux portent sur la définition des connaissances à un niveau abstrait indépendant de la représentation symbolique informatique. Cette abstraction des connaissances a donné lieu à ce que L. Steel a appelé les « *systèmes experts de seconde génération* » [127]. Dans ces systèmes, le modèle abstrait des connaissances, **Knowledge Level** est associé à un **Computed Level** caractérisé par la recherche d'abstractions pour décrire les processus de résolution [38].
La définition des connaissances au **Knowledge Level** se fait suivant trois principes essentiels [98] : *role limiting*(1), *knowledge typing*(2) et *reusability*(3).
(1) Un rôle spécifique est affecté à chaque entrée de la base de connaissances et limite l'utilisation de cette donnée à la tâche pour laquelle elle est définie.
(2) Le système de typage des données est l'outil nécessaire à l'application du premier principe. Il s'agit de définir des structures permettant d'identifier clairement le rôle de chaque élément dans la résolution du problème. Les éléments de la base de connaissances sont définis comme étant des tâches, des méthodes de résolution de problème ou encore des ontologies du domaine.
(3) Enfin, la définition d'abstractions doit, ou devrait, permettre l'identification de composants génériques des systèmes d'aide à la décision. La réutilisation de ces composants présente le double avantage de « normaliser » les applications grâce au partage d'éléments communs et de faciliter leur évolution par ajout de modules déjà écrits.

Nous retrouvons donc les concepts objets énoncés dans la section précédente (illustrés par la figure 1.6), le mécanisme de typage des données (2) notamment dans la construction de hiérarchies d'objets, l'encapsulation et la modularité (1) assurent un rôle spécifique aux objets, enfin la réutilisabilité (3) est une conséquence de la mise en œuvre des concepts d'abstraction et de modularité.

Une des méthodes d'analyse objet la plus répandue en Intelligence Artificielle est KADS [121]. Un modèle d'expertise KADS comprend 3 niveaux : le *Domain Layer* contient l'ensemble des concepts de base du domaine, l'*Inference Layer* est celui des inférences qui définissent les relations entre les concepts du domaine et le *Task Layer* rassemble les tâches qui managent ces inférences. Cette méthode a servi de cadre à la modélisation de nombreux systèmes à base de connaissances, et de fait, fut une approche standard pour la construction de systèmes d'aide à la décision [135], on en trouvera des exemples en médecine dans [116] [91] [120]. Dans les années 90, les auteurs de la méthode en ont proposé une version revisée appelée CommonKADS [119]. Cette version intègre la notion de bibliothèques de composants réutilisables comme les ontologies (nous présenterons les ontologies à la fin de cette section).

Depuis le début des années 90, deux approches objet se développent et trouvent un écho important dans le domaine des systèmes à base de connaissances médicales : l'une axée sur l'identification de tâches génériques, l'autre sur la définition d'ontologies.

✔ Les tâches génériques

Chandrasekaran et son équipe [27] ont développé des bibliothèques de **tâches génériques** telles que : *select and test*, *temporal abstraction* ou encore *cover and differanciate* dont le but est de construire la meilleure explication d'un ensemble d'événements ou d'états.
Bien que les principes de modularité et d'encapsulation soient respectés, la définition d'un système en terme de tâches reste proche du découpage algorithmique d'un problème. Le

raisonnement est assimilé à un ensemble de tâches à accomplir (la figure 1.7 montre une tâche de diagnostic). Chaque tâche manipule ses propres données suivant une organisation et un typage qui lui est propre. Cette façon de développer des modèles de raisonnement a conduit Chandrasekaran à affirmer qu'à chaque tâche correspond une structuration des données qui lui est propre et que vouloir partager un modèle de représentation des données entre plusieurs méthodes de raisonnement est illusoire [28]. Pour Chandrasekaran, ce ne sont donc uniquement que les tâches qui sont réutilisables.

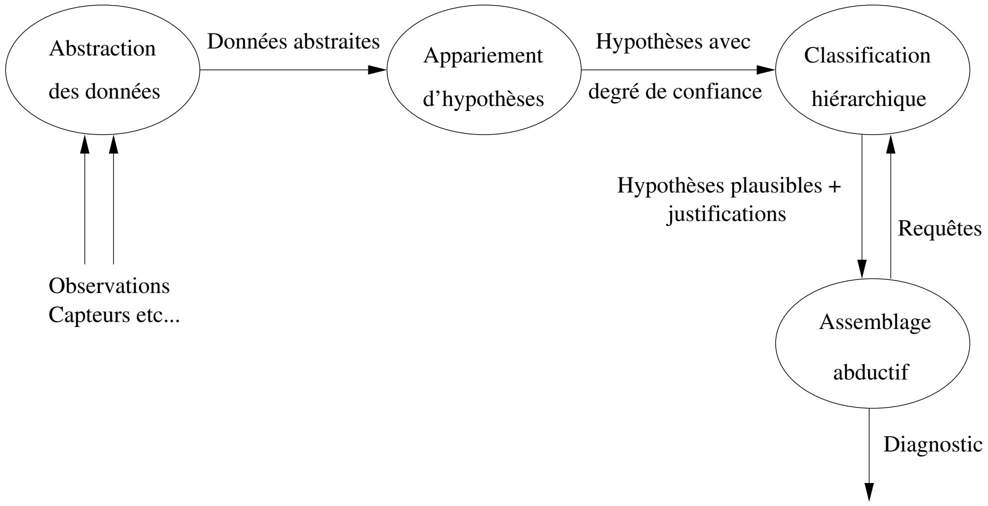

FIG. 1.7: modélisation d'une tâche de diagnostic.

✔ Les ontologies

D'autres équipes comme celle de M.A. Musen [132] ou de M. Stephanelli [47] [48] se sont dirigées vers la description des systèmes à partir d'**ontologies**[4]. L'objectif est de construire des bibliothèques d'ontologies qui peuvent être partagées entre plusieurs applications, ceci afin d'améliorer la capacité de communication de ces applications entre elles et d'alléger les charges de développement. Il s'agit de définir des modèles abstraits des connaissances tels que *domain ontology* pour la description abstraite de concepts dans un domaine

[4] L'ontologie est la science de l'être : ce qui est et comment cela advient [59].

d'application, ou encore *problem ontology* pour la description de modules de raisonnement ou de traitements spécifiques. Les ontologies peuvent être regroupées en modules, appelés *théories*. Ce concept est très proche de celui des objets, toutefois il s'en distingue par une analyse strictement *top-down*. Cela a pour conséquence de favoriser une organisation hiérarchique des concepts au détriment de l'analyse transversale qui permet l'identification de structures communes à plusieurs modules indépendants.

Notre approche se situe à mi-chemin de ces deux démarches, recueillir et modéliser les connaissances pour ce qu'elles sont avant de se soucier de savoir ce que le système en fait, mais les formaliser et les structurer sans perdre de vue qu'elles seront utilisées par un système d'aide au diagnostic informatisé. C'est dans ce but que nous nous sommes tournés vers des méthodes développées dans le cadre objet mais ayant leurs propres spécificités.

✔ La prise en compte de la dimension logicielle : l'orienté objet

Le développement de l'informatique a eu pour conséquence la construction de logiciels toujours plus gros et toujours plus nombreux. Aux besoins de gestion de la complexité inhérente à la taille d'un programme s'est ajoutée la prise de conscience du coût de maintenance des logiciels. Le coût réel d'une application informatique n'est pas seulement dû au coût de création, mais doit également intégrer le coût induit par la durée de vie de ce logiciel (modifications, adaptations, mises à jour). Il est apparu comme indispensable, d'une part de définir des normes qui puissent canaliser les développements, d'autre part de trouver des moyens de ne pas toujours reprendre à zéro l'écriture des applications (« from scratch » pour les anglophones). Les objets constituaient une réponse à ces attentes mais pour des raisons sensiblement différentes de celles évoquées par les chercheurs en Intelligence Artificielle.
En effet, les ingénieurs logiciels ont cherché à réutiliser des « bouts de code » plutôt qu'à générer des « coquilles » génériques longues et complexes à concevoir. Ils vont commencer par identifier les petits morceaux de programme qui reviennent toujours dans les applications, pour les factoriser et ainsi limiter les réécritures fastidieuses. Pour cela, ils vont

s'approprier les principes objet en les appliquant d'une façon plus dirigée vers l'ingénierie logicielle et moins vers la théorie des systèmes. Ils inventent une nouvelle expression : *orienté objet*, qui va s'appliquer en premier aux langages de programmation puis aux méthodes de conception et enfin à celles d'analyse.

Historiquement, les langages de programmation orientés objet ont donc été conçus avant que les méthodes d'analyse et de conception ne soient totalement formalisées, d'où les critiques de manque de formalisme ou de faiblesse théorique souvent énoncées à l'encontre de ces méthodes. Bien que G. Booch, un des pères des méthodes orientées objet, proclame que

> « *le développement par objets est un processus conceptuel indépendant des langages de programmation, ce type de développement est fondamentalement une nouvelle manière de penser et non pas une technique de programmation* » [20],

ces méthodes conservent l'image d'une approche empirique, fortement liées à l'implémentation du code. Toutefois un de leurs atouts majeurs reste une prise en compte globale du problème de la réalisation de logiciels avec des méthodes d'analyse, de conception et des langages de programmation qui respectent tous les mêmes paradigmes.

Orienté Objet signifie que l'on organise le logiciel comme une collection d'objets dissociés comprenant chacun une structure et un comportement. Les objets peuvent être concret (un fichier) ou conceptuel (un système d'aide à la décision). De nombreuses méthodes orientées objet ont été développées. Plutôt que d'en faire une énumération fastidieuse pour le lecteur, nous le laisserons consulter l'ouvrage de I. Graham [59] ([60] en version française) pour une présentation détaillée des plus représentatives.

Après une période d'abondance de méthodes[5], en 1995 deux des figures les plus marquantes du domaine, J. Rumbaugh (père de OMT [114]) et G. Booch ([20]) ont entamé une démarche d'unification de leurs méthodes. Ils ont ensuite été rejoints en 1996 par I. Jacobson (concepteur de OOSE [76]). Leur travaux ont conduit à la définition de UML :

[5] Au début des années 90, on en comptait plus d'une cinquantaine.

Unified Modeling Language [133] [22] [6]. C'est le langage de modélisation objet que nous avons utilisé pour décrire le modèle final du système.

Nous verrons que des méthodes comme UML permettent de formaliser le développement orienté objet d'un système complexe avec un niveau d'abstraction suffisant pour ne pas être attaché à un langage de programmation spécifique, même s'il est souhaitable que celui-ci soit un langage objet.

1.3 Les étapes de l'analyse orientée objet du domaine

La première étape de l'analyse est une étape de **désagrégation**. Avec les méthodes orientées objet, on fait l'hypothèse que le système ou l'application forme un tout. Pour détailler ou exprimer la solution informatique, on postule que ce tout est composé de parties cohérentes séparables [83]. Cette décomposition du problème à traiter suivant ses entités essentielles est fondamentalement différente de l'approche classique de la décomposition algorithmique (représentative de l'approche fonctionnelle) [21].
Pour analyser un système d'aide au diagnostic, nous n'abordons donc pas la décomposition à partir des tâches à accomplir : observer le patient, calculer une probabilité de maladie . . . , mais à partir des éléments dont nous disposons : description des maladies, dossier du patient . . .
La seconde étape consiste à reconstruire le système après l'avoir désassemblé. Le regroupement en classe, des modules générés pendant l'étape précédente, se fait par la détection d'attributs ou de comportements communs entre ces modules. Dans notre application, nous cherchons à constituer des groupes de signes partageant les mêmes caractéristiques (touchant les mêmes parties anatomiques, évalués par les mêmes méthodes, . . .).

[6] Cette méthode a fait depuis l'objet de nombreuses collaborations avec d'autres concepteurs ou firmes tel que IBM, DEC, Oracle, Rational Software . . . La version 1.0 a été déposé à l'Object Management Group, organisme international de normalisation, en 1997. La version 1.3 est en cours de validation auprès de cet organisme.

Pourtant plutôt que de parler de démarche descendante (*top-down*) ou de démarche ascendante (*bottom-up*), comme pour la définition d'ontologies, on préfère parler de démarche de droite à gauche ou de gauche à droite [83] suivant l'action réalisée :

$$\texttt{un module} \xRightarrow{\texttt{désagréger}} \texttt{une suite de modules}$$
$$\texttt{un module} \xLeftarrow{\texttt{agréger}} \texttt{une suite de modules}$$

Le système est appréhendé « morceau par morceau ». A chaque étape nous devons considérer les compositions, ou classifications, qui sont créées sous l'angle des différents modes de définition des pathologies puis analyser si ces « assemblages » conviennent pour décrire un patient, si les procédures de raisonnement vont pouvoir utiliser les structures choisies ... On décompose, on recompose et ce processus est réitéré jusqu'à l'établissement d'un satisfecit. C'est-à-dire jusqu'à l'obtention des entités nécessaires à chaque module pour remplir son rôle.

Dans notre contexte, l'objectif de ce processus est de dégager des concepts génériques qui seront communs à plusieurs pathologies. C'est à partir de ces concepts génériques qu'il est ensuite possible de construire la description du patient. Chaque mot retenu est regroupé avec d'autres mots relevant du même concept. L'architecture du système émerge en fonction des liens que l'on définit : liens hiérarchiques, liens de composition, ou encore pas de lien si les objets communiquent entre eux par envoi de messages. C'est également à cette étape qu'est défini le rôle (le comportement) de chaque objet et donc les limites du domaine prises en compte par le système.
Les paragraphes suivants montrent comment cette analyse a été conduite sur les descriptions et les raisonnements.

1.3.1 Analyser une description

L'identification et le choix des objets à représenter, est une tâche importante dans la conception d'un système. G. Booch [19] propose d'initier l'analyse par l'étude des flots de données. Pour cela, il utilise une méthode inspirée de celle de R.J. Abbott portant sur l'analyse textuelle [1]. Partant d'un texte, elle consiste essentiellement à associer les noms aux objets et les verbes aux méthodes. La figure 1.8 montre un exemple d'analyse d'une courte description médicale.

« *Mr Bertrand est un patient qui a une douleur lombaire. Il devra être radiographié avant d'envisager un traitement* » .

Type des élt. du discours	Composant du modèle	Exemple
nom propre	instance	Mr Bertrand
nom commun	classe	douleur, patient, traitement
verbe « faire »	opération	radiographier, envisager
verbe « être »	classification	est un
verbe « avoir »	composition	a une
adjectif	valeur d'attribut ou classe	lombaire

FIG. 1.8: analyse textuelle

Ces techniques bien que non-formelles, peuvent être utilisées avec succès dans des domaines où il existe une certaine simplicité sémantique des données à manipuler ou bien lorsque l'analyse se focalise sur les aspects dynamiques du système : contrôle de commande, flots de données, changement d'états ... Par exemple, cette méthode pourrait être employée efficacement pour modéliser l'activité d'un service hospitalier ou une succession d'examens médicaux à faire suivre à un patient. Par contre, elle fournit une analyse peu performante lorsqu'il s'agit d'analyser des descriptions et non des activités, ce qui est le cas de nos définitions de pathologies (cf. l'exemple suivant).

Quelques éléments de la description du rhumatisme psoriasique

- « *Cette pathologie se caractérise principalement par une atteinte poly ou oligoarticulaire et généralement par la présence d'un psoriasis cutané. On observe parfois des douleurs des sacro-iliaques* »

Ce type de définition contient peu de verbes et une multitude de noms communs désignant des concepts différents. Dans ce cas, il est difficile de faire émerger une organisation des données à partir d'un découpage semblable à celui de la figure 1.8.

✔ Désagréger une description

Les descriptions des 13 pathologies de la rhumatologie inflammatoire ont fait l'objet d'une simple décomposition lexicale. Nous avons relevé les termes (noms propres, noms communs ou adjectifs) qui apparaissent, sans a priori du rôle qui sera le leur dans l'architecture du système.

A partir de la définition précédente, on obtient le résultat ci-dessous. Les mots soulignés correspondent aux termes relevés dans cette description.

- « *Cette pathologie se caractérise principalement par une <u>atteinte</u> <u>poly</u>(<u>articulaire</u>) ou <u>oligoarticulaire</u> et généralement par la <u>présence</u> d'un <u>psoriasis</u> (manifestant une <u>atteinte</u>) <u>cutané(e)</u>. On observe parfois des <u>douleurs</u>(<u>articulaires</u>) des <u>sacro-iliaques</u>.* »

Ce processus est conduit sur toutes les descriptions qui font partie de la base de connaissances. Certains termes font alors l'objet d'une description plus détaillée par le praticien. Par exemple, le terme *atteinte articulaire* recouvre des notions comme la *douleur*, la *destruction* ou l'*inflammation* d'une articulation.

✔ Classification et agrégation

Le choix et le nombre de classes à retenir est un problème difficile. Parsons [105] pense que l'utilisation du modèle cognitif humain est un bon guide pour ce choix. Il permet d'équilibrer la tendance à maximiser la connaissance portant sur les instances en les classifiant et le besoin de minimiser la connaissance des détails hors de propos dans la plupart des cas, retenir un nombre de classes trop grand fait perdre l'avantage de la classification. Nous allons regarder sur un exemple comment cette classification est construite à partir des éléments fournis par la phase de désagrégation.

Considérons le mot `douleur` et les différents autres mots auxquels il est associé dans une ou plusieurs définitions. Une `douleur` peut être générale, c.-à-d. diffuse, sans localisation particulière. Elle peut être `musculaire` ou `articulaire`, et porter sur une `localisation` précise. On peut lui associer des caractéristiques liées à sa `fréquence` (douleur en crise ou en continue) ...

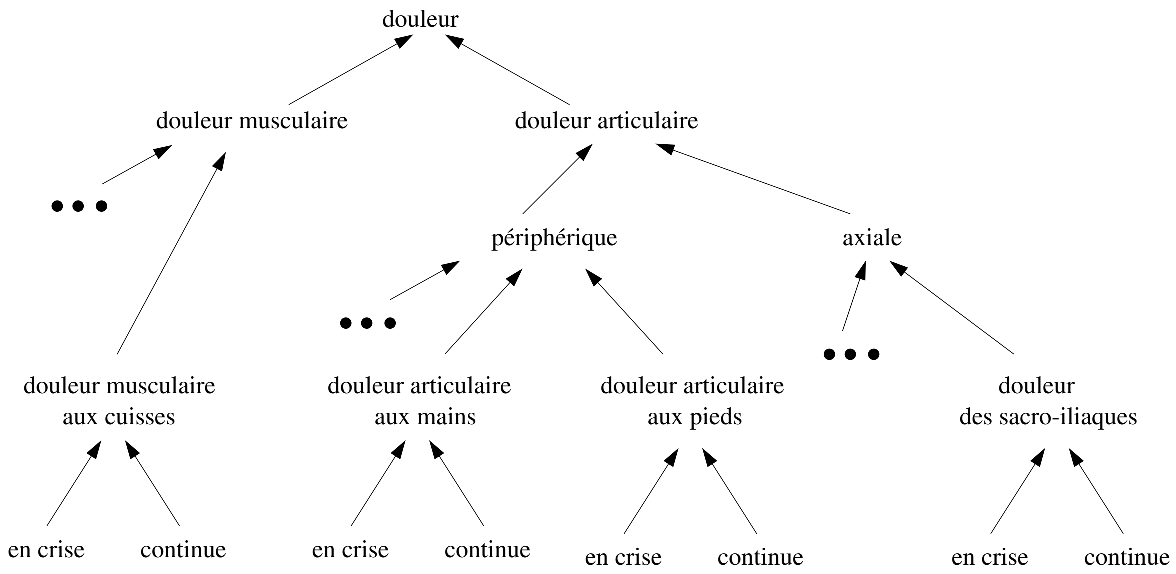

FIG. 1.9: mauvaise organisation des concepts liée à une trop forte hiérarchisation.

Plusieurs stratégies d'organisation de ce vocabulaire sont possibles comme s'appuyer sur un système d'héritage « fort », depuis le concept de `douleur` jusqu'à la spécification terminale de `douleur articulaire` des `sacro-iliaques en crise` (figure1.9). Cette organisation conduit à la définition de structures hiérarchiques de grande taille et surtout

risquant d'être de profondeur importante. La maintenance et l'accès à de telles arborescences pose toujours des problèmes d'implémentation. De plus les performances du système s'en trouvent fortement pénalisées, non pas au niveau du temps de calcul mais en terme de facilité d'accès pour l'interface utilisateur. D'autre part, le partage de caractéristiques comme la `localisation` ne sera que très difficilement réalisable. Ainsi, alors que les localisations des douleurs et des destructions musculaires ont de fortes chances d'être les mêmes, ce partage ne sera que difficilement pris en compte par cette architecture. On peut bien sûr décrire des graphes de dépendances entre les objets (au lieu d'arbres) mais cela introduit de nouveaux problèmes de cohérence, de gestion des cycles ...
Un tel mode d'organisation conduit à des erreurs d'organisation comme celles décrites par C. Vogel [140] : aberrations paradigmatiques ou binaires. L'aberration paradigmatique concerne un sur-développement de l'arbre en largeur, comme l'introduction d'une distinction tronc/membres dans la douleur musculaire pour faire le pendant à la distinction axial/périphérique de la douleur articulaire. L'aberration binaire concerne un sur-développement de l'arbre en profondeur, comme dans la figure 1.9 où une partie de la hiérarchie du mot `douleur` doit être remplacée par la valuation d'attributs de `localisation`.

Pour déterminer les concepts à privilégier, nous avons repris le point de vue de l'observateur, c.-à-d. du médecin examinant un patient. Celui-ci cherche à retrouver dans son observation du patient, la survenue d'événements comme une *douleur articulaire* ou une *douleur musculaire* ... Il s'agit là d'un concept de base, représenté par un objet `Evénement`.
Le terme « événement » désigne des éléments d'observation, il est utilisé ici sans sa connotation temporelle.
Pour spécialiser un objet `Evénement`, on peut lui associer des caractéristiques, qu'il s'agisse de parties du corps, de fréquence etc.
Dans notre modèle abstrait nous reconstituons ainsi le concept de signe (ou syndrome), familier des systèmes d'aide au diagnostic, à partir de l'objet `Evénement` dont l'un des attributs est une agrégation d'éléments définissant ses caractéristiques. Ces attributs ne

doivent pas être vus comme de simples valeurs possibles d'un champs de l'objet `Evénement` mais comme de véritables objets `Attributs` qui possèdent une structure, des méthodes définissant leur comportement et qui peuvent communiquer avec les autres objets du modèle.

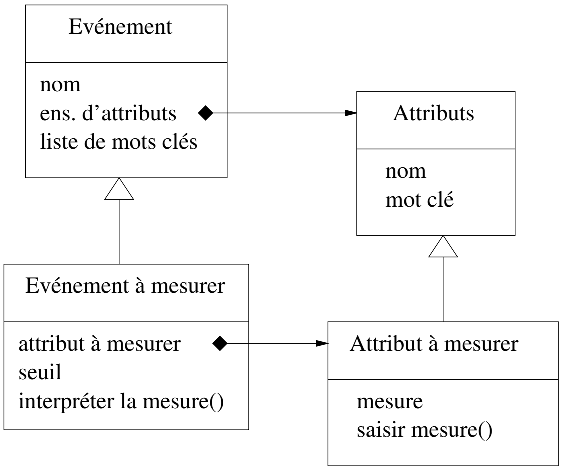

FIG. 1.10: organisation du vocabulaire autour du concept d'`Evénement`

Nous avons retenu deux types d'événements, ceux qui prennent en compte des attributs simples (définis essentiellement par leur nom) et ceux qui prennent en compte des attributs nécessitant une mesure. On a donc une super classe `Evénement` et une classe dérivée `Evénement à mesurer` qui en hérite (cf figure 1.10). Nous avons aussi deux classes pour les attributs, une super classe `Attribut` et une classe `Attribut à mesurer`. Ces derniers possèdent un champs qui permet de mémoriser la valeur de la mesure : l'interprétation de cette mesure est réalisée grâce à un seuil mémorisé dans l'`Evénement`.
Par exemple, l'événement *syndrome inflammatoire biologique* est en fait un `Evénement à mesurer` dont un des `Attributs à mesurer` est la *Vitesse de Sédimentation (VS)* du patient. Cette *VS* existe bien indépendamment de l'interprétation qui en est faite dans l'application. Le seuil de cette mesure ($20mm/h$) qui définit le *syndrome inflammatoire biologique*, n'est pas directement lié à la mesure mais à son interprétation faite par le praticien.
Avec cette organisation (cf figure 1.10), la valeur du seuil peut-être modifiée pour appliquer

un raisonnement différent, ou encore peut-être dynamiquement calculée en fonction d'un contexte associant d'autres éléments (âge du patient, ...).

Ainsi la modélisation des descriptions n'utilise que ces quatre classes (`Evénement`, `Evénement à mesurer`, `Attribut`, `Attribut à mesurer`). Les `Evénements` modélisent les concepts dépendants du domaine d'application ici la rhumatologie inflammatoire. Les `Attributs` représentent des notions médicales plus générales et modélisent le contexte[7] dans lequel apparaissent les `Evénements`.

On pourrait être tenté à l'issu de l'analyse sémantique du vocabulaire, de créer différentes classes d'attributs pour les organiser dans le système : attributs de localisation, attributs de cardinalité, attributs de fréquence etc. Ceci permettrait de contrôler la cohérence des associations entre événements et attributs lors de la saisie des descriptions (maladies ou patients). En effet, tous les attributs ne sont pas « *autorisés* » pour tous les événements. Par exemple, il est incorrect de chercher à associer une douleur articulaire avec une localisation systémique.

Suivant les principes de l'analyse objet, il n'existe pas de différences de classes entre ces notions de localisation, de cardinalité ou de fréquence, car il n'y a pas de différence de comportement et donc pas de méthode supplémentaire entre un attribut de localisation (main, cuisse, ...) et un attribut de fréquence (en crise, continue). Tous ces attributs sont désignés et manipulés uniquement par leur nom. C'est un exemple de la différence qui existe entre la classification humaine et celle employée par l'ordinateur. Le praticien a besoin de connaître et de maîtriser les différences sémantiques du vocabulaire qu'il utilise. Le système n'a en aucune façon besoin de « *comprendre* » ce type de différences et donc de les matérialiser.

[7]Dans le dictionnaire, le contexte se définit comme étant l'ensemble des circonstances qui accompagnent un événement.

Il est tout de même nécessaire de gérer la cohérence des associations `Evénement/Attribut` pendant la saisie des définitions. Pour cela, il suffit de trier les attributs à l'aide de `mot-clés`. La classe `Evénement` possède un champs pour mémoriser les `mot-clés` autorisés. Ces informations sont stockées dans des dictionnaires d'événements et d'attributs, elles sont utilisées dans le programme d'interface pour paramétrer la saisie, afin que le praticien ne dispose que des attributs autorisés pour caractériser ses événements.
La figure 1.11 montre l'organisation du vocabulaire en terme d'événements, de mot-clés et d'attributs.

✔ Avantages de cette structuration

Grâce au mécanisme d'héritage et à l'encapsulation de l'information, tous les événements sont traités par les autres modules de façon uniforme qu'ils requièrent ou non une mesure. Il en est de même pour les attributs. Nous obtenons donc un modèle très simple pour manipuler le contenu des descriptions, en séparant les éléments nécessitant un traitement spécifique (la gestion d'une mesure), des éléments qui ne relèvent que de la classification « sémantique » humaine. Cette simplicité de structure permet pourtant de reconstituer toute la complexité du vocabulaire médical par des emboîtements successifs d'objets. Cette construction est finalement présentée au praticien sous la forme d'une seule entité : le *signe*.

Reprise des éléments de la description du rhumatisme psoriasique

- *douleur articulaire : polyarticulaire ou oligoarticulaire*,
- *atteinte cutanée de psoriasis*,
- *douleur articulaire des sacro-iliaques.*

L'ensemble des descriptions manipulées par le système est donné en annexe A. Nous n'avons pas encore pris en compte la valuation des signes, ce thème est abordé dans le paragraphe suivant à propos de la désagrégation et de l'agrégation des méthodes de raisonnement.

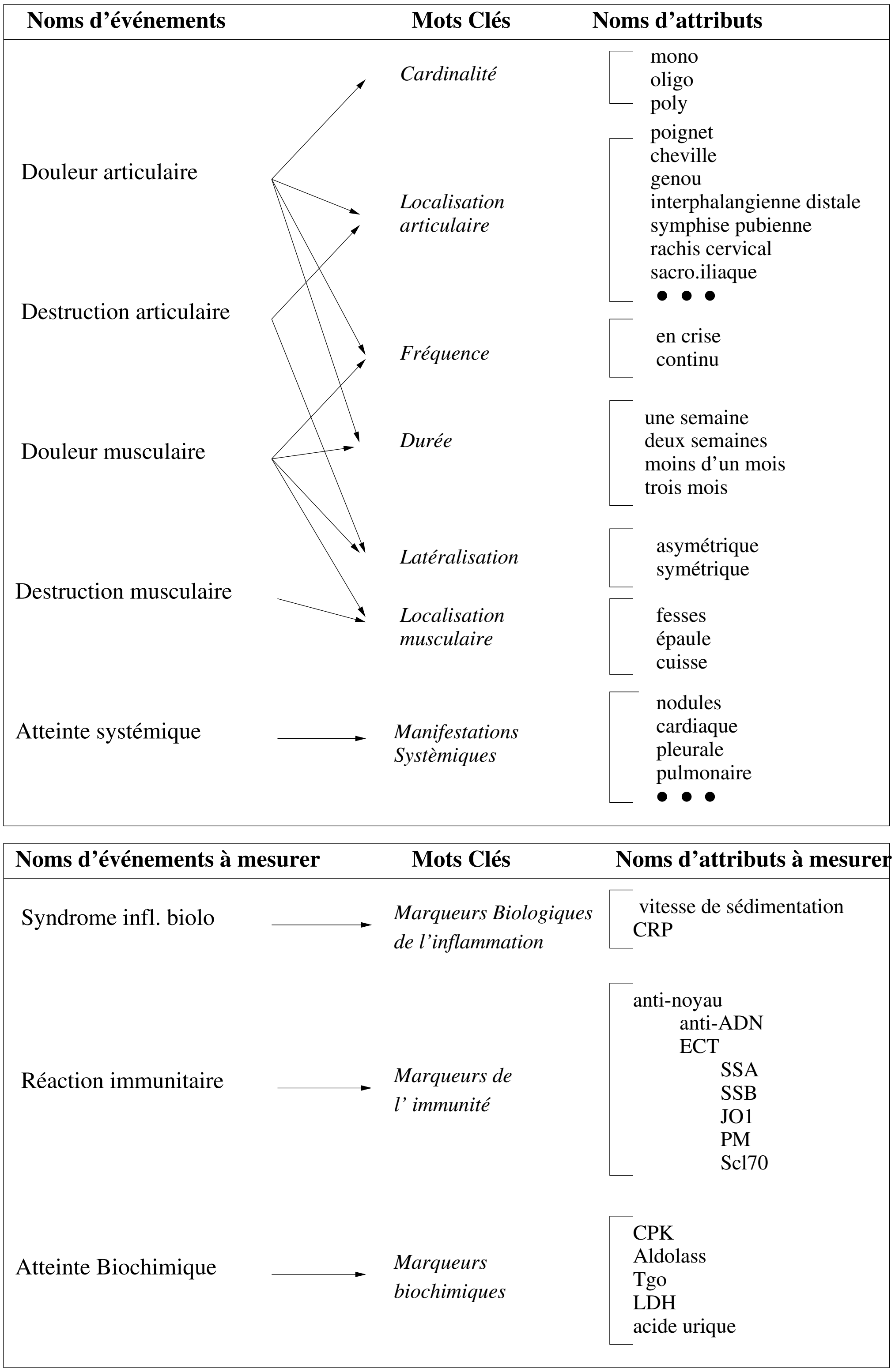

Fig. 1.11: organisation d'une partie du vocabulaire.

1.3.2 Analyser un raisonnement

Toutes les méthodes de diagnostic médical ont en commun une étape liée à l'observation du patient. Il s'agit pour nous de la première étape du processus de raisonnement. Pour traiter les informations issues de cette observation nous avons retenu deux méthodes associées chacune à un type de raisonnement différent. Le premier est un système à base de **score** qui réalise une classification des pathologies. Il s'agit de comparer le patient à toute les maladies contenues dans la base de connaissances, c'est un processus d'**évocation**. Le deuxième utilise des **listes de critères** diagnostiques et compare le patient à chaque pathologie dont la liste est connue. Il s'agit dans ce cas d'une étape de **confirmation** des hypothèses émises au cours de la phase précédente.

Observation → *Evocation* → *Confirmation*

✔ Désagréger un raisonnement

La phase d'**observation** se résume à une saisie de la description du patient. Cette description se réalise à partir du vocabulaire retenu pour la description des maladies. Il s'agit d'une simple tâche de saisie sous contrôle, entièrement gérée par l'interface. Ce n'est donc pas une réelle tâche diagnostique.

La phase d'**évocation** est comparable à un raisonnement de type *classification.* Son objectif est de prendre en compte sur un pied d'égalité toutes les pathologies appartenant à la base de connaissances. Le processus associé est un calcul de score. Pour chaque pathologie, on « *additionne* » les signes observés chez le patient qui participent à la description de cette maladie. Cette méthode nécessite l'attribution d'une valeur numérique à chaque signe décrivant une maladie. Ces valeurs numériques reflètent le sens des termes *parfois*, *souvent*, *généralement*, ... relevés dans le discours de l'expert.
Les différents objets manipulés par cette méthode sont : les descriptions, les valeurs qui

permettent l'interprétation des signes contenus dans ces descriptions, une opération de construction du score, et un résultat (un score par pathologie).

Dans la phase de **confirmation**, la méthode de liste de critères manipule des valeurs symboliques : critères majeurs, critères mineurs, critères d'exclusion. L'opération associée à cette méthode s'apparente à l'activation d'un règle du type : valider au moins 1 critère majeur et 1 mineur.

Ces deux méthodes de raisonnement relèvent de traitements de natures fondamentalement différentes. Ils s'appuient sur des types de valeurs qui leur sont propres (numériques pour le calcul de score, symboliques ou logiques pour les listes de critères) et en font chacun une interprétation différente. Les détails de ces 2 modes opératoires sont donnés au chapitre 3.

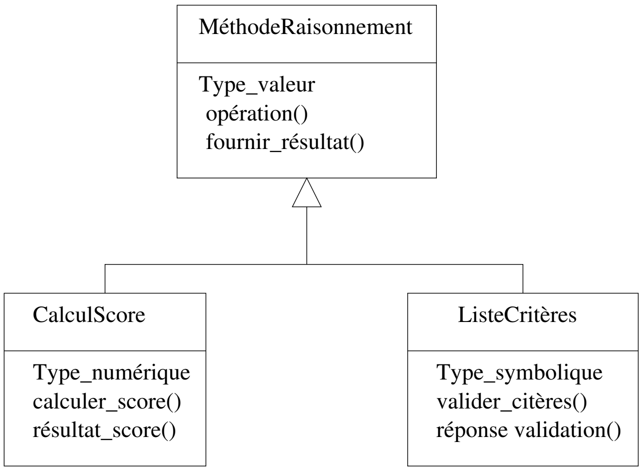

FIG. 1.12: hiérarchie des méthodes de traitement.

✔ **Agrégation : les éléments communs aux deux méthodes**

On peut donc décrire chaque méthode de raisonnement diagnostique à partir de trois concepts : la définition d'un *type de valeur* à associer aux signes pour permettre leur interprétation, la mise en œuvre d'une *opération* qui matérialise la méthode diagnostique, et le *résultat* qui sera transmis au module demandeur.

La démarche d'agrégation aboutit à la définition d'un objet `MéthodeRaisonnement` contenant le champs `Type_valeur` et les méthodes `opération` et `fournir_résultat`.

L'ensemble du processus diagnostique prend en compte plusieurs méthodes de traitement. Il n'est pas dans le rôle de chacune de ces méthodes de gérer leur collaboration. Cette tâche sera réalisée au niveau d'un « *manager* » général du système qui assure également l'accès aux descriptions (ces points sont développés au chapitre suivant). Il n'y a pas de communication directe entre les méthodes de diagnostic, chacune d'elles ignore que d'autres traitements peuvent être demandés. Cela assure la possibilité d'ajouter un nouveau module sans interférer avec les modules déjà présents.

1.4 Résultats de l'analyse

L'analyse menée à partir des principes des « *systèmes experts de seconde génération* » (cf. section 1.2.6) nous conduit à exprimer les connaissances représentant le domaine (Knowledge Level) pour produire les connaissances nécessaires au système (Computed Level) (cf figure 1.13).
Le **Knowledge Level** regroupe les entités `Signe`, `Patient` et `Maladie` pour les données et les deux méthodes de raisonnement `Evocation` et `Confirmation` héritent du même concept de `Diagnostic`. Dans le **Computed Level**, les `Patients` et les `Maladies` sont représentés respectivement par une `DescriptionPatient` et une `DescriptionMaladie` dont les éléments (`Signes`) sont regroupés dans des `Dictionnaires`. La méthode `Score` implémente l'évaluation de toutes les pathologies appartenant à la base de connaissances.

La méthode des `Listes de Critères` permet la confirmation ou non des pathologies arrivées en tête de la méthode du `Score` [15].

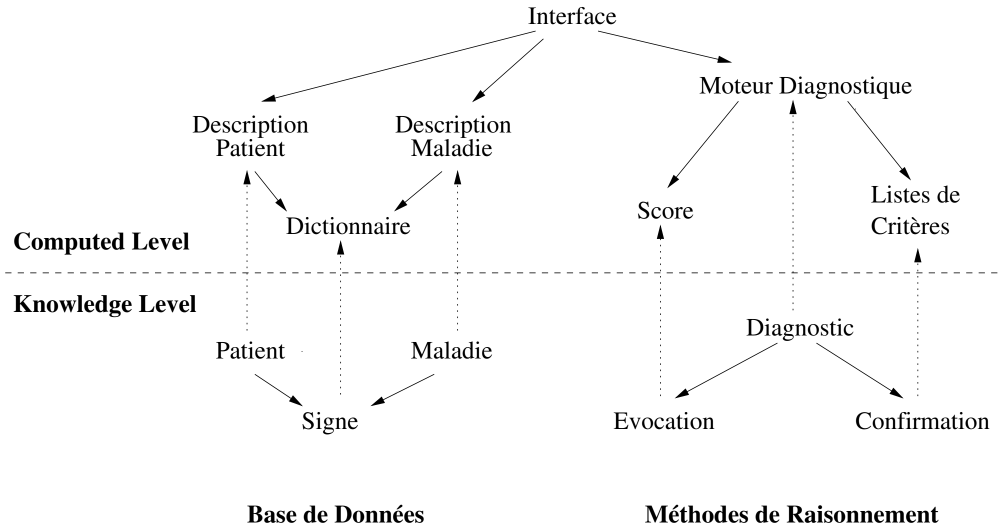

FIG. 1.13: organisation générale du système

La deuxième phase de l'analyse est réalisée au travers du filtre de l'*orientation objet.* Elle fournit les différentes briques qui composent chaque entité : événement, attribut, type de valeur, opération etc. Il reste ensuite à reconstruire l'ensemble du système.
Kayser [80] rappelle qu'il existe toujours plusieurs « *bonnes* » représentations des connaissances. Les objectifs du système permettent de trancher entre telle ou telle représentation. Nos choix sont orientés par le fait que nous faisons du diagnostic. Dans une analyse orientée objet, c'est le point de vue de l'observateur qui est privilégié. Par conséquent, l'organisation finale du système a pour objectif de modéliser la vision du praticien pendant une tâche diagnostique. Le schéma 1.14 décrit cette organisation en langage UML.

De même que le praticien cherche à identifier les plaintes du patient, le modèle du système est centré sur les événements qui surviennent. Le point de départ n'est pas la structure `Maladie` mais l'ensemble des signes qui composera ensuite indifféremment la

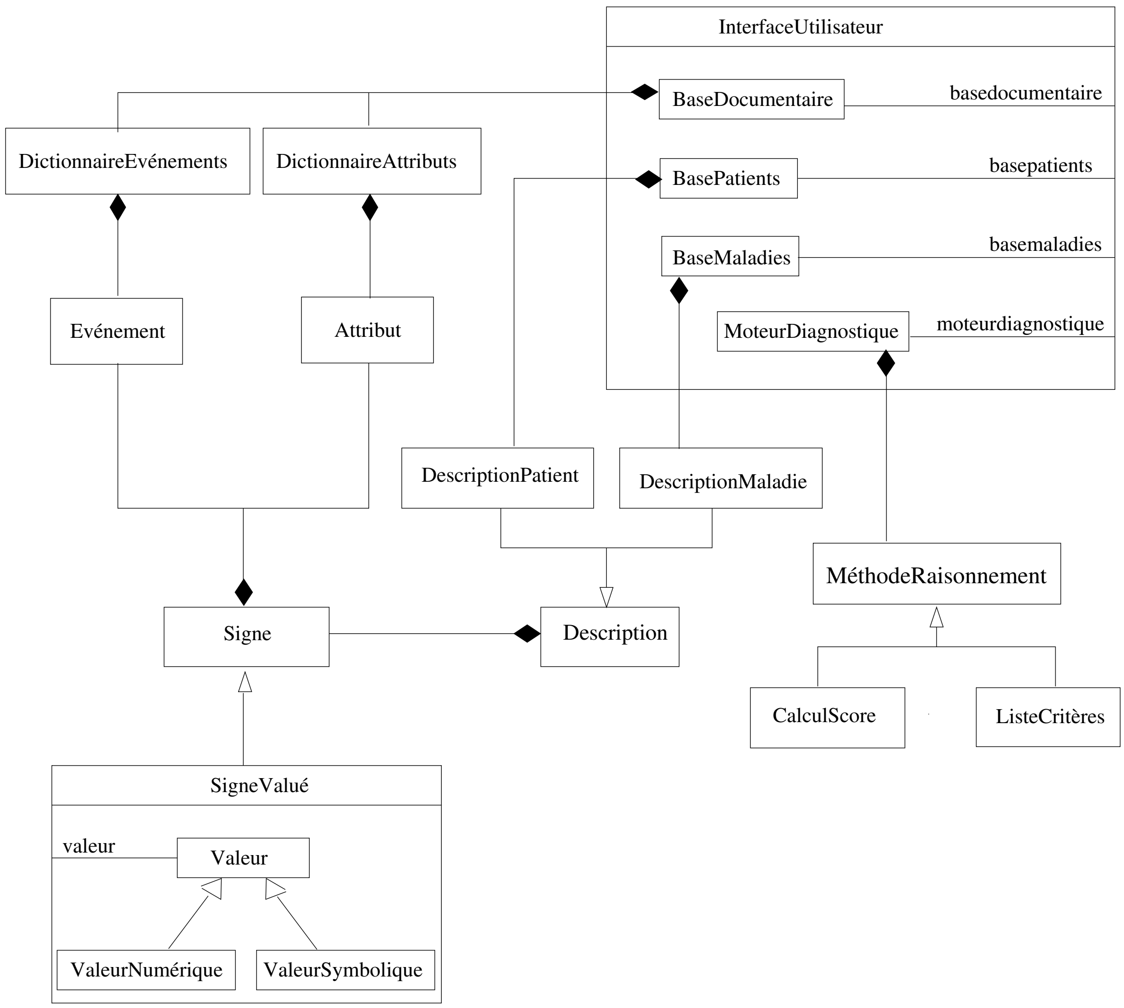

FIG. 1.14: modèle objet du système

description d'un patient ou d'une maladie. Si le système était orienté vers un type de description, patient ou maladie, il serait ensuite nécessaire de « *traduire* » l'autre description pour effectuer la comparaison. Nous avons extrait des définitions de maladies des concepts suffisamment généraux (*douleur articulaire*, *destruction articulaire* ...) pour décrire patient et maladies dans un même moule.

La structure générique `Description` est une agrégation de signes. Le fait de faire hériter la description du patient et de la maladie de cette structure générique garantit que le patient sera représenté dans une forme comparable à celle des pathologies.

Un `Signe` est soit une composition entre un `Evénement` et un `Attribut`, soit un `Evénement` seul. Pour la description des pathologies, on utilise des `Signes Valués`, c.-à-d. des instances d'une classe dérivée de `Signe` auxquels on ajoute une `Valeur`. Cette `Valeur` peut être de type numérique ou symbolique.

Les noms des événements et les noms des attributs associés à leurs mot-clés, sont regroupés dans des `Dictionnaires` qui seront utilisés par l'interface lors de la saisie des patients. Pour décrire un patient on dispose uniquement des noms des événements et des noms d'attributs qui ont été cités au moins une fois dans la description des pathologies car ce sont les seuls pour lesquels le système possède une valeur, c.-à-d. la possibilité de les interpréter.

Puisque les deux méthodes de traitement héritent du même objet générique, l'objet `Interface` dispose d'un accès homogène aux résultats. L'encapsulation des valeurs dans une classe permet aux descriptions des pathologies de ne pas dépendre des méthodes de traitement inférées. L'`Interface` est l'élément qui met en œuvre la communication entre les modules de la base de connaissances et les traitements. C'est elle qui gère les résultats obtenus.

Dans un processus de développement logiciel utilisant les méthodes orientées objet, les étapes d'analyse et de conception sont deux étapes dont les frontières ne sont pas très précises. G. Booch propose de commencer la conception dès que le modèle formel ou informel (éventuellement incomplet) du problème à formuler est obtenu [20]. En commençant la conception trop tôt, l'analyste peut ne pas connaître suffisamment le problème et donc être incapable de faire des compromis d'architecture intelligents. En commençant trop tard, il risque de gâcher des ressources en faisant une analyse trop détaillée, ce qui peut ensevelir le concepteur sous un torrent de détails surabondants et inutiles.
La stratégie préconisée consiste à « analyser un peu » puis « à concevoir un peu », ainsi chaque tentative de conception aide à recentrer le processus d'analyse en le dirigeant sur des aspects du système qui sont importants pour l'élaboration de la solution. Le modèle

que nous venons de voir (cf figure 1.14) s'il n'est pas encore complet, est assez précis pour que nous puissions envisager de passer à l'étape de conception.

Chapitre 2

Conception du système

Un système de raisonnement peut être comparé à un œuf. Le contenu de l'œuf est protégé par sa coquille, et pourtant si vous voulez transporter l'œuf, vous ne ferez pas confiance à cette seule coquille ; vous le mettrez dans un emballage soigneusement choisi.

D. Hofstadter, *Gödel Escher Bach, chap. VII (1985)*

Avec les méthodes orientées objet, les étapes d'analyse et de conception ne sont pas vraiment dissociables. L'analyse est un étape de compréhension du domaine. Mais comprendre le domaine d'un problème implique simultanément d'appréhender la structure, les composants, les modèles de calcul et autres constructions mentales qui prennent en compte le domaine des solutions possibles. Cette activité inventive correspond au processus de conception [60].

Dans le chapitre précédent nous avons identifié les principaux objets issus de l'analyse du domaine. Parler d'objets et de classes est déjà une intrusion dans la conception. Nous allons poursuivre avec l'étude d'objets plus directement liés à la conception logiciel. L'intégration de ces objets au modèle final suscite un retour sur l'analyse. Nous retrouvons ainsi le principe d'aller-retour déjà évoqué lors de l'analyse des descriptions (de patient et de maladie) et des raisonnements.

Dans une description, la séparation claire des événements qui la composent du mode de valuation de ces événements, répond à deux objectifs. Premièrement, cette séparation permet de définir une structure commune au patient et à la maladie en dehors du mode de valuation (rappelons que seuls les signes des descriptions de maladies sont valués). Deuxièmement, l'encapsulation du type de valeur, sépare la structure de description des processus de raisonnement qui seront appliqués. L'étape de conception va devoir enrichir le premier modèle de l'analyse afin d'y ajouter les objets nécessaires à la réalisation informatique et ce en gardant en point de mire les deux objectifs que nous venons de citer. Il faudra également résoudre le problème de la collaboration des méthodes de raisonnement et fournir un mécanisme qui respecte la modularité et l'encapsulation des opérations.

Pour parvenir à ce résultat nous avons utilisé des « *modèles de conception* », les *design patterns*. Après une présentation de ce concept, nous détaillons dans la suite de ce chapitre les patterns que nous avons effectivement utilisés, puis nous revenons sur le modèle du système incluant l'ensemble des objets. Nous concluons ce chapitre avec une discussion sur l'architecture orientée objet pour les systèmes d'aide au diagnostic.

Les différents schémas présentés suivent les conventions généralement appliquées dans les méthodes objet. Le nom des classes débute par une majuscule alors que les méthodes (procédures) commencent par une minuscule. La deuxième partie du nom d'une méthode rappelle celui de la classe. Les noms des instances des classes sont en minuscules. Les conventions graphiques sont celles du langage UML.

2.1 Des outils pour la conception : les patterns

C'est un architecte, Christopher Alexander dans son ouvrage « *Timeless way of building* » [3] qui a développé l'idée de définir un ensemble de *modèles de conception* pour lesquels il a utilisé le nom de *pattern*. La traduction du terme pattern est souvent sujette à polémique. Le titre du livre faisant référence dans la communauté informatique, *Design*

Patterns, est traduit par « Modèles de conception réutilisables », expression qui en reflète bien le sens mais lourde à utiliser tout au long de ce mémoire. Un des mots français souvent retenu est celui de *patrons* mais il ne fait pas l'unanimité à cause des nombreuses interprétations possibles de ce mot dans notre langue. Nous conserverons donc le mot anglais qui identifie parfaitement le concept dans la « communauté objet ». Alexander appelle un *pattern* une entité définie comme suit :

> *Chaque pattern décrit un problème qui se manifeste constamment dans notre environnement, et donc décrit le cœur de la solution de ce problème, d'une façon telle que l'on peut réutiliser cette solution des millions de fois, sans jamais le faire deux fois de la même manière* [4].

Le terme « langage de patterns » ne doit pas être interprété comme celui de langage de programmation. Pour Alexander, un langage de patterns est formé d'un ensemble de patterns dont chacun décrit un problème particulier de la construction d'une maison. Quatre chercheurs en informatique, travaillant dans le domaine de l'objet et surnommé *la bande des quatre* (*the Gang of Four*) : E. Gamma, R. Helm, R. Johnson et J. Vlissides, ont repris ce concept et l'on replacé dans le contexte de l'architecture logicielle.

> *Design patterns capture the static and dynamic structure of solutions that occur repeatedly when producing applications in a particular context* [34].

Leur ouvrage « Design Patterns » [54] ([55]pour la traduction française) est à la fois une présentation des principes de l'utilisation des patterns dans le cadre du développement de logiciels, et un catalogue de 23 patterns dont les caractéristiques et les conditions d'utilisations sont décrites précisément. Depuis, d'autres ouvrages ont été publiés [1] mais ce premier catalogue reste une référence majeure pour les patterns.

Nous allons maintenant développer la présentation de 4 patterns du catalogue de Gamma *et al* et montrer comment chacun d'eux apporte une solution en différents points stratégiques du logiciel. Ils vont fournir une autre vision de l'architecture du système qui sera ensuite superposée avec celle obtenue lors de la précédente étape d'analyse du domaine. Le pattern `Composite` intervient dans la structuration des connaissances, les

[1] on trouvera une bibliographie complète à l'URL : http://hillside.net/patterns

patterns `Itérateur` et `Stratégie` fournissent un modèle abstrait des tâches à réaliser, et le pattern `Etat` modifie la modélisation du système de valuation dans les descriptions. Enfin, nous verrons comment peuvent s'assembler ces différents patterns pour collaborer entre eux ou avec les autres objets du système.

Les catalogues de patterns sont généralement organisés en rubriques : intention, motivation, structure, exemple d'utilisation, conséquences et collaboration. Nous suivons le même schéma mais en nous contentant de rappeler brièvement le principe général du pattern, les détails étant disponibles dans [54], et en nous attardant sur la motivation spécifique à notre contexte et sur la structure appliquée à nos exemples. Les cas d'utilisation se restreignent aux exemples tirés de notre application, notre domaine ne faisant pas partie des exemples d'applications cités couramment dans la littérature sur les patterns. Les conséquences seront développées lorsqu'elles ont un caractère différent de celui généralement présenté pour ces patterns. Les collaborations sont étudiées dans une section particulière.

2.2 Signes et syndromes : `Composite`

✔ Intention

Le pattern `Composite` permet de traiter les agrégations d'objets de la même façon que les objets élémentaires.

✔ Motivation

La conception de logiciels suppose de pouvoir identifier des structures et une organisation commune pour l'ensemble des données manipulées. La structure qui va permettre la représentation informatique d'un signe ou d'un syndrome importe peu aux méthodes de raisonnement. Le modèle d'analyse a défini un signe - ou un syndrome - comme une agrégation d'événements associés ou non à des attributs, pour lequel on dispose d'une valeur qui représente le poids de ce signe - syndrome - dans le diagnostic de la maladie. Le signe est donc manipulé comme une seule entité quelle que soit sa complexité interne.

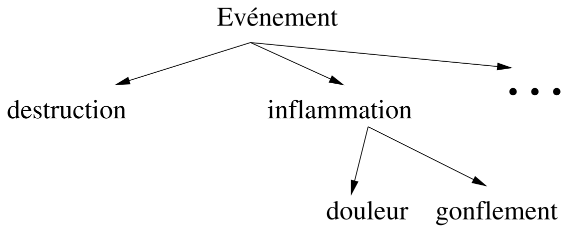

FIG. 2.1: hiérarchie des événements.

Bien que les maladies prises en compte appartiennent toutes à la rhumatologie inflammatoire, la phase de conception du logiciel doit gérer une certaine hétérogénéité des descriptions ou plutôt des signes qui les composent. Cette hétérogénéité peut donc exister à l'intérieur d'une description.

Par exemple, les *douleurs articulaires aux mains* sont définies à partir d'un événement *douleurs articulaires* et d'un attribut de localisation *mains*. Mais une *inflammation des articulations des mains* se compose des deux événements *douleurs articulaires* et *gonflements* pour un même attribut de localisation *mains* (cf figure 2.1). Cette composition pourrait être ignorée du modèle en ce sens que l'on peut fournir un événement *inflammation* à charge pour l'utilisateur de savoir qu'il s'agit d'une composition des événements cités ci-dessus. Mais dans cette configuration, si le praticien saisit séparément les deux événements, le système devrait alors savoir l'associer à *inflammation* et effectuer (ou proposer) le remplacement. Or, ceci n'est pas possible si cette équivalence n'est pas explicitée dans la base de connaissances. Grâce au pattern `Composite`, le système aura tous les moyens de construire des événements complexes et de les manipuler comme s'il s'agissait d'événements simples [14].

✔ Structure

On distingue deux classes d'objets : `Composant` et `Composite`. Le `Composite` est une structure récursive qui se place dans l'arborescence au même niveau que les objets élémentaires (cf figure 2.2). Les objets élémentaires sont les `Feuilles` de la hiérarchie. Tous les objets héritent de la classe abstraite `Composant` qui est l'interface avec les autres modules du système, le mécanisme d'héritage permet alors un accès unique à ces objets quelle que soit leur structure interne.

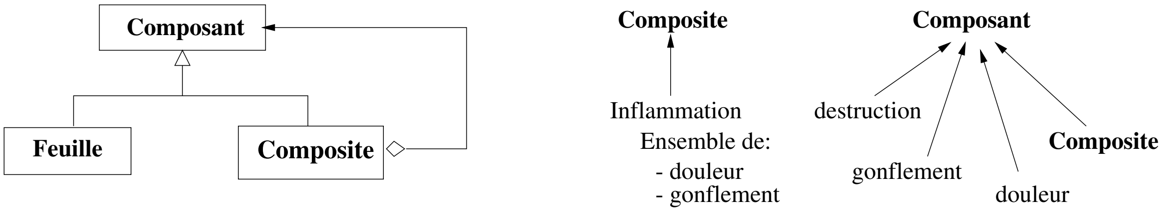

FIG. 2.2: les objets `Composite` et `Composant`.

La figure 2.3 présente le graphe d'héritage pour les instances d'événements. Tous les événements, simples ou mesurés (cf la description des événements page 47), appartiennent à la classe `Composant`. Un événement appartient à la classe `Composite` s'il nécessite une agrégation d'événements comme pour la construction de l'instance l'*inflammation*.

✔ Utilisation

Le *Composant* implémente les traitements liés à la communication avec les clients, et les comportements communs à tous les éléments de la composition. Dans un système d'aide au diagnostic la tâche essentielle est de comparer les signes présentés par le patient à ceux appartenant aux définitions des pathologies. Que le signe soit composé d'événements simples ou complexes, le client (la méthode de raisonnement) désire seulement connaître la réponse à cette question :

- « *ce signe présenté par le patient appartient-il à la description de la maladie ?* »

C'est cette réponse qui va conditionner les différents traitements. Il est donc intéressant

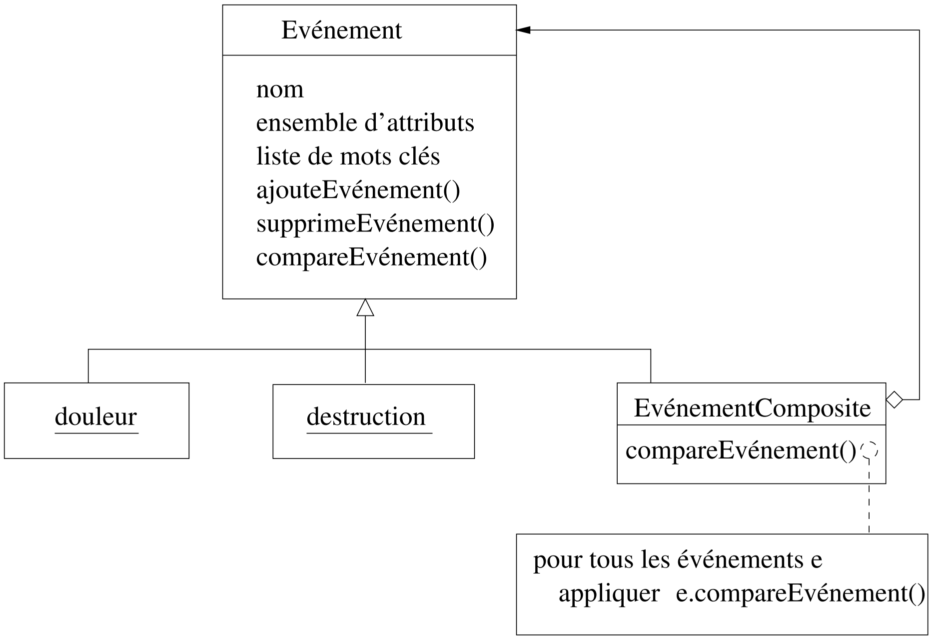

FIG. 2.3: organisation finale des événements.

que les signes montrent aux autres modules une interface unique, la gestion de la complexité éventuelle de ce signe étant intégralement gérée par la structure `Composant` de façon interne.

Cette architecture a été initiée pour la création de composition d'événements, elle peut évidemment être reproduite pour gérer les attributs de ces événements. Lors de l'analyse des listes de critères, il est plus fréquent de trouver des compositions à tous les niveaux. En effet, la volonté de minimiser le nombre de critères spécifiques à retenir, a pour conséquence de construire chaque critère à partir d'un grand nombre de caractéristiques.
Par exemple dans la *maladie de Still*, l'attribut *température* doit non seulement avoir été mesuré à au moins $39°C$ mais également l'avoir été depuis au moins une semaine. L'événement *Altération de l'état général* doit donc être associé à un attribut *température* qui est de type élément à mesurer, et un second attribut *durée* dont la valeur est *une semaine*.

✔ **Conséquences**

Comme l'accès à ces objets est unique, les procédures de recherche des différents signes d'une maladie sont elles aussi uniques et plus faciles à implémenter.
Notons que si ce pattern correspond bien au besoin de la gestion des signes, il ne doit pas être introduit dans tous les agrégats. Ainsi, bien que la base de connaissances contienne un ensemble de maladies, cet ensemble n'est pas un `Composite` car il existe seulement un objet maladie et il n'est pas nécessaire d'introduire une interface entre cet objet et ses clients. La seule motivation pour créer ce type d'arborescence de maladies serait de vouloir créer une maladie ayant plusieurs formes. Des exemples de telles configuration existent, comme le *syndrome de Sharp* (association de signes du *lupus*, de la *sclérodermie* et de la *dermato/polymyiosite*), mais il se trouve que nous n'avons pas retenu de maladies de ce type dans cette version du système.

2.3 Identification d'une tâche générique : `Itérateur`

✔ **Intention**

Le pattern `Itérateur` fournit un accès séquentiel à une agrégation d'objets sans exposer la structure interne de celle-ci. Il est utilisé classiquement pour parcourir le pattern `Composite` et c'est également ce que nous avons fait. Mais par ailleurs, ce pattern nous a aussi permis d'améliorer l'organisation de nos traitements et c'est cet aspect de la modélisation que nous allons développer.
Dans notre contexte, le pattern `Itérateur` modélise une activité toujours présente dans un système d'aide au diagnostic mais généralement non dissociée des différentes méthodes de traitements spécifiques. Cette tâche concerne l'accès à chaque élément appartenant aux descriptions de patients et de maladies. Notre approche consiste à isoler cette tâche dans un objet spécifique.

✔ **Motivation**

L'analyse d'une tâche diagnostique comme dans l'exemple de Chandrasekaran (cf figure 1.7) met en exergue des tâches cognitives humaines. De la même façon, nous avons identifié pour chaque raisonnement étudié, des tâches assez proches de la démarche diagnostique du praticien comme « *comparer le patient à toutes les pathologies de la base* » pour le système de score ou « *tenter de valider une liste de critères spécifiques* » pour la deuxième méthode. Notre analyse orientée objet nous a ensuite conduit à réaliser un découpage de la tâche diagnostique plus proche d'un modèle informatique, en définissant un objet générique `MéthodeRaisonnement`(cf page 53). Mais en descendant à un niveau plus fin d'analyse, on s'aperçoit qu'il existe encore des possibilités de détecter des éléments communs à toutes les méthodes et qui n'ont pas été explicités.

La tâche diagnostique peut être regardée comme étant principalement une **énumération** de tous les signes observés chez le patient afin de les comparer à ceux contenus dans la description des pathologies. Cette tâche est en effet réalisée quelle que soit la structure des définitions et les opérations appliquées dans les modules de raisonnement. Cette partie générique du diagnostic peut être décrite au moyen de l'algorithme suivant :

```
pour tous les signes qui appartiennent à la description du patient,
    si ce signe est le même que celui de la description de la maladie,
    alors prendre en compte la valeur de ce signe.
```

La première amélioration consiste à implémenter un module de gestion de listes, de tableaux, d'arbres ... et d'appeler la fonction d'accès correspondant à la structure de stockage des éléments (accès à partir d'un indice, parcours d'un arbre binaire, d'un arbre AVL). Un niveau supplémentaire d'abstraction est obtenu avec l'introduction d'un objet `Itérateur` qui fournit un accès générique aux éléments, indépendamment de la structure de données.

✔ **Structure**

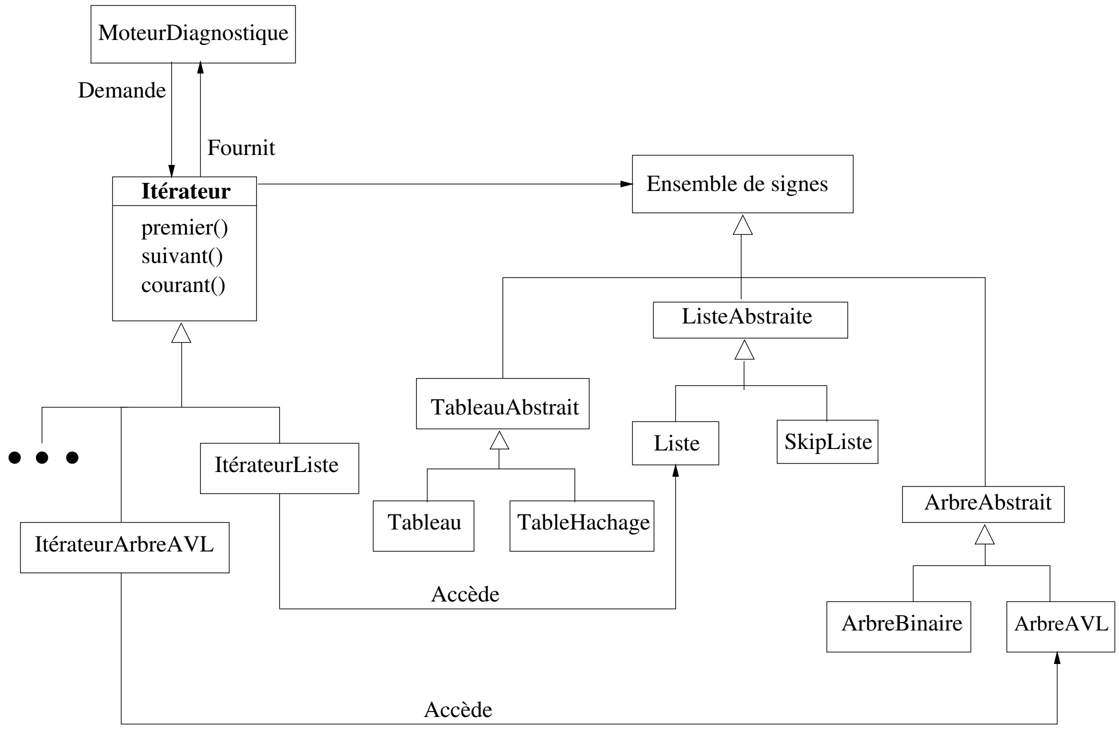

FIG. 2.4: utilisation du pattern `Itérateur`.

L'`Itérateur` définit l'ensemble des méthodes nécessaires pour accéder aux différents éléments d'un agrégat (figure 2.4). Il joue le rôle d'interface entre les implémentations spécifiques de cette agrégation (liste, arbre ...) et le moteur diagnostique qui lui envoie les requêtes d'accès. Les modules clients de l'`Itérateur` disposent donc d'un accès unique quelle que soit la structure effective de cet agrégat.
Il est nécessaire de définir des itérateurs concrets (dans la figure 2.4 `ItérateurListe`, `ItérateurArbreAVL`) qui prennent en charge l'implémentation des procédures spécifiques de parcours. Ces itérateurs héritent de l'`Itérateur` générique qui met à la disposition du client des procédures comme `premier`, `suivant` et qui assure une itération polymorphe sur l'agrégat.

✔ Utilisation

Des méthodes de traitement, nous avons donc extrait les parties d'algorithme qui assurent l'itération sur une description. Grâce à l'`Itérateur`, ces parties communes ont

été factorisées. L'objet `Description` instancie un `Itérateur` en fonction du type d'agrégation des signes. Il s'agit d'un itérateur externe[2] car les appels aux différents éléments (par les fonctions `premier()`, `suivant()`, ...) sont explicitement réalisées par le client : `MoteurDiagnostique`.
Une fois que le `MoteurDiagnostique` a obtenu les signes qu'il demande, par exemple le premier signe observé chez patient et le premier appartenant à la description de la première maladie étudiée, il faut qu'il sache si ces signes ont une composition d'événements et d'attributs identique. La comparaison est effectuée par une méthode de la classe `Signe` qui possède ses propres itérateurs pour traiter ses compositions.

Les langages de programmation fournissent des classes pré-définies pour gérer les différents types d'agrégats. En langage Java, on trouve pour chacune de ces classes : liste, tables de hachage ... les fonctions qui permettent aussi de les parcourir.

✔ Conséquences

Une description de patient ou de maladie est un ensemble de signes et souvent l'accès à ces signes est fortement lié à la méthode de raisonnement utilisée ou à l'implémentation du logiciel : liste prolog, tableau de valeurs statistiques ... La séparation des structure de données et des traitements est un des principes de base de la modélisation objet [37], mais cela suppose le développement de modes de communication entre chaque module. Le pattern `Itérateur` remplit le rôle d'une telle interface entre les données et les modules de traitements.

D'une façon générale, les méthodes de traitement sont les clientes de l'`Itérateur`. Dans notre cas, elles le sont par l'intermédiaire du `MoteurDiagnostique`. Il envoie une requête à l'itérateur afin d'obtenir l'item courant (c.-à-d. le signe courant) mais n'a jamais accès directement à cet item. Ainsi l'implémentation d'une nouvelle structure de données n'implique aucune modification dans les procédures de calcul du diagnostic. Réciproquement,

[2]Aussi appelé actif par G. Booch [20] en référence au rôle du client de l'itérateur.

une modification ou une addition de procédure de calcul ne provoque pas de changement dans l'accès aux définitions.

2.4 Présence ou absence de signe : `Etat`

✔ Intention

Avec le pattern `Etat`, les différents états possibles d'un objet sont représentés explicitement. C'est une alternative à l'implémentation de cet état par une valeur.
Nous avons modifié l'utilisation classique de ce pattern pour qu'il corresponde à nos besoins. Initialement, le pattern `Etat` permet à un objet de modifier son comportement quand son état interne change. Dans notre application, c'est l'interprétation de cet objet par les autres modules qui varie suivant son état plutôt que son comportement. Mais de la même façon notre objet semblera changer de classe. P. Coad dans [33] avait proposé un « *Etat across a collection* » proche de notre conception du pattern `Etat` mais la présentation en était assez succincte, nous continuerons donc à nous référer à la définition plus complète de l'ouvrage de Gamma *et al* pour illustrer notre propos.

✔ Motivation

Le pattern `Etat` concerne la modélisation des objets *Signes*. Dans les descriptions de maladies, le praticien spécifie s'il s'attend à trouver un signe donné, ou si au contraire l'absence de ce signe aura une valeur diagnostique. Dans la plupart des systèmes, un poids est affecté à chaque signe appartenant à la définition d'une maladie. La présence ou l'absence d'un signe est alors codée par l'intermédiaire de ce poids. Or, chaque méthode de raisonnement travaillant avec son propre type de valeur, il est indispensable d'analyser précisément le rôle et la sémantique de cette valeur.
Confondre le codage de la présence et de l'absence avec celui du poids du signe pose deux problèmes. Premièrement, il existe une différence entre les deux définitions suivantes : « *la destruction d'une articulation est rarement observée dans la sclérodermie* », (c.-à-d. la

présence d'une destruction n'est pas significative, et « *on ne doit pas trouver de destruction articulaire dans le lupus* », c.-à-d. l'absence de destruction est en faveur du Lupus.
Si la « *présence* » et l'« *absence* » des signes sont codées dans le poids, il est difficile de fournir une interprétation de ce poids pour ces deux cas en même temps.
Deuxièmement, pour une maladie donnée, le fait de rechercher la « *présence* » versus l'« *absence* » d'un signe est une action commune à toutes les méthodes de raisonnement quel que soit le type de valeur utilisée. Si cette information de « *présence* » ou d'« *absence* » est classiquement codée dans la valeur, il est obligatoire de recoder cette information pour tous les types de valeurs alors qu'en fait elle est indépendante du système de valuation.

En ce qui concerne la description du patient, les signes de cette description ne possèdent pas de valeur et pourtant on désire pouvoir spécifier pour eux aussi leur caractère *présent* ou *absent*. C'est un point important de la définition d'un patient qui est rarement détaillé. Lorsque les valeurs expriment à la fois le poids d'un signe et son degré de présence, l'interprétation de cette valeur est différente suivant qu'elle appartient à la description du patient ou à celle de la maladie.
Pour notre part, nous tenons à respecter notre modèle qui définit un même objet `Description` pour les patients et les maladies. Instancier ces objets avec des éléments sémantiquement différents serait un facteur de confusion dont les conséquences se repercuteraient sur les méthodes de raisonnement. Il faudrait alors prendre en compte et traiter ces différences.
La séparation de ce concept de *présence/absence*, du concept de valeur va permettre une meilleure représentation sémantique de cette information dans le respect de notre modèle.

✔ Structure

Le pattern `Etat` fournit l'interprétation du signe aux autres modules mais le signe n'a aucun besoin de connaître son propre état. Un signe ne modifie pas son comportement car ce n'est pas un objet dynamique mais statique. La plupart des cas d'utilisation du pattern `Etat` rencontrés dans la littérature, concerne des objets qui changent d'état pendant le processus. Ils modifient ainsi leur propre comportement. Pour nous, un signe n'est qu'un objet de description. C'est la description qui possède cette information sur les signes.

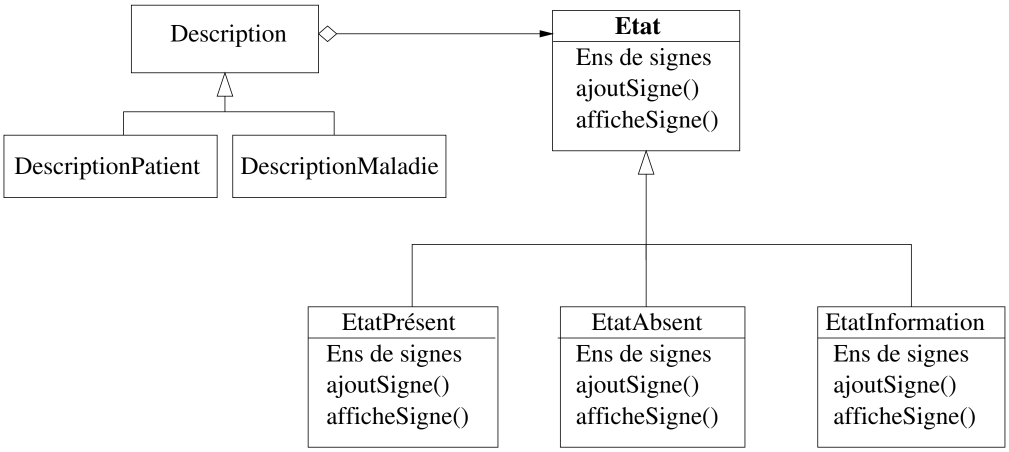

FIG. 2.5: les différents états possibles des signes..

La figure 2.5 montre qu'une `Description` est une agrégation de 3 états possibles : l'`EtatPrésent` et l'`EtatAbsent` dont nous avons déjà parlé, et l'`EtatInformation` dont l'usage est développé au paragraphe suivant. Chacun de ces états contient l'ensemble des signes dans cet état spécifique, ainsi que les fonctions nécessaires pour ajouter, supprimer ou consulter les signes.

✔ Utilisation

Dans cette architecture, les signes ne connaissent jamais leur propre état. C'est l'état qui connaît les signes qui lui appartiennent. Nous avons choisi cette organisation car elle facilite l'écriture de procédure comme : « *chercher tous les signes présents communs à la description du patient et à celles des maladies* », « *exclure les pathologies pour lesquelles le patient n'a pas de signe présent* » ...

La représentation explicite de l'état permet d'exprimer si un signe a été cherché et trouvé (présent) ou non (absent) ou encore, n'a pas été cherché (pas mentionné dans la description du patient).

Pendant la phase de validation de notre modèle, il est apparu qu'un état que nous avons appelé `Information` serait utile. Certaines indications, comme par exemple l'âge et le sexe, peuvent être considérées comme des signes au sens de « caractéristiques du patient », sans pour autant avoir un caractère pathologique ; ils sont toujours présents mais un état présent ou absent n'a pas de sens dans ce cas (il peut juste advenir que l'on ne possède pas cette information). Ces signes informatifs sont une définition du patient « hors contexte » contrairement aux signes présents et absents qui caractérisent son état actuel.
Nous avons donc créé un état supplémentaire : `Information`, pour ces signes. Nous avons étendu ce concept aux *antécédents médicaux* pour exprimer le fait que l'on ne désire pas les considérer comme « *présents* » puisqu'ils ont eu lieu dans le passé, bien que l'on veuille continuer à prendre en compte cette information.

✔ Conséquences

Dans notre application, nous réalisons une description ponctuelle du patient, correspondant aux observations du praticien lors de la consultation. Il n'y a donc pas de changement dynamique d'état pour les signes. Pour les événements dont les attributs sont mesurés, le système calcule à partir de la définition du seuil si l'événement doit être considéré comme présent ou absent, et l'affecte à l'état correspondant. Pour tous les signes dont les événements sont associés à des caractéristiques simples, il suffit au praticien de nommer ces événements et de stipuler s'ils sont présents ou non pour que ceux-ci soient ajouter à la liste correspondante. La liste des signes pouvant appartenir à l'état information est définie de manière fixe. Leur saisie est réalisée séparément et leur affectation à l'état information est automatique.

Cette représentation de l'`Etat` des signes les rend indépendants des méthodes de traitements et simplifie le code. L'utilisation de ce pattern élimine les tests conditionnels dans les traitements.
Ce pattern a modifié l'analyse sémantique du système en ce qui concerne les valeurs de

signes. Il a ajouté un niveau supplémentaire d'interprétation, indépendant du système de valuation. De plus, le concept de signes « présents » est un bon reflet de la classification faite par le praticien pendant qu'il observe le malade.

2.5 Procédure de raisonnement : `Stratégie`

✔ Intention

Le rôle du pattern `Stratégie` est de définir une classe abstraite d'algorithmes. Il encapsule chacun d'entre eux et les rend interchangeables. De cette façon chaque algorithme peut évoluer indépendamment des clients qui l'utilisent.

✔ Motivation

Puisque notre objectif consiste à implémenter dans le système plusieurs méthodes de raisonnement, et donc de traitement, nous avons ajouté un degré de complexité supplémentaire lié au besoin de gérer cette diversité. Une des solutions pour maîtriser cette complexité est l'encapsulation de ces méthodes dans des modules indépendants. Cela permet de développer chacun des modules séparément sans tenir compte de la présence des autres. Mais on risque alors d'introduire un surcoût de développement, dû au fait que les modules clients doivent alors s'adapter à chaque méthode existante.
L'utilisation du pattern `Stratégie` est un moyen de se garantir contre ces problèmes car il définit une interface commune à tous les modules qui implémentent une méthode de diagnostic. Cette pratique est classique pour les concepteurs de logiciels surtout dans le « monde objet ». Mais dans la plupart des systèmes d'aide à la décision, le logiciel est organisé autour d'une unique méthode de raisonnement, sorte de « marque de fabrique » du logiciel : base de règles, système statistique, réseaux ... Dans ce cas, le raisonnement est l'élément central du système et toute l'organisation est fortement dépendante du processus de raisonnement choisi. Si celui-ci change, le système doit être reconstruit.

Les principes de modularité sont souvent préconisés en Génie Logiciel mais rarement observés dans les programmes d'Intelligence Artificielle. La modularité et l'encapsulation des traitements sont des éléments indispensables dans une application qui partage réellement la base de connaissances entre plusieurs modules de raisonnements.

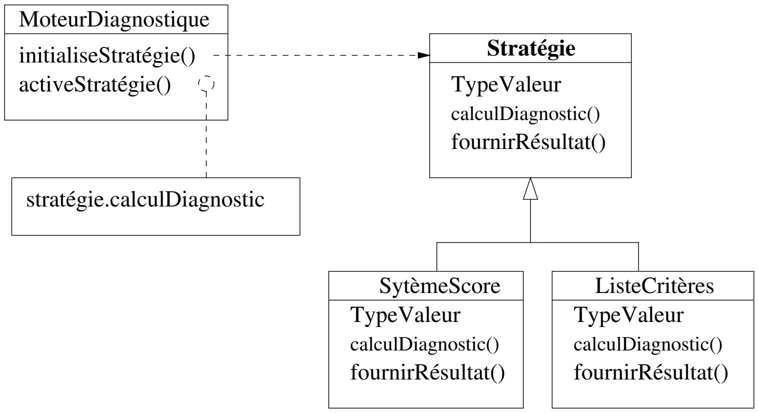

FIG. 2.6: utilisation du pattern `Stratégie`.

✔ Structure

Les deux familles de traitement, le `Système de Score` et les `Listes de Critères`, sont modélisées grâce à deux classes qui héritent de la classe abstraite `Stratégie`. Chaque famille modélise une méthode de raisonnement différente et implémente son propre comportement : recherche de tous les signes observés pour le premier, validation d'une liste restrictive de signes pour le second. Grâce au mécanisme d'héritage, les classes dérivées ont la même interface. De ce fait passer d'une méthode à une autre reste transparent pour les modules clients. De plus, nous pouvons définir des procédures partagées par toutes les méthodes : « *prendre en compte uniquement les signes présents* », « *considérer comme étant absents tous les signes cliniques non spécifiés pendant l'examen du patient* » etc. Ces procédures sont définies dans la super classe `Stratégie`.

✔ Utilisation

Le `MoteurDiagnostique` a la responsabilité d'instancier un objet `Stratégie` avec une des méthodes de traitement. C'est également lui qui fait ensuite appel aux opérations de calcul du diagnostic.
Le rôle de chaque module diagnostique consiste à définir le `Type de Valeur` nécessaire à l'algorithme, à implémenter les `opérations` qui assurent la construction du diagnostic et à `transmettre un résultat` à son client : le `MoteurDiagnostique`.

✔ Conséquences

Avec l'organisation de la figure 2.6, le module utilisateur de `Stratégie` doit avoir connaissance de ce que produit chaque méthode. Chaque module héritant de `Stratégie` encapsule le traitement spécifique qu'il implémente, c.-à-d. comment il obtient le résultat. Toutefois, il communique le type de valeur dont il a besoin pour calculer ce résultat, ainsi que le type du résultat proprement dit. Pour ce faire, il est donc nécessaire de rompre l'encapsulation du module `Stratégie`. C'est un choix dicté par le fait que nous avons voulu créer un « *manager* » général du système qui dialogue avec l'utilisateur et organise l'enchaînement des procédures. Si nous avions choisi au contraire, de conserver ces informations cachées, les différentes méthodes de raisonnement auraient produit un résultat qui n'aurait pas pu être agrégé pour élaborer un avis diagnostique global.

Ce pattern est issu d'une motivation classique de la conception des logiciels : créer une hiérarchie de traitements. En effet si nos deux méthodes de traitement modélisent deux raisonnements différents, elles n'en relèvent pas moins de concepts algorithmiques identiques (type de valeur, opération, production de résultats). L'introduction dans ce système de méthodes de raisonnement basées sur des outils différents (réseaux de neurones, arbres de décision ...) conduirait à la définition de nouvelles familles d'algorithmes, et donc de nouvelles classes `Stratégie`. Chacune d'entre elles implémenteraient des méthodes spécifiques. Nous n'avons pas implémenté de telles classes mais elles pourraient être écrites comme suit :

- `Stratégie arbre de décision :`
  ```
  Type de Valeur
  construction Arbre()
  opération()
  fournir Résultat()
  ```
- `Stratégie réseaux de neurones :`
  ```
  Type de Valeur
  construction Réseaux()
  apprentissage()
  simulation()
  opération()
  fournir Résultat()
  ```

2.6 Architecture du système

Grâce aux patterns, nous avons décrit des éléments toujours présents dans un système d'aide au diagnostic mais dissimulés à l'intérieur des objets identifiés par l'analyse du domaine [17]. Les patterns apportent ainsi un point de vue complémentaire du système, plus orienté vers l'architecture du système. Il faut noter que ces patterns mis en évidence pendant la phase de conception, ne sont pas dépendants d'une implémentation spécifique [58]. En ce sens les patterns ne sont pas assimilables aux librairies d'un langage de programmation.

2.6.1 Collaboration des patterns : un langage de patterns

Dans notre application, nous avons identifié quatre patterns présents dans le modèle à différents niveaux. Deux patterns ont été intégré à des positions stratégiques pour encapsuler la représentation interne des signes (`Composite`) ou pour rendre effective le polymorphisme des traitements (`Stratégie`). Les deux autres ont amélioré l'implémentation du logiciel en séparant l'interprétation des signes du type de valeur (`Etat`) et en extrayant une tâche générique (`Itérateur`) des algorithmes de traitements.

La figure 2.7 présente la position de ces patterns dans le modèle abstrait du système, lui même présenté dans la figure 1.14. Certains viennent remplacer des objets identifiés par l'analyse (`MéthodeRaisonnement` par `Stratégie`), ou complètent le modèle initial (`Composite`), et d'autres s'intercalent et modifient la communication entre les objets (`Itérateur` et `Etat`).

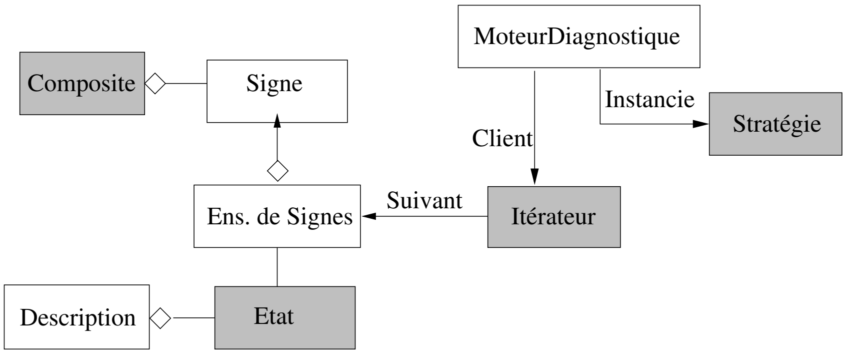

FIG. 2.7: la collaboration des 4 patterns.

Lorsqu'un ensemble de patterns permet de décrire un type de système, on parle alors de langage de patterns. Notre langage de patterns permet de décrire une **composition** de signes dans un certain **état**, auxquels on accède grâce à un **itérateur**, afin de leur appliquer une **stratégie**.

Nos quatre patterns associés aux objets de base du modèle abstrait définissent le noyau d'un système d'aide au diagnostic. Ce langage de patterns est un noyau fonctionnel. L'analyse fournit un modèle du système, les patterns nous donnent les moyens d'organiser réellement le fonctionnement de ce système [13].

Notre système d'aide au diagnostic définit un mode de représentation des connaissances, un accès à ces connaissances et une stratégie de traitement de ces connaissances. Les patterns ne résument pas « *ce qu'est* » un système d'aide au diagnostic, mais ils donnent un squelette d'un tel système en relation avec le modèle objet de l'analyse. La figure 2.7 montre cette superposition.

2.6.2 Management du système : `Superviseur`

L'introduction des patterns a induit la création de nouveaux objets : quatre classes d'objets leur correspondant, trois classes dérivées pour les états, et les différentes structures travaillant avec `Itérateur`. Plus de vingt objets composent le cœur du système d'aide au diagnostic, ils sont montrés dans la figure 2.8 (les modules liés à l'interface graphique ne sont pas représentés). Le modèle est organisé autour de l'objet `SystèmeDiagnostique`

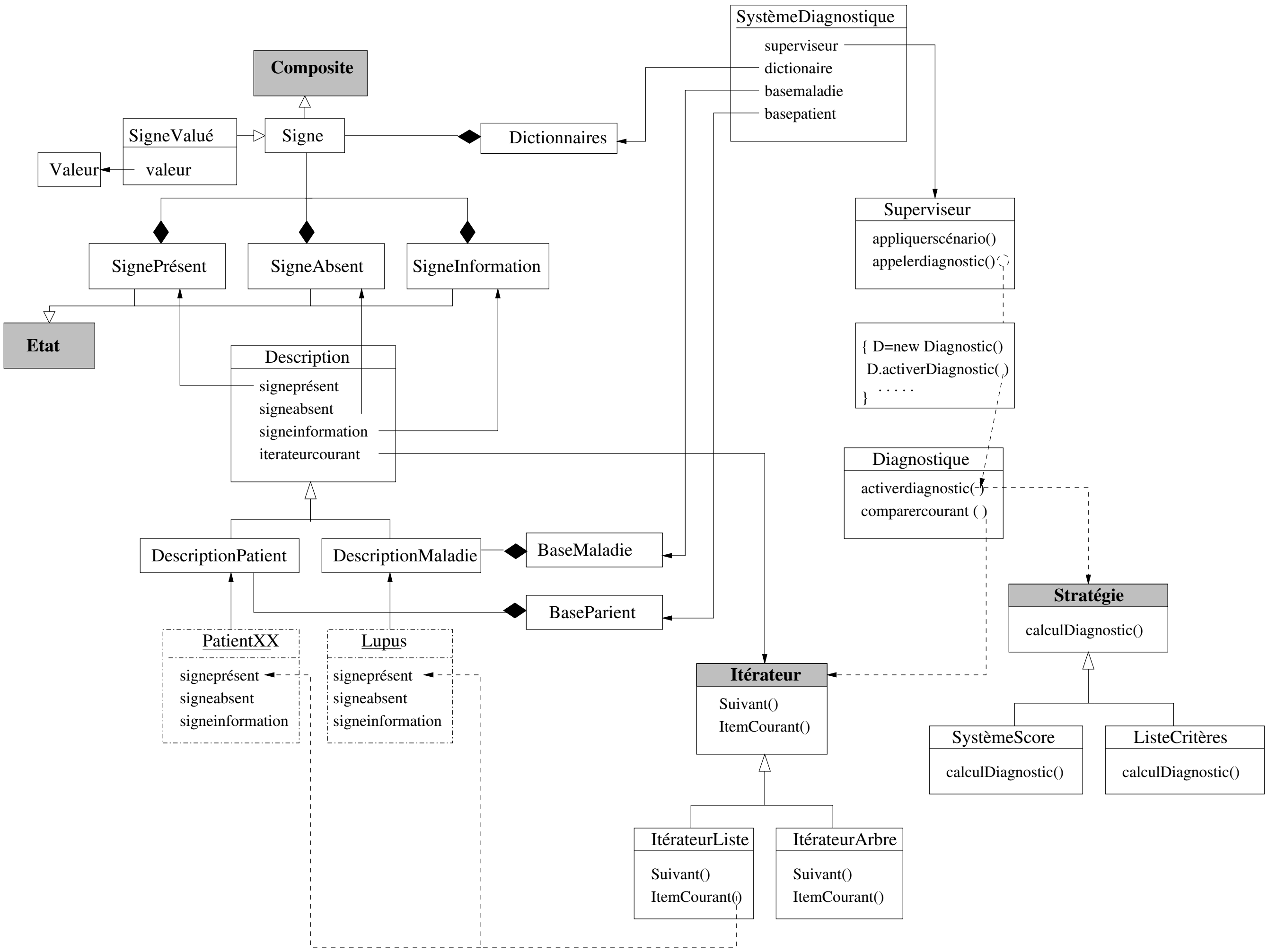

FIG. 2.8: les éléments principaux du modèle final.

qui est l'interface entre le système et l'utilisateur.

Le `SystèmeDiagnostique` instancie un superviseur, une base de maladies, une base de patients et des dictionnaires. Chaque maladie et chaque patient sont décrits grâce à trois ensembles de signes triés par état. L'accès à chaque `Signe` appartenant à une *Description* est réalisé par un objet `Itérateur`. Un nouvel objet appelé `Superviseur` est défini afin de mettre en œuvre l'agrégation des résultats des deux méthodes de raisonnement. Il instancie un objet `MoteurDiagnostique` pour chaque `Stratégie` choisie. Le `Superviseur` mémorise les résultats produits par les deux traitements et dispose d'une méthode `scénario` qui décrit comment cette collaboration fonctionne : « *quelle procédure de raisonnement* *est appliquée la première ? quand est appelée la suivante ?* ». Le scénario d'enchaînement des procédures et d'agrégation des résultats est décrit en détail dans le chapitre suivant, section 3.3.3.

2.7 Apport des patterns et des objets dans l'architecture d'un logiciel

Les patterns offrent un « autre visage » de l'analyse orientée objet. Ils donnent un point de vue différent du système, plus efficace et plus orienté vers l'architecture logicielle. Grâce aux patterns, le modèle objet a été amélioré par l'addition d'abstractions relevant du problème.

En réponse à Heathfield dans un débat sur le professionnalisme en informatique médical [68], Giuse a remarqué l'absence de guide ou d'ouvrage de références sur les règles précises de conception logicielle, il n'existe pas non plus de modules ou de bibliothèques de programmes bien documentés pour l'aide à la décision médicale [57]. L'approche des concepteurs de patterns est directement liée à ce souci. Ils définissent un vocabulaire communs pour des modules bien identifiés et qui correspondent à des tâches précises.

Dans la pratique courante, l'utilisation de ces outils ne fait que commencer car ils supposent une connaissance préalable de la conception et du développement orientés objet. Même si Doug Lea explique que lorsque les patterns sont intégrés, les choses paraissent si évidente qu'il vous semble que « *même votre grand-mère aurait pu y penser* » [86], l'utilisation des patterns ne dispense pas d'une bonne « culture » objet.

Les catalogues de patterns comme « Design Patterns » sont une source d'inspiration et une opportunité de partager l'expérience des « experts » en conception orientée objet. Les patterns ne remplacent pas les méthodes objet, ils les complètent.

Par exemple, le principe d'encapsulation, toujours recommandé dans les méthodes orientées objet, n'est pas facile à mettre en œuvre. L'`Itérateur` donne tous les moyens pour respecter ce principe. Il est un bon exemple d'un objet qui modélise correctement une tâche du système, mais qui n'aurait pas été spontanément implémenté avec ce niveau d'abstraction sans l'exemple de ce pattern.

La démarche orientée objet couvre toutes les étapes de la construction d'un système informatique, son analyse, sa conception puis son implémentation grâce aux langages objet. Les modèles de conception sont particulièrement utiles dans le passage d'un modèle d'analyse à un modèle de développement. En effet, l'analyse seule ne suffit pas pour obtenir un système informatique. La conception d'un système, souple et réutilisable fera émerger des objets qui ne figurent pas dans le modèle d'analyse.

Le fait de réutiliser un logiciel nécessite fréquemment de le réorganiser ou de le reconstruire [104]. Cette notion est encore plus réelle lorsqu'il s'agit d'Intelligence Artificielle. D'ailleurs T. Menzies [94] voit dans cet objectif de réutilisation un véritable leurre. Il compare même l'introduction des patterns à celle du *Knowledge Level* (cf section 1.2.6), basée sur les mêmes objectifs (dont celui de la réutilisabilité) mais n'ayant pas réellement atteint ce but de programmation d'objets stables et immuables.

A notre sens, il faut distinguer patterns et langage de patterns. Celui-ci a pour vocation de décrire l'organisation de fonctionnalités, il se rapproche en ce sens du *Knowledge Level* ; alors que ceux-là sont des briques indépendantes de toute application, un sorte de « morceaux » de fonctionnalité. En descendant à un niveau plus fin d'analyse, on s'aperçoit qu'il est possible de reconnaître des « pas » communs aux différents algorithmes et de les factoriser pour en faire des objets réutilisables du système. Chacun de ces « morceaux » est identifié universellement par un nom, celui du pattern. La communauté informatique peut ainsi partager cette fonctionnalité. Selon W. Zimmer [143], ce vocabulaire commun est l'avantage principal des *Design Patterns*. A partir de ces objets élémentaires, le concepteur élabore un langage de patterns, orienté vers son domaine d'application. E. Gamma reconnaît qu'il n'est probablement pas possible de proposer une offre globale pour toute architecture logicielle, à l'opposé d'Alexander qui prétend fournir grâce à son catalogue de patterns un langage de modélisation pour toute architecture de bâtiment.

Chapitre 3

Réalisation du logiciel

- Les ordinateurs ne peuvent faire que ce qu'on leur dit de faire.
En un sens, c'est vrai, mais cette réflexion passe à coté de l'important : on ne connaît pas à l'avance les conséquences de ce qu'on dit à un ordinateur de faire.

D. Hofstadter, *Gödel Escher Bach, chap. X (1985)*

Le modèle issu des phases d'analyse et de conception a donné lieu à l'écriture d'un système d'aide au diagnostic : LADRI (Logiciel d'Aide au Diagnostic en Rhumatologie Inflammatoire). Ce chapitre lui est consacré.
La première section présente le contexte dans lequel cette réalisation a été effectuée, et la façon dont les données ont été recueillies. Nous expliquons ensuite l'intégration des données dans le système, grâce à l'usage d'une grammaire formelle pour les données stockées sur disque, ou par l'intermédiaire d'une interface pour leur saisie interactive. Puis, nous détaillons les deux procédures de traitement liées aux deux modes de raisonnement, ainsi que l'algorithme qui permet l'agrégation de leurs résultats. Nous terminons avec les résultats obtenus par l'analyse d'une centaine de dossiers patients et une discussion sur les perspectives de ce logiciel.

3.1 Les données du problème

Les informations et les données médicales utilisées par LADRI, ont été obtenues auprès du service de rhumatologie du Centre Hospitalier Universitaire de Bordeaux, dirigé par le Professeur J. Dehais. La réalisation de notre application a été menée en collaboration avec le Docteur J. Ph. Vernhes, chef de clinique de ce service au commencement du projet. Ce praticien a rempli pour nous le rôle de spécialiste du domaine. Cette collaboration s'est poursuivie pendant trois années et nous a permis de disposer de données concernant les patients du CHU de Bordeaux et du Centre Hospitalier de Libourne.

3.1.1 Recueil de l'expertise

La première phase de ce projet a consisté à définir les limites du domaine analysé. La rhumatologie inflammatoire est une branche de la rhumatologie, assez vaste pour fournir un cadre représentatif du diagnostic au sein d'une spécialité médicale.
Notre spécialiste a retenu 13 pathologies recouvrant la majeur partie des cas de rhumatismes inflammatoires. Ces pathologies sont :

- *le lupus érythémateux disséminé,*
- *la spondylarthrite ankylosante,*
- *l'arthrite réactionnelle,*
- *la dermatomyosite,*
- *la pseudo-polyarthrite rhizomélique,*
- *la chondrocalcinose,*
- *la maladie de Still.*
- *la polyarthrite rhumatoïde,*
- *le rhumatisme psoriasique,*
- *la sclérodermie,*
- *la polymyosite,*
- *la maladie de Horton,*
- *la goutte,*

Si l'on fait exception de la *goutte* et de la *chondrocalcinose*, les maladies que nous avons retenues sont des maladies dites « systémiques » [1].

La deuxième phase a été pour nous une phase de familiarisation avec ce domaine médical. Dans cet optique, la consultation des archives du service de rhumatologie et d'ouvrages

[1]Maladies systémiques : pathologies touchant effectivement tous les éléments d'un même système ou tissu. Une maladie systémique recouvre un ensemble d'affections disparates (dites parfois « inflammatoires » non spécifiques) qui n'ont en commun que l'obscurité de leur origine (appelée aussi origine idiopathique) [56].

destinés aux étudiants en médecine, nous a permis de prendre connaissance du vocabulaire et des concepts médicaux manipulés par les praticiens.

Les rhumatismes inflammatoires sont des pathologies chroniques. La plupart des patients sont suivis régulièrement (soit en consultations externes, soit au cours d'hospitalisations) sur de très longues périodes.

Le dossier patient détenu par le service sous forme « *papier* », est essentiellement constitué des éléments suivants :

- de compte rendus d'examens cliniques réalisés pendant les consultations,
- de compte rendus d'examens biologiques transmis par les laboratoires d'analyse,
- de compte rendus d'examens radiologiques (souvent accompagnés des clichés),
- de documents stéréotypés remplis par les étudiants en médecine (compte rendus de visite).

La phase suivante a été celle du recueil de l'expertise proprement dite. Ce recueil a débuté par un entretien enregistré, de plus de 4h. Cet entretien a permis de fixer l'essentiel des définitions, sans chercher à établir une complète valuation des signes. Le détail des définitions et la valuation des signes se sont mis en place progressivement, en parallèle avec les différentes phases de développement du système.

Notre travail a ensuite consisté à mener conjointement la conception du logiciel, et la validation progressive du modèle. Cette validation a été réalisée grâce à des prototypes successifs qui ont permis de tester l'organisation des données et les réponses apportées par ce modèle aux problèmes de mise en place du système. Des entretiens réguliers avec le praticien ont permis la définition d'une échelle de valeur pour le système de score, et la valuation des définitions. Au total il a été sollicité pour une cinquantaine d'heures de travail destinées à la formalisation de ses connaissances sur les 13 pathologies et à son intégration dans le modèle. La figure 3.1 montre la traduction de quelques éléments de la description de la *maladie de Still* dans ce système.

Evénement	*Attribut*	*Etat*	*Valeur*
douleur articulaire	asymétrique	présence	parfois
destruction articulaire	périphérique	présence	parfois
atteinte cutanée	macules rosées	présence	généralement
facteur rhumatoïde		absence	très général

FIG. 3.1: éléments de la description de la *maladie de Still*

Le praticien n'a jamais eu à valider le modèle informatique des données, c.-à-d. l'architecture telle que nous l'avons décrite dans la figure 2.8. Par contre, il en a validé le résultat, c.-à-d. d'une part la possibilité de construire des définitions par composition d'événements et d'attributs, et d'autre part le mode opératoire du système de score (échelle de valeur, type de valeur utilisé).
Au cours de l'avancement du projet, plusieurs saisies de dossiers patients ont été réalisées. Les tous premiers dossiers saisis n'ont pas été conservés dans les résultats finals car ils ont servi à définir l'échelle de valeur du système de score.

3.1.2 Les critères de classification

Les données nécessaires à l'utilisation de la deuxième méthode de raisonnement, basée sur les listes de critères diagnostiques, ont ensuite été incluses dans le système.

Les critères de classification que nous avons utilisés sont :

- Les critères de l'European Spondylarthropathy Study Group [44] et d'AMOR [6] pour les *spondylarthropathies*.
- Les critères de la *polyarthrite rhumatoïde* proposés par l'ACR [5].
- Le *lupus érythémateux disséminé* proposés par l'ACR [103]
- Les critères de la *polymyosite* et de la *dermatomyosite* proposés par Tanimoto [129].
- Les critères de la *sclérodermie* proposés par l'ACR [93].
- Les critères de la *maladie de Horton* proposés par l'ACR [74].
- Les critères de la *maladie de Still* proposés par Kahn [79] et Yamaguchi [142]

Ces critères reposent sur des résultats statistiques obtenus à partir d'enquêtes épidémiologiques. Les maladies systémiques n'ont pas de signes pathognomoniques[2], leur identification est fondée sur un faisceau d'arguments cliniques, biologiques, radiologiques, ou histologiques. Une des démarches diagnostiques possibles est l'élaboration de critères nosologiques [3] définis à partir de grandes séries de dossiers médicaux [7].
Ces listes de critères diagnostiques ont été intégrées au système. L'intégration réussie de cette autre « *forme* » de connaissances a permis la validation de l'architecture objet définie à partir des données fournies par l'expertise. En effet, l'essentiel de l'architecture et principalement l'organisation des données, a été conçue à partir du raisonnement du praticien. En introduisant dans le système les informations liées aux listes de critères, nous avons pu tester les capacités de modification et d'évolution de notre organisation.

Le point de vue du praticien par son caractère « transversal » cherche à considérer toutes les pathologies de façon homogène. Il nous fournit les éléments utiles à la découverte de signes génériques partagés par un ensemble de maladies. Les listes de critères tentent au contraire d'isoler les caractéristiques les plus spécifiques des pathologies. En conséquence, leurs règles de construction à partir des événements et des attributs, sont souvent plus riches : associations obligatoires entre plusieurs items, avec exclusion d'autres signes. Par exemple, le critère numéro 5 du *lupus érythémateux disséminé* (cf. annexe B) est *une arthrite non-érosive (sans destruction) touchant au moins deux articulations périphériques.* L'introduction de ces formes plus complexes n'a pas provoqué de remise en cause de l'architecture initiale. La figure 3.2 montre une partie des critères diagnostiques de la *maladie de Still* selon [79], tels qu'ils ont été pris en compte par le système.

[2] Pathognomonique : qui est spécifique d'une maladie, signe ne se rencontrant que dans une maladie bien définie et suffisant à lui seul à la caractériser et à poser le diagnostic [56].

[3] Nosologie : étude des caractères distinctifs qui permettent de définir les maladies et de les classer [56].

Evénement	*Attribut*	*Etat*	*Valeur*
douleur articulaire		présence	critère majeur
atteinte cutanée	macules rosées	présence	critère majeur
douleur	pharynx	présence	critère mineur
antécédent personnel	polyarthrite rhumatoïde	présence	critère d'exclusion

FIG. 3.2: structuration de quelques critères diagnostiques de la *maladie de Still*.

3.2 La mise en forme des données

L'architecture résultant des phases d'analyse (cf paragraphe 1.3.1) et de conception (cf figure 2.8) met l'accent sur un concept de description générique, englobant maladies et patients. Nous avons donc réalisé un système centré sur cet objet `Description`. Que le programme manipule un patient ou une maladie (lecture, saisie, affichage, ...), le cœur des procédures est le même.

Ce système gère à la fois des données stockées sur disque et des données saisies interactivement. C'est à partir de ces données que le programme instancie les objets du modèle. Les données sur disque sont relues au moyen d'une grammaire formelle. La saisie interactive est contrôlée par une interface.

Nous présentons le fonctionnement de la grammaire à travers l'exemple des descriptions des maladies. A chaque session cette tâche de relecture de l'ensemble des descriptions de maladies est obligatoire. Lorsqu'on désire relire les données de le base de patients (par exemple pour revoir un dossier déjà mémorisé), le processus est identique.

La saisie interactive est abordée par l'exemple de la saisie des données d'observation d'un patient. L'ajout ou la modification d'une maladie se fait sur le même principe, mais l'accès à ces procédures est réservé au responsable de la maintenance des descriptions de maladies.

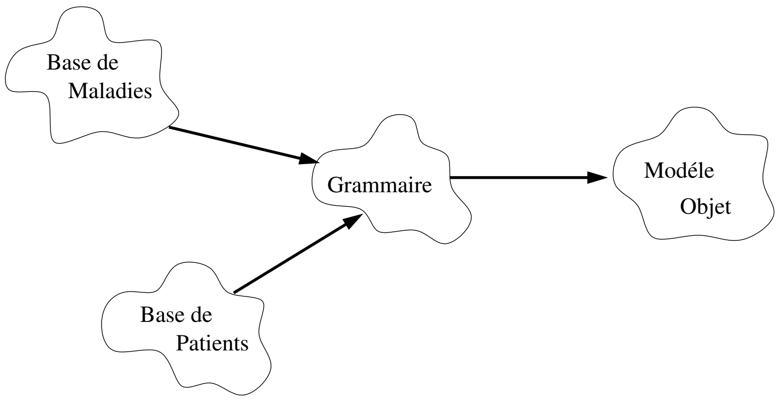

FIG. 3.3: gestion des données.

3.2.1 L'accès aux données de la base de connaissances : la grammaire

Comme le montre la figure 3.3 les données recueillies sont stockées dans des fichiers à plat[4], lus au moyen d'une grammaire non-contextuelle ou BNF (acronyme de Backus-Naur Form).
Une grammaire non-contextuelle comprend quatre composants :

- un ensemble d'unités lexicales appelées symboles terminaux ;
- un ensemble de non-terminaux ;
- un ensemble de règles de production où chaque règle est constituée d'un non-terminal, appelé *partie gauche* de la règle, d'une flèche, et d'une suite d'unités lexicales et de non-terminaux appelée *partie droite* de la règle ;
- la désignation d'un des non-terminaux en tant que symbole de départ ou axiome.

On pourra trouver une présentation complète de l'utilisation des grammaires pour l'écriture de compilateurs dans l'ouvrage de A. Aho *et al.* [2]. Les grammaires initialement conçues pour décrire des langages de programmation permettent de garantir la correction syntaxique et dans une certaine mesure lexicale d'un ensemble de données fournies en entrée.

[4] à plat en opposition aux fichiers dont l'accès est indexé.

Nous avons donc écrit l'ensemble des règles de production qui régissent l'instanciation des objets du modèle à partir des fichiers contenant la base de connaissances. Dans cette grammaire le symbole de départ est **Base**, les symboles terminaux sont écrits en majuscules et les symboles non-terminaux en minuscules. Lorsque plusieurs règles peuvent être appliquées au même non-terminal, on utilise l'opérateur `OU` représenté par une barre verticale |. Chaque règle de production est de la forme :

non-terminal	→	non-terminaux
	|	TERMINAUX.

La figure 3.4 donne l'essentiel des règles de production de la grammaire qui permet de lire la `base de maladies`. Les TERMINAUX : NOM, ENTIER, BORNE_INF, ... sont reconnus grâce à l'analyseur lexicale.

Base	→	Base maladie
maladie	→	NOM ens_signes
ens_signes	→	ens_signes signe
	|	signe
signe	→	ens_événements (valeur)
ens_événements	→	ens_événements événement
	|	événement
événement	→	NOM [ens_attributs]
	|	NOM
ens_attributs	→	ens_attributs attributs
attributs	→	NOM
	|	NOM borne
borne	→	BORNE_INF ENTIER
	|	BORNE_SUP ENTIER
	|	BORNE_INF ENTIER BORNE_SUP ENTIER
valeur	→	ENTIER NOM RÉEL
	|	ENTIER RÉEL
	|	ENTIER NOM

FIG. 3.4: règles syntaxiques de lecture de la `base de maladies`

Grâce à cette grammaire formelle, nous pouvons conserver la base de connaissances dans un format texte indépendant de toute implémentation. Les règles qui lient cette base à la structure des objets instanciés en mémoire, sont définies explicitement dans la grammaire.
La description du patient peut provenir de la `base de patients`, fichier lui aussi relu par une grammaire (très proche de celle utilisées pour les maladies). Pour les données saisies, les différents contrôles de cohérence pour instancier les objets sont dévolus à l'interface.

✔ Avantages de l'utilisation d'une grammaire

L'utilisation d'un grammaire comporte de nombreux avantages. Premièrement, l'ensemble des règles de traduction du fichier texte pour instancier le modèle objet est réuni dans un seul fichier, de petite taille et au maniement aisé pour toute personne connaissant le principe d'un analyseur syntaxique.
Ensuite, il existe de nombreux outils qui permettent de générer automatiquement l'arbre syntaxique correspondant à la grammaire qui a été conçue. Nous avons pour notre part utilisé l'analyseur syntaxique *bison* et l'analyseur lexical *flex*[5] lors d'un premier prototype du système en langage C, puis leurs équivalents pour le langage Java : JavaCup et JavaLex[6]. Actuellement, il est possible d'utiliser JavaCC[7] qui est à la fois un analyseur lexical et un analyseur syntaxique.
Enfin, ces outils facilitent la compréhension des liens qui unissent les objets et autorisent la modification et l'ajout de nouvelles règles de réécriture, engendrées par exemple par une modification du modèle. La grammaire est une sorte de traduction du modèle conceptuel des données à l'usage du programme.

[5]ces logiciels distribués librement sont directement inspirés de yacc et lex[89]. Ils peuvent être téléchargés depuis les sites miroirs gnu comme http://ftp.jussieu.fr/pub/gnu/
[6]JavaCup [73] et JavaLex [10] sont disponibles à l'URL http://spectral.mscs.mu.edu/javadev/parsers/
[7]http://www.suntest.com/JavaCC/download.html

3.2.2 La création des données : la saisie interactive

L'analyse (paragraphe 1.3.1) nous a conduit à structurer le vocabulaire du praticien en *événements* et *attributs*. Ces termes sont regroupés dans des dictionnaires (figure 1.11). Le dictionnaire d'événements comporte à ce jour, 25 événements organisés en couple événement - mot-clés, le dictionnaire d'attributs contient 81 attributs organisés en couple mot-clé - attributs. C'est à partir de ce vocabulaire « *autorisé* » qu'est contrôlée la saisie du patient.

La saisie est organisée à partir de concepts habituels pour le praticien tels que l'identité de son patient (nom, sexe, âge), et les signes qu'il observe ou non chez celui-ci.
L'interface reconstitue pour lui le concept de signes, à partir des événements et des attributs. Le praticien construit la description de ses observations (les signes) à partir d'un nom choisi dans la liste proposée par l'interface (les noms des événements) auquel il peut associer des caractéristiques choisies dans une seconde liste apparaissant dynamiquement (les noms des attributs « autorisés » pour cet événement). Les mot-clés sont utilisés pour gérer ces affichages mais ne sont pas obligatoirement visibles pour le praticien.
Cette organisation de la saisie relève d'un souci d'ergonomie par rapport à l'utilisateur et n'est pas directement le reflet de l'organisation interne du programme. Ainsi, la distinction entre les concepts d'attribut et d'attribut_à_mesurer n'est pas montrée à l'utilisateur. Choisir un attribut_à_mesurer dans la liste des attributs provoque simplement l'affichage d'une zone de saisie de la mesure réalisée sur le patient.

La saisie à partir de listes préétablies dispense de vérification orthographique, de gestion de synonymies, ... et garantit que les événements décrivant un patient figurent dans la description d'au moins une pathologie. Ils seront donc « interprétables » par le système.
Cette forme de saisie est plus rapide pour le praticien qu'une saisie par texte libre, mais limite les possibilités de descriptions de l'état du patient au contenu de la base de connaissances à l'instant t.
Il est bien sûr possible de réaliser des saisies de maladies sur le même modèle. Si le praticien

désire utiliser un nouvel événement, il devra alors créer cet événement dans le dictionnaire et déclarer quels attributs il souhaite lui affecter. Ces différents éléments seront ensuite disponibles pour la saisie de la pathologie ou des patients.

Les objets `Descriptions` possèdent des méthodes de sauvegarde. Elles permettent de recréer les fichiers de la base de connaissances dans le respect des règles de grammaire. Le modèle de description commun à la maladie et au patient permet de factoriser la majeure partie des procédures de traitement (lecture, saisie, sauvegarde) de ces objets.

3.3 Les méthodes de raisonnement

3.3.1 L'évocation des diagnostics possibles : le système de score

Le système de score utilise les données fournies par le praticien. Après avoir recensé les signes appartenant aux descriptions des maladies, chaque signe a été valué par le rhumatologue à l'aide d'une échelle ordinale à 7 valeurs (cf. la figure 3.5) où chaque valeur symbolique a été associée à une valeur numérique. Les valeurs *jamais* et *toujours* sont peu acceptées pour qualifier les maladies sans signes pathognomoniques, elles n'ont donc pas été intégrées à cette échelle. Les valeurs symboliques sont plus parlantes pour le rhumatologue, ce sont donc celles-ci qui ont été utilisées lors des entretiens concernant la définition des maladies.

Valeur symbolique	Valeur numérique
très rarement	2
rarement	3
parfois	4
souvent	5
le plus souvent	6
généralement	7
très généralement	8

FIG. 3.5: échelle de valuation des signes

✔ Méthode de calcul

Dans un système de score, on définit pour chaque signe appartenant à la définition d'une pathologie le poids de ce signe dans le diagnostic de cette pathologie. Ces poids sont ensuite sommés pour représenter le total de la pathologie. Ce total définit le nombre de points maximum qu'un patient peut marquer pour le diagnostic de cette pathologie.

Chaque signe observé chez le patient représente un nombre différent de point en fonction de la maladie considérée. Au terme de la procédure, on connaît la somme des points obtenus pour chaque maladie. Ces scores sont ensuite standardisés afin d'être rendus comparables et ordonnés dans une liste.

Le système de score fonctionne avec la règle suivante :

$Soit$ W_{ij} le poids du signe S_i dans le diagnostic de la maladie M_j,

$Soit$ un patient observé O et la fonction $O(S_i)$ définie comme suit :

SI le signe S_i appartient à l'ensemble des signes du patient

$ALORS \quad O(S_i) = 1$

$SINON \quad O(S_i) = 0.$

Ce patient O obtient pour la maladie M_j le total T_j qui est égal à :

$$T_j = \frac{\sum_i O(S_i).W_{ij}}{\sum_i W_{ij}}$$

Pour ce traitement, il n'est pas nécessaire que notre spécialiste donne, pour chaque maladie, une valeur à tous les signes appartenant à la base de connaissances. La description d'une maladie ne contient strictement que les signes qui la caractérisent. De cette façon, on n'augmente pas inutilement la taille de cette base. Dans le cas où aucune information n'est fournie sur le poids d'un signe dans le diagnostic d'une maladie, la valeur 0 lui sera assignée au moment du calcul.

Exemple : Suivant les définition données en annexe A, un patient qui présente un *Syndrome de Raynaud avec une dystrophie capillaire* marquera 4 points pour la *Polyarthrite rhumatoïde*, la *Dermatomyosite*, la *Polymyosite* et le *lupus*, et 6 points pour la *Sclérodermie.*

✔ Raffinement de l'interprétation du score

Le calcul prend également en compte les trois états possibles pour les signes. Nous avons vu que les maladies sont souvent décrites, d'une part en combinant la présence ou l'absence de signes pathologiques, d'autre part à l'aide d'un ensemble de données non-pathologiques comme le *sexe*, l'*âge* ou les *antécédents médicaux* du patient. Pour chaque pathologie le module de score calcule la somme des poids des signes *présents, absents* et d'*information* : PW_{ij}, AW_{ij}, IW_{ij}.

$$T_j = \frac{\sum_i O(S_i).PW_{ij} + \sum_i O(S_i).AW_{ij} + \sum_i O(S_i).IW_{ij}}{\sum_i W_{ij}}$$

Il est ensuite possible d'analyser l'influence sur le total T_j de chaque ensemble de signes (*présents, absents* ou *information*). Si le patient n'a pas de signe présent, soit $\sum_i O(S_i).PW_{ij} = 0$, la maladie M_j est éliminée de la liste des maladies possibles. Cette opération permet d'éviter de diagnostiquer sur la seule base de signes *absents* ou d'*information*, et donc en l'absence de signes pathologiques.

En reprenant notre exemple du patient présentant un *Syndrome de Raynaud* avec une *dystrophie capillaire*, sachant par ailleurs qu'il s'agit d'une femme, et qu'elle exprime l'antigène HLAB27 (les marqueurs génétiques sont aussi des signes de type *information*), le système lui affectera les résultats de la figure 3.6. On remarque sur cette figure que le total T_j n'est pas calculé pour toutes les pathologies pour lesquelles le patient ne présente aucun « *signe présent* ».

La standardisation du score permet de traiter de la même façon des pathologies définies par un grand nombre de signes et celles définies par un petit nombre d'entre eux.

Maladie	$\sum PW_{ij}$	$\sum AW_{ij}$	$\sum IW_{ij}$	T_j
Spondylarthrite Ankylosante	0	0	9	✗
Rhumatisme Psoriasique	0	0	9	✗
Arthrite Réactionnelle	0	0	10	✗
Polyarthrite Rhumatoïde	4	0	5	0.092
Lupus Systémique	4	0	7	0.109
Polymyosite	4	0	5	0.083
Dermatomyosite	4	0	5	0.071
Sclérodermie	6	0	5	0.096
Pseudo-Polyarthrite Rhizomélique	0	0	5	✗
Maladie de Horton	0	0	5	✗
Chondrocalcinose	0	0	5	✗
Goutte	0	0	4	✗
Maladie de Still	0	0	5	✗

FIG. 3.6: exemple de calcul de scores

Cette méthode est bien adaptée au diagnostic de classification, et tout particulièrement lorsque le niveau de définition des pathologies est très variable. En effet, le nombre de signes retenus pour décrire chaque pathologie peut doubler dans certains cas. Dans la figure 3.7 la partie centrale montre pour chaque pathologie le nombre de signes retenus dans sa définition.
La partie droite de la figure 3.7 et l'histogramme de la figure 3.8 montrent une comparaison des niveaux de définition obtenus pour les 13 pathologies de LADRI. On notera qu'une pathologie ayant des scores faibles ne doit pas être considérée comme mal définie mais seulement comme une pathologie n'ayant que peu de signes diagnostiques.

Le calcul et l'interprétation des scores est un bon reflet de la démarche du médecin quand il évalue le cas d'un patient. Chacun des signes présentés par ce patient est estimé pour chaque pathologie appartenant à la base de connaissances.
Le système de score donne un « avis » comparable pour toutes les pathologies car lorsque le praticien a valué les signes dans les définitions, il a pris en compte la même échelle de

	Nb. de signes retenus				Score Maximum			
Pathologies	Prés.	Abs.	Info.	**Total**	Prés.	Abs.	Info.	**Total**
Spondylarth. Ankylosante	17	3	7	27	81	21	39	141
Rhumatisme Psoriasique	17	3	7	27	73	22	40	135
Arthrite Réactionelle	17	3	6	26	69	21	31	121
PolyArth. Rhumatoïde	15	2	2	19	72	15	10	97
Lupus	18	1	2	21	87	7	12	106
Polymyosite	18	1	2	20	91	7	10	108
Dermatomyosite	21	1	2	24	109	7	10	126
Sclérodermie	20	0	2	22	104	0	10	114
Pseudo-Polyarth. Rhizom.	6	4	2	12	37	29	12	78
Maladie de Horton	11	3	2	16	67	29	12	108
Chondrocalcinose	16	4	2	22	87	28	12	127
Goutte	11	3	2	16	65	21	11	97
Maladie de Still	15	1	2	18	67	16	10	93

FIG. 3.7: nombre de signes et scores maximums des pathologies

valeurs pour toutes les maladies étudiées. L'interprétation des différents scores *présent, absent* et *information* permet de supprimer certaines pathologies de la liste des diagnostics possibles et d'ordonner les diagnostics restants. Le système de score fournit rarement un diagnostic absolu.

3.3.2 La confirmation du diagnostic : la méthode des critères

La méthode précédente est essentiellement un compte-rendu de la pratique du praticien, mais d'autres informations à propos des maladies, sont disponibles. Les *sociétés savantes médicales* publient des listes de critères pour établir le diagnostic de certaines pathologies ; par exemple, l'ACR pour la *polyarthtrite rhumatoïde* [5] ou de l'European Spondylarthropathy Study Group pour les *spondylarthropathies* [44].

Dans ces listes de critères, les valeurs de signes manipulées sont : majeur, mineur ou exclusion. L'opération utilisée est une règle sur le nombre de signes nécessaires pour valider le diagnostic. Cette règle est établie en fonction de la valeur des signes. Par exemple pour

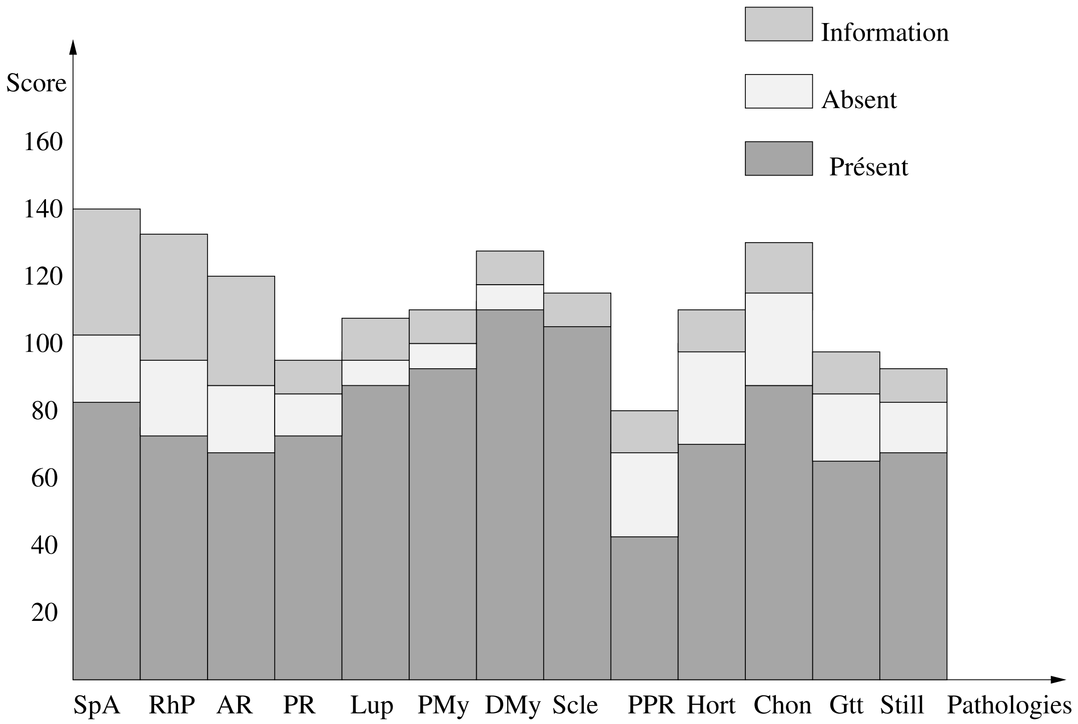

FIG. 3.8: histogrammes des différents scores pour chaque pathologie

confirmer la *maladie de Still* en utilisant les critères de Khan [79], 5 signes sont requis avec au moins 2 signes majeurs et aucun signe d'exclusion. La confirmation d'un diagnostic nécessite la validation totale de la règle correspondante. Si le patient ne satisfait que partiellement ces conditions, il n'est pas possible d'obtenir un avis « *partiel* ». Bien sûr, cette méthode de diagnostic permet d'éliminer les maladies pour lesquelles le patient présente un signe d'exclusion.

Pour illustrer la méthode, la figure 3.9 donne un exemple de liste, celui de la liste des critères de Khan [79] pour la *maladie de Still*.

Il faut toutefois relever certains problèmes qui expliquent notre choix de ne considérer cette méthode que comme un complément de l'expertise du praticien et non comme la base de notre système.

Critères majeurs :

1. Arthralgies depuis au moins 2 semaines
2. Rash malaire ou maculo-pulpeux
3. Fièvre sup. $\geq 39°C$
4. Hyperleucocytose $\geq 10000/mm^3$

Critères mineurs :

1. Pharyngite
2. Adénopathies
3. Anomalies du bilan hépatique
4. Latex Waaler Rose et facteurs anti-nucléaires négatifs

Critères d'exclusion :

1. Infection
2. Cancer
3. Affections rhumatologiques (PR)

Règle : 5 critères dont 2 majeurs et aucun d'exclusion

FIG. 3.9: la liste des critères diagnostiques de la *maladie de Still* [79]

Premièrement, on ne dispose pas systématiquement d'une liste de critères pour toutes les pathologies que l'on désire prendre en compte. Dans notre cas, trois pathologies sont regroupées sur une liste commune, celle des *spondylarthropathies*. Il s'agit de la *spondylarthrite ankylosante*, du *rhumatisme psoriasique* et de l'*arthrite réactionnelle*. Il n'est plus possible de les différencier par la suite. Trois autres pathologies n'ont pas fait l'objet de publication de listes de critères. Il s'agit de la *pseudo-polyarthrite rhizomélique*, de la *chondrocalcinose* et de la *goutte*. Cette absence interdit la construction par la méthode des critères, d'un diagnostic prenant en compte l'ensemble du domaine que nous étudions.

Deuxièmement, ces critères n'ont pas été construits dans le but d'établir des informations équivalentes d'une pathologie à l'autre. Il n'est pas possible de comparer les nombres de critères validés par plusieurs pathologies. Ainsi face à un patient présentant 3 critères en faveur du *lupus* et 2 en faveur de la *maladie de Still*, on ne peut préjuger d'un risque plus grand de *lupus vs maladie de Still*.

Toutefois, il reste important d'intégrer cette méthode dans un système de diagnostic. Ces listes de critères sont validées par les spécialistes du domaine et sont bien connues des praticiens. Elles sont des éléments de leur formation académique, et en ce sens sont acceptées de tous.

3.3.3 L'agrégation des résultats

Le diagnostic définitif est une combinaison de ces résultats [16]. Chacune des deux procédures de raisonnement construit une liste de pathologies : liste ordonnée pour le calcul de score, ensembles de pathologies confirmées ou exclues pour la méthode de critères.

- On commence par considérer la liste ordonnée fournie par le système de score. Chaque pathologie disposant d'au moins un signe *présent* observé chez le patient se voit affectée un rang dans cette liste. R_1 pour la pathologie en tête de liste, ... R_n pour la dernière. Le rang est inversement proportionnel au score : R_1 est associé au score le plus élevé et R_n au score le plus faible.
- Les maladies pour lesquelles le patient présente des signes d'exclusion sont retirées de cette liste.
- Si une ou plusieurs pathologies ont été validées par la méthode des critères :
  - On s'intéresse parmi elles à celle qui a le rang maximum dans la liste. On retire de cette liste toute pathologie de rang supérieur.
  - Parmi les pathologies encore présentes dans la liste, on retire celles qui ont une liste de critères et qui ne l'ont pas validée.
  - Le résultat est constitué de l'ensemble des pathologies restantes.
- Si aucune pathologie n'est validée par la méthode des critères, le résultat est la liste ordonnée des pathologies restantes.

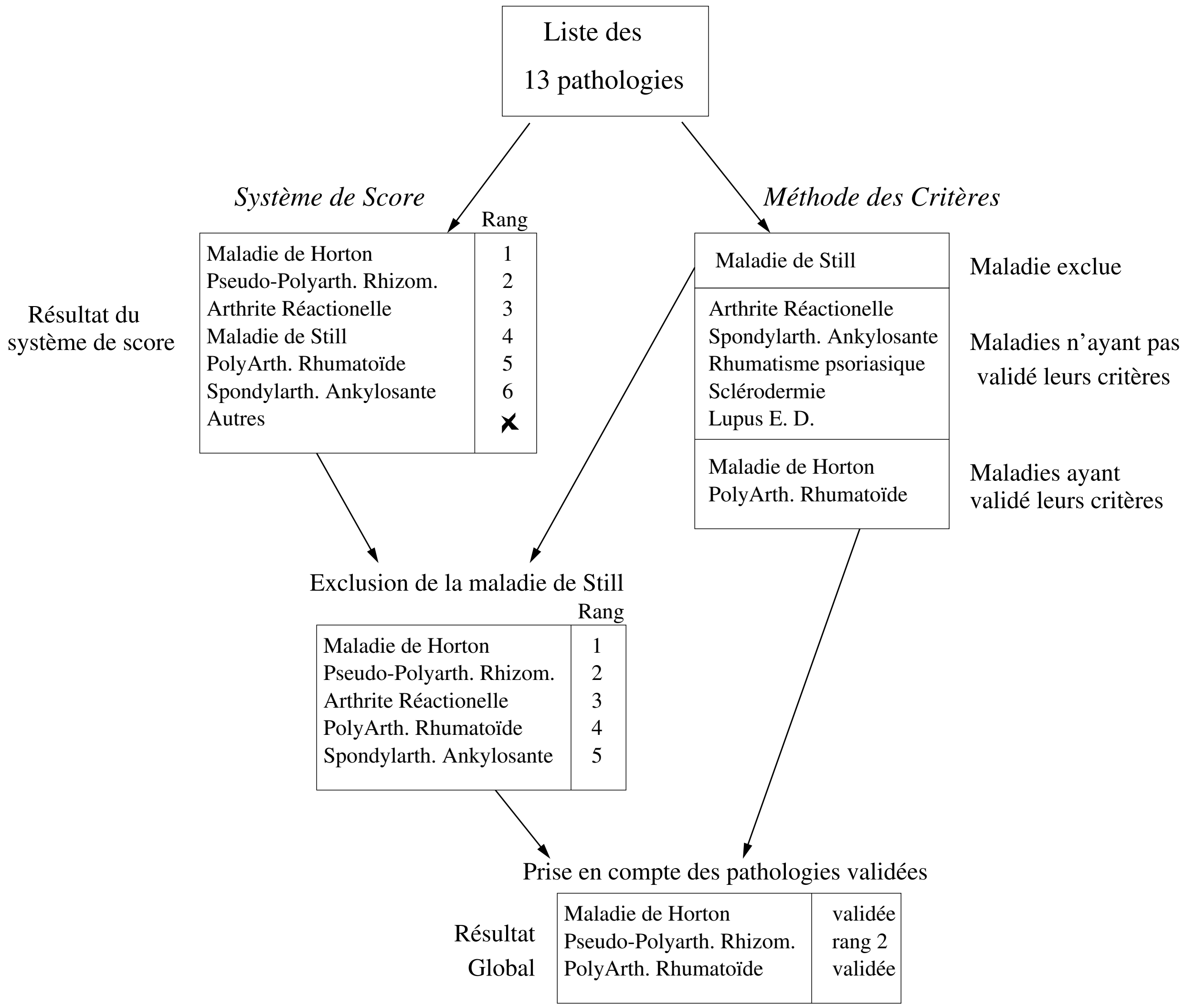

FIG. 3.10: évolution de liste des diagnostics possibles.

Cette forme d'agrégation accepte les conflits entre les deux méthodes de raisonnement. Si le système de score place en tête une pathologie qui ne valide pas sa liste de critères (mais qui possède une telle liste) et que par ailleurs, la méthode des critères valide une autre pathologie, c'est cette dernière qui (à tord ou à raison) sera retenue pour le diagnostic.

La figure 3.10 présente sur un exemple (tout à fait fictif) comment les différents résultats des méthodes de raisonnement sont agrégées pour fournir un résultat global.

3.4 Implémentation du système

LADRI a été implémenté avec le langage objet Java. Le choix d'un langage objet facilite la traduction des entités du modèle conceptuel en structure de données du programme informatique. De plus, ce langage offre un ensemble d'outils pour programmer les interfaces graphiques, le travail en mode client-serveur et l'accès au réseau internet. En effet, il nous parait très important de prévoir un environnement aussi ouvert que possible à ce type de logiciel. Un praticien doit pouvoir créer une base de patients « à demeure », c.-à-d. sur sa propre machine tout en accédant, s'il le désire, aux définitions de maladies sur un site distant.
Le langage Java permet aussi de résoudre les problèmes de disparité des systèmes d'exploitation, les environnements de développement des informaticiens étant le plus souvent très différents des environnements des utilisateurs. Dans ce domaine, le succès rapide du langage Java et de son concept de machine virtuelle (qui lui n'a rien de nouveau), laisse espérer une modification profonde des relations entre les informaticiens et les utilisateurs finals. Chacun pourra continuer à utiliser le système qui le satisfait tout en partageant les mêmes applications.

3.5 Tests du système

LADRI a été testé avec 107 dossiers de patients. Un résultat partiel (78 dossiers) a été présenté au 10^{e} congrès français de rhumatologie [18][137]. Tous les dossiers ont été choisis suivant les critères suivants :

- les diagnostics réels étaient disponibles (ceci afin de savoir si le système s'était ou non trompé) et il s'agissait dans tous les cas de patients atteints d'une des pathologies que nous étudions,
- les dossiers devaient concerner des patients au cours de leur première (ou d'une des toutes premières) consultation, c.-à-d. au moment où ils posent un véritable problème de diagnostic. Tester le système sur des patients connus du service depuis plusieurs années n'aurait eu que peu d'intérêt, car dans ce cas ils présentent tous les signes caractéristiques de la pathologie.

Alors qu'une maladie est en moyenne caractérisée par une vingtaine de signes (cf figure 3.7), la description d'un patient au stade du début de la maladie, est de l'ordre d'une douzaine de signes.

La figure 3.11 donne les résultats qui ont été obtenus lors des tests. La 1re colonne contient le nombre de cas pour chaque pathologie. La 3^{e} colonne montre que pour 89 dossiers (83%), le système de score a classé en tête de liste le diagnostic attendu. La 4^{e} colonne montre que 67 dossiers (62.6%) ont présenté un tableau clinique complet pour la pathologie attendue. La 5^{e} colonne montre que l'algorithme assurant l'agrégation des deux résultats a permis de classer correctement 97 dossiers (90.7%), le diagnostic attendu étant soit « *validé* », soit en position de tête de liste.

Remarquons qu'aucun patient n'a validé deux listes de critères. Parmi les 23 patients présentant une pathologie sans liste de critères (*pseudo-polyarthrite rhizomélique*, *chondrocalcinose* et *goutte*) pas un seul n'a validé une autre liste.
Un seul cas (un *rhumatisme psoriasique*) a validé à tord une liste de critères qui n'était pas la bonne (une *polyarthrite rhumatoïde*), il s'agissait d'un patient ayant à la fois une *at-*

Pathologies	*Nb de cas*	*Système de Score*	*Méthodes des Critères*	*Agrégation*
Spondylarth. Ankylosante	13	10	*pour les 24 cas* 21 validés	13
Rhumatisme Psoriasique	6	3		3
Arthrite Réactionelle	5	3		5
PolyArth. Rhumatoïde	30	29	25	29
Lupus	5	5	4	5
Polymyosite	1	1	1	1
Dermatomyosite	4	1	2	2
Sclérodermie	4	4	4	4
Pseudo-Polyarth. Rhizom.	13	13	missing	13
Maladie de Horton	10	8	9	9
Chondrocalcinose	8	4	missing	4
Goutte	6	6	missing	6
Maladie de Still	2	2	1	2
Total	107	89	67	97

FIG. 3.11: les diagnostics obtenus pour 107 patients

teinte polyarticulaire aux *poignets* et une *atteinte cutanée de psoriasis*. Cette « erreur » du système n'apparaît pas dans la figure 3.11 car un autre cas de *rhumatisme psoriasique* non classé par le premier système, a validé sa liste de critères.
Le choix de sélectionner des dossiers de patients débutant leur maladie et donc présentant des tableaux cliniques incomplets, explique les résultats relativement faibles de la méthode des critères.

La pathologie arrivant le plus souvent en bonne position dans la liste des scores est la *pseudo-polyarthrite rhumatoïde*. Ceci est dû au petit nombre de signes caractérisant cette maladie et à la standardisation des scores. Cinq des « *échecs* » du système de score proviennent d'un classement à tord de la *pseudo-polyarthrite rhumatoïde* en tête de liste.

✔ Interprétation des résultats

Le système s'adresse à des praticiens qui possèdent une connaissance générale du domaine. Notre objectif n'est pas de fournir un diagnostic absolu mais une *Aide* au diagnostic. Le praticien doit interpréter les résultats fournis par le système.

Dans les conditions ordinaires d'utilisation du système, le praticien ne dispose pas comme dans notre test, du diagnostic réel. S'il avait dû observer les 107 cas que nous avons étudiés, il aurait eu comme résultat :

- 68 dossiers avec un diagnostic portant la mention « *validé* », dont une erreur (le cas du *rhumatisme psoriasique*),
- 39 dossiers pour lesquels l'interprétation porte sur la liste des scores, parmi lesquels 30 ont le bon diagnostic en tête.

Le système réalise un diagnostic différentiel portant sur des pathologies dont certaines sont extrêmement proches. On considère généralement les 3 *spondylarthropathies* comme faisant partie d'un même groupe ; *polymyosite* et *dermatomyosite* se différencient essentiellement par une atteinte cutanée, *pseudo-polyarthrite rhizomélique* et *maladie de Horton* par l'atteinte de l'artère temporale, *chondrocalcinose* et *goutte* par une différence de localisation des atteintes articulaires. L'interprétation des résultats doit donc se faire dans ce contexte.

✔ Perspectives du système

LADRI a été réalisé dans le but de valider l'architecture proposée pour un système d'aide à la décision. Dans l'optique d'une diffusion de ce programme, des compléments s'avèrent indispensables.

Chaque signe appartenant aux dictionnaires doit être accompagné d'une définition décrivant l'intention de son utilisation. Il s'agit de préciser par des commentaires explicites, le sens du vocabulaire employé.

Le logiciel doit également être testé d'une part sur un nombre plus grand de dossiers, reflétant l'activité en routine d'un rhumatologue, et d'autre part avec plusieurs utilisateurs (internes, praticiens hospitaliers et médecins spécialistes libéraux).

Conclusion

Nous venons de présenter les étapes qui nous ont permis de développer un logiciel d'aide au diagnostic en rhumatologie inflammatoire. L'analyse des données et des traitements nous a conduit à définir une architecture d'objets répondant au problème de l'aide au diagnostic. La conception du système a introduit des nouveaux éléments dans cette architecture, ce sont des « *modèles de conception réutilisables* » dédiés aux processus informatiques. La réalisation du logiciel nous a permis de valider cette architecture à partir de tests réalisés sur 107 dossiers de patients.

Cette conclusion se propose de resituer notre travail dans le contexte de la problématique de l'aide à la décision présentée en introduction. Les trois points soulevés par l'analyse des systèmes passés étaient : l'accès des praticiens à l'informatique, les méthodes utilisées pour modéliser les tâches diagnostiques, les attentes des praticiens.

Accès à l'informatique.

Internet est devenu un outil de diffusion des logiciels qui permet de s'abstraire des systèmes d'exploitation. L'utilisation de langages de programmation comme Java sur des machines connectées en réseau permet une utilisation multi-centrique transparente. Des applications exécutées sur une machine serveur (quel que soit son système d'exploitation) peut répondre aux requêtes de machines clientes (aux systèmes d'exploitation eux aussi indifférents).

Notre Logiciel d'Aide au Diagnostic en Rhumatologie Inflammatoire a été écrit en Java. Nous avons testé des versions de LADRI en installant les descriptions des maladies sur une machine distante et celles des patients sur une machine locale, l'utilisation d'une telle configuration ne requiert aucune connaissance informatique de la part du praticien. A l'heure actuelle, chacun peut bénéficier des avancées technologiques ou méthodologiques sans devenir pour autant un spécialiste en informatique. De plus, l'utilisation des programmes à distance permet d'user des ressources mises à disposition par un serveur et n'impose plus de posséder soi-même des machines très puissantes.

Des exemples existent de diffusion de connaissances cliniques sur les patients à travers le réseau, mais pour l'instant ils concernent essentiellement les Etats-Unis [43] [81]. Ils amorcent la concrétisation de ce que Shortliffe en 1993 [125] souhaitait voir réaliser : le développement de nouveaux moyens de communication nationaux et internationaux pour échanger des informations médicales. Dans ce domaine, nous entrons dans une nouvelle ère.

Méthodes logicielles.

Par sa complexité et son caractère très appliqué, la médecine a toujours été un terrain de prédilection pour tester les recherches en Intelligence Artificielle. Des congrès, des revues sont spécifiquement dédiés à l'Intelligence Artificielle Médicale et pourtant les applications médicales touchent à tous les aspects de l'Intelligence Artificielle. A tel point qu'une discussion s'est ouverte dans la communauté scientifique française pour savoir s'il existe ou non une spécificité de l'Intelligence Artificielle en Médecine (cf le numéro 38 de juillet 1999 du bulletin de l'Association Française pour l'Intelligence Artificielle).

Le domaine de la décision médicale, du fait de la taille des systèmes actuels, pose à l'Intelligence Artificielle des problèmes très concrets d'ingénierie logicielle. De nombreux acteurs de ce domaine plaident pour l'introduction d'un plus grand professionnalisme dans le développement des applications informatiques [68] [97] [8].

Les méthodes statistiques ou logiques n'ont cessé d'évoluer. Elles permettent de manipuler toujours plus d'informations et génèrent elles-mêmes des quantités de plus en plus grande de données. La définition d'une architecture logicielle adéquate et une structuration efficace sont devenues des éléments essentiels de la réussite d'un projet informatique. Heathfield [68] a noté que souvent en médecine, l'implémentation d'une application informatique révèle des carences dans la cohérence de la conception et dans la stratégie de développement. Dans une discussion entamée dans la revue Methods of Information in Medicine, Musen [97] ajoute que les principes de base du Génie Logiciel ne sont pas encore appliqués de façon systématique dans le développement de systèmes en médecine clinique ; les travaux actuels sur la conception des programmes informatiques ne sont pas largement connus (et reconnus) dans la communauté de l'informatique médical.

Notre modèle d'architecture a pour but de mettre en évidence les caractéristiques essentielles d'un système d'aide au diagnostic en pratique clinique. Il contient des éléments issus de l'analyse sémantique du domaine et des éléments liés à l'implémentation du processus informatique.

Nous avons basé notre travail sur une approche objet, ce qui n'est pas une nouveauté en soi car les objets ont été « *inventés* » pour les méthodes d'analyse sémantique et de représentation des connaissances. De nombreux auteurs sont d'ailleurs arrivés à la conclusion que l'approche objet est une réponse au problème de la structuration des bases de connaissances médicales [67] [45] [98]. Si le modèle objet est considéré comme étant clair, maintenable et facile à comprendre lorsqu'il est traduit en langage objet, il faut constater que parfois cette vision idéalisée fait l'impasse sur la mise en pratique des *principes* orientés objet tels qu'ils sont appréhendés par le Génie Logiciel [76].

La réalisation d'un système objet dépasse la simple analyse objet du domaine de l'application. Nous avons vu qu'il était possible d'identifier des objets dont la présence n'est motivée que par le fait que nous construisons un système informatique. Ainsi les patterns `Composite` et `Itérateur` n'ont de sens que pour optimiser un processus algorithmique.

Par contre, le pattern `Etat` révélé par notre choix de faire partager la base de connaissances par plusieurs modules de traitement, fournit un nouveau mode d'appréhension du concept de valeur attachée au signe et intervient directement dans l'analyse sémantique du domaine. La présence du pattern `Stratégie` est essentiellement motivée par notre volonté d'intégrer plusieurs processus de raisonnement dans le même système d'aide au diagnostic. Il fournit un modèle générique de fonction de décision. Ce pattern est le garant de la modularité et de l'encapsulation des modules de raisonnement.

Des critiques à propos des patterns et des méthodes orientées objet évoquent une sur-modélisation entraînant un surcoût excessif de développement et une tendance à faire grossir de plus en plus le nombre de modules à implémenter. En fait, il existe bien un niveau supplémentaire d'abstraction avec `Etat`, `Itérateur` et `Stratégie` mais la quantité de codage effectif pour ces classes est faible et le surcoût est compensé par les bénéfices réalisés en maintenance. D'autre part, chaque pattern ne modélise qu'une seule tâche, ce qui semble bien correspondre aux attentes de Musen [99] lorsqu'il cherche à définir la *bonne taille* des éléments à partager et à réutiliser.

Bien que plusieurs ouvrages détaillent les méthodes d'analyse et de conception orientées objet, il n'y a pas de manuel précis sur le « *comment faire un logiciel* » [57]. Les catalogues de patterns tendent à combler cette lacune en restituant une expertise dans le domaine de l'ingénierie logicielle, à destination des concepteurs. D. Lea remarque qu'il n'existe pas de *program museums* où l'on puisse contempler des *chefs d'œuvres de la programmation* afin de s'en inspirer. Les catalogues de patterns sont comme ces musées où le travail des maîtres est montré. Plutôt que de formuler une théorie globale sur le « *comment faire* », les patterns suggèrent des exemples, aussi achevés que possibles, et les présentent « *prêts à l'emploi* » [86].
L'utilisation des patterns n'est pas aisée pour un novice en développement objet [32] et un effort est nécessaire pour se les approprier et découvrir quel pattern est le plus adapté au problème à traiter. Néanmoins, les patterns apportent une contribution essentielle à

la qualité des logiciels et à leur évolution.

Le Génie Logiciel cherche à limiter les charges de développement en construisant des logiciels dont les modules puissent être réutilisables. Ceci relève d'une démarche différente de celle de l'Intelligence Artificielle qui, pour le même objectif, utilise des moyens différents. Les chercheurs en Intelligence Artificielle travaillent sur la conception de modèles généraux du raisonnement humain. C'est dans cette optique qu'ont été mis au point les générateurs de systèmes experts c.-à-d. une organisation qui par son caractère universel pourrait répondre à toutes les attentes liées à une activité experte.
La réutilisabilité du développement se conçoit dans un cas comme la récupération de briques (les patterns) qui fournissent un service précis, à charge pour le développeur de la nouvelle application de refaire un assemblage qui lui convienne. Dans l'autre cas, on propose un système qui s'adapte à n'importe quelle application par l'ajout des connaissances expertes décrivant cette application.
Si l'Intelligence Artificielle est encore aussi réticente à intégrer les principes du Génie Logiciel, c'est qu'elle n'en ressent pas encore les besoins. Ces principes ont été établis par des développeurs qui « au quotidien », ont découvert les fondements de la réutilisabilité : écrire quelques dizaines de lignes de code remplissant une tâche bien spécifique et pouvant être appelées depuis n'importe quelle application.

L'activité qui consiste à réassembler des briques logicielles correspond à ce qu'Alexander a défini sous le concept de « Langage de Patterns ». Un langage de patterns à la capacité de définir la construction d'un type d'applications à l'aide d'un groupe de patterns collaborants [34] [138] [92]. Notre travail s'inscrit dans cette démarche en proposant l'architecture d'un type de système d'aide au diagnostic à partir d'un ensemble de patterns.

La demande des praticiens.

En ce qui concerne l'automatisation du diagnostic, la demande des praticiens n'a pas véritablement évolué mais certains signes montrent que ceci pourrait changer rapidement.

L'informatisation des praticiens, le développement des réseaux et surtout la production des données par les systèmes informatiques aboutissent à une véritable *avalanche de données*. Puisque ce sont les machines qui génèrent actuellement les informations, il est normal que ce soient également des machines qui assistent les individus dans le traitement de ces données. Le changement d'échelle dans la quantité des informations à gérer, passant d'un ordre de grandeur humain, à un ordre de grandeur gigantesque (chiffré en Giga-octets) est un processus irréversible.

En fournissant toujours plus d'informations sans donner aux utilisateurs les moyens de les structurer, on court le risque de créer les *cimetières de données* évoqués par F. Gremy [61]. Les travaux actuels dans le domaine de la *fouille de données* analysent les voies de réponse à ce problème. Ils établissent des outils de recherche et des structures de données destinés à alimenter des traitements automatiques.

Nous avons défini une manière d'organiser des données pour contribuer à l'établissement d'un diagnostic. L'objectif n'est pas de singer le raisonnement du praticien mais de créer une collaboration avec l'utilisateur [35].

Cette collaboration n'est possible que si l'utilisateur comprend la finalité des traitements. Le système doit apporter une aide sur la quantité des données à traiter et non sur la qualité des raisonnements impliqués. Le système doit également afficher clairement ses limites.
J.C. Courbon [35] raconte son expérience quant à la réalisation d'un programme de gestion de planning hebdomadaire du personnel infirmier. Il explique que ce programme ne savait pas résoudre certains conflits d'emploi du temps, laissant l'infirmière chef compléter *à la main* les quelques cas problématiques. Obtenir un système qui fonctionne à 100% aurait nécessité beaucoup d'efforts pour un faible accroissement de la valeur ajoutée. De plus cette intervention de l'utilisateur a été ressentie comme une valorisation de son rôle. Il conclue par cette remarque :

il n'est pas inconcevable de générer des solutions incomplètes voire qui ne satisfont pas les contraintes (solution impossible). Laissons ensuite les utilisateurs grâce à l'interactivité faire le dernier bout de chemin qui va rendre la solution réalisable ou complète.

Notre système adopte ce point de vue. Il fournit un diagnostic confirmé quand une liste de critères est validée, sinon il affiche une liste de diagnostics possibles attendant que l'utilisateur cherche à compléter les informations recueillies sur son patient. Son véritable rôle est de prendre en charge en parallèle l'examen d'un nombre plus grand de pathologies que ne peut le faire le praticien.

Annexe A

Description des pathologies

A.1 Spondylarthrite Ankylosante

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		3	rarement
		oligoarticulaire		4	parfois
		monoarticulaire		4	parfois
		Mot clé : localisation articulaire			
		rachis cervical		4	parfois
		rachis dorsal		4	parfois
		sacro-iliaque		7	généralement
		asymétrique		5	souvent
destruction articulaire	présent	*Mot clé : localisation articulaire*			
		sacro-illiaque		7	généralement
		rachis dorsale		4	parfois
		lombaire		6	le plus souvent
		cervicale		5	souvent
		périphériques		5	souvent
inflammation articulaire	présent	*Mot clé : localisation articulaire*			
		orteils		3	rarement
inflammation	présent	*Mot clé : localisation*			
		oeil-uvétite		5	souvent
		enthèses		7	généralement
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	7	généralement
		Vit. Sédimen. sup.	100	4	parfois
		CRP inf	20	7	généralement
		CRP sup	20	4	parfois
facteur rhumatoïde	absent			7	généralement
réaction Immunitaire	absent			7	généralement
atteinte systémique	absent			7	généralement
altération état gén.	présent	*Mot clé : température*			
		borne sup. : $39°C$ le soir		2	trés rarement
réaction au traitement	présent	*Mot clé : classe thérapeutique*			
		AINS		7	généralement

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		7	généralement
		Femme	10 %	2	trés rarement
		Mot clé : Age			
		âge pic	30 ans	5	souvent
marqueur génétique	information	*Mot clé : type HLA*			
		B27		7	généralement
antécédent personnel	information	*Mot clé : pathologie*			
		spondylarthropathie		4	parfois
		psoriasis		4	parfois
		uvéite		5	souvent
antécédent Familial	information	*Mot clé : pathologie*			
		spondylarthropathie		7	généralement

A.2 Rhumatisme Psoriasique

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		4	parfois
		oligoarticulaire		4	parfois
		monoarticulaire		3	rarement
		Mot clé : localisation articulaire			
		interphalangienne dist.		4	parfois-spécif.
		Sacro-iliaque		4	parfois
		asymétrique		5	souvent
inflammation articulaire	présent	*Mot clé : localisation articulaire*			
		orteils		4	parfois
inflammation	présent	*Mot clé : localisation*			
		enthèses		7	généralement
		oeil (uvétite)		4	parfois
destruction Articulaire	présent	*Mot clé : localisation articulaire*			
		sacro-illiaque		4	parfois
		rachis dorsale		4	parfois
		lombaire		4	parfois
		rachis cervicale		4	parfois
		périphérique		4	parfois
atteinte cutanée	présent	*Mot clé : forme d'att. cutanée*			
		psoriasis		7	généralement
atteinte systémique	absent			7	généralement
facteur rhumatoïde	absent			8	trés général.
	présent				fortement négatif
réaction immunitaire	absent			7	généralement
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	6	le plus souvent
		Vit. Sédimen. sup	100	4	parfois
		CRP inf	20	5	souvent
		CRP sup	20	5	souvent
altération état gén.	présent	*Mot clé : température*			
		borne sup. : 39°C le soir		3	rarement
réaction au traitement	présent	*Mot clé : classe thérapeutique*			
		AINS		7	généralement

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		5	souvent
		Femme		4	parfois
		Mot clé : Age			
		bornes : 30 - 45 ans		5	souvent
marqueur génétique	information	*Mot clé : Type HLA*			
		B27		5	souvent
antécédent Personnel	information	*Mot clé : Pathologie*			
		spondylarthropathie		7	généralement
		psoriasis		7	généralement
		uvéite		4	parfois
antécédent Familial	information	*Mot clé : Pathologie*			
		spondylarthropathie		7	généralement

A.3 Arthrite Réactionnelle

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur articulaire	présent	*Mot clé :cardinalité*			
		polyarticulaire		4	parfois
		oligoarticulaire		5	souvent
		monoarticulaire		3	rarement
		Mot clé :latéralisation			
		asymétrique		5	souvent
		Mot clé : localisation articulaire			
		interphalangienne dist.		4	parfois
		Sacro-iliaque		4	parfois
inflammation articulaire	présent	*Mot clé :localisation articulaire*			
		orteils		5	souvent
inflammation	présent	*Mot clé : localisation*			
		oeil-uvétite		4	parfois
		enthèses		4	parfois
destruction articulaire	présent	*Mot clé : localisation articulaire*			
		sacro-illiaque		4	parfois
		rachis dorsale		4	parfois
		Lombaire		4	parfois
		rachis cervicale		4	parfois
		périphérique		4	parfois
atteinte cutanée	présent	*Mot clé : forme d'att. cutanée*			
		psoriasis		4	parfois
atteinte systémique	absent			7	généralement
facteur rhumatoïde	absent			7	généralement
réaction immunitaire	absent			7	généralement
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	5	souvent
		Vit. Sédimen. sup	100	6	le plus souvent
		CRP inf	20	5	souvent
		CRP sup	20	6	le plus souvent
altération état gén.	présent	*Mot clé : température*			
		borne sup. : $39°C$ le soir		4	parfois
réaction au traitement	présent	*Mot clé : classe thérapeutique*			
		AINS		6	le plus souvent

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		5	souvent
		Femme		5	souvent
		Mot clé : âge			
		bornes : 20 - 50 ans		5	souvent
Marqueur génétique	information				
		Mot clé : type HLA			
		B27		5	souvent
antécédent Personnel	information	*Mot clé : pathologie*			
		épisode infectieux		8	trés général.
antécédent Familial	information	*Mot clé : pathologie*			
		Spondylarthropathie		4	parfois
		Psoriasis		4	parfois

A.4 Polyarthrite Rhumatoïde

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		7	généralement
		oligoarticulaire		4	parfois
		monoarticulaire		3	rarement
		Mot clé : localisation articulaire			
		cou - rachis cervical		5	souvent
		symétrique		7	généralement
destruction Articulaire	présent	*Mot clé : localisation articulaire*			
		périphérique		7	généralement
		rachis cervicale		4	parfois
	absent				
		Mot clé : localisation articulaire			
		Sacro-iliaque		8	trés généralement
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	7	généralement
		Vit. Sédimen. sup.	100	4	parfois
		CRP inf	20	7	généralement
		CRP sup	20	4	parfois
synd. Raynaud	présent	*Mot clé : capillaroscopie*			
		dystrophie capill.		4	parfois
atteinte systémique	présent	*Mot clé :localisation systémique*			
		nodule rhumatoïde		7	généralement
		pleurale		3	rarement
		pulmonaire		3	rarement
		cardiaque		3	rarement
atteinte cutanée	absent	*Mot clé : forme d'att. cutanée*			
		psoriasis		7	généralement
facteur rhumatoïde	présent			6	le plus souvent
réaction immunitaire	présent	*Mot clé : marqueurs de l'immunité*			
		anti-noyau $> 1/250$		4	parfois
altération état gén.	présent	*Mot clé : température*			
		borne sup. : $39°C$ le soir		4	parfois

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		4	parfois
		Femme		5	souvent
		Mot clé : Age			
		bornes : 30 - 50 ans		5	souvent

A.5 Lupus Erythémateux Disséminé

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		5	souvent
		oligoarticulaire		4	parfois
		monoarticulaire		2	rarement
		Mot clé : latéralisation			
		symétrique		6	le plus souvent
destruction Articulaire	absent			7	généralement
douleur musculaire	présent			3	rarement
atteinte cutanée	présent				
		Mot clé : forme d'att. cutanée			
		masque lupique		8	très généralement
atteinte systémique	présent	*Mot clé : manifestation systémique*			
		rénale	ou 70 - 80%	5	souvent
		pleurale		4	parfois
		pulmonaire		3	rarement
		cardiaque		4	parfois
		neuro-psy		3	rarement
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	7	généralement
		Vit. Sédimen. sup.	100	4	parfois
		CRP inf	20	6	généralement
		CRP sup	20	4	parfois
synd. Raynaud	présent	*Mot clé : capillaroscopie*			
		dystrophie capill.		4	parfois
facteur rhumatoïde	présent			4	parfois
réaction Immunitaire	présent	*Mot clé : marqueurs de l'immunité*			
		anti-noyau $> 1/250$		7	généralement
		anti-noyau ECTSm		8	trés général.
		anti-noyau SSA		4	parfois
		anti-ADN > 10		8	trés général.
altération état gén.	présent	*Mot clé : température*			
		borne sup. : $39°C$ le soir		4	parfois

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		2	trés rarement
		Femme		7	généralement
		Mot clé : Age			
		bornes : 20 - 50 ans		5	souvent

A.6 Polymyosite et Dermatomyosite

Nom de l'événement	Etat	Nom de l'attribut	Val. N.	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		4	parfois
		Mot clé : latéralisation			
		asymétrique		4	parfois
		symétrique		5	souvent
douleur musculaire	présent			7	généralement
destruction musculaire	présent			7	généralement
destruction articulaire	absent			7	généralement
synd. Raynaud	présent	*Mot clé : capillaroscopie*			
		dystrophie capill.		4	parfois
atteinte systémique	présent	*Mot clé : type*			
		nodule rhumatoïde		4	parfois
		Mot clé : localisation systémique			
		pulmonaire		4	parfois
		cardiaque		3	rarement
facteur rhumatoïde	présent			4	parfois
réaction Immunitaire	présent	*Mot clé : marqueurs de l'immunité*			
		anti-noyau $> 1/250$		7	généralement
		anti-noyau ECTPm		7	généralement
		anti-noyau ECTJo1		7	généralement
atteinte Biochimique	présent	*Mot clé : marqueurs biochimiques*			
		CPK (seuil sup. 175)		7	généralement
		CPK + LDH (seuil sup. 250) = +1			
		Aldolass (seuil sup. 8)		7	généralement
		Aldo. + Tgo (seuil sup. 50) = +1			
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	6	le plus souvent
		Vit. Sédimen. sup.	100	4	parfois
		CRP inf	20	6	le plus souvent
		CRP sup	20	4	parfois
altération état gén.	présent	*Mot clé : température*			
		borne sup. : $39°C$ le soir		4	parfois

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		5	souvent
		Femme		5	souvent
		Mot clé : Age			
		bornes : 20 - 50 ans		5	souvent

A.6.1 Spécificité de la Dermatomyosite

Nom de l'événement	Etat	Nom de l'attribut	Val. N.	Valeur	
att. cutan. dermatom.	présent	*Mot clé : forme d'att. cutanée*			
		érythèmes violacés ou oedèmes		8	trés général.
		Mot clé : localisation			
		paupières		7	généralement
		face		6	le plus souvent
		main		6	le plus souvent

A.7 Sclérodermie

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		5	souvent
		oligoarticulaire		4	parfois
		monoarticulaire		3	rarement
		Mot clé : latéralisation			
		asymétrique		4	parfois
		symétrique		5	souvent
destruction osseuse	présent	*Mot clé : localisation*			
		périphérique		4	parfois
douleur musculaire	présent			4	parfois
atteinte cutanée	présent	*Mot clé : forme d'att. cutanée*			
		sclérose diffuse		8	trés général.
		sclérodactylie		8	trés général.
synd. Raynaud	présent	*Mot clé : capillaroscopie*			
		dystrophie capill.		6	le plus souvent
		Mega-capillaire		8	trés général.
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	6	le plus souvent
		Vit. Sédimen. sup.	100	4	parfois
		CRP inf	20	6	le plus souvent
		CRP sup	20	4	parfois
Facteur rhumatoïde	présent			4	parfois
Réaction Immunitaire	présent	*Mot clé : marqueurs de l'immunité*			
		anti-noyau > 1/250		7	généralement
		anti-noyau Scl70		8	trés général.
		anti-centromères > 0		7	généralement
atteinte systémique	présent	*Mot clé : manifestations systémiques*			
		nodule rhumatoïde		4	parfois
		cardiaque		4	parfois
		pulmonaire		5	souvent
		rénale		4	parfois
		gastro-intestinale		5	souvent
altération état gén.	présent	*Mot clé : température*			
		borne sup. : 39°C le soir		4	parfois

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		4	parfois
		Femme		5	souvent
		Mot clé : Age			
		bornes : 20 - 50 ans		5	souvent

A.8 Pseudo-Polyarthrite Rhizomélique

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur Rhizomélique	présent			7	généralement
douleur musculaire	présent	*Mot clé : localisation musculaire*			
		fesses, épaules,cuisses		5	souvent
destruction Articulaire	absent			8	trés général.
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	7	généralement
		Vit. Sédimen. sup.	100	4	parfois
		CRP inf	20	7	généralement
		CRP sup	20	4	parfois
facteur rhumatoïde	absent			7	généralement
réaction Immunitaire	absent			7	généralement
atteinte systémique	absent			7	généralement
altération état gén.	présent	*Mot clé : température*			
		borne sup. : 39°C le soir		4	parfois
réaction au traitement	présent	*Mot clé : classe thérapeutique*			
		corticoïde		7	généralement

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		4	parfois
		Femme		5	souvent
		Mot clé : Age			
		bornes inf : 60 ans		7	généralement

A.9 Maladie de Horton

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur céphalique	présent	*Mot clé : localisation céphalique*			
		artère temporale		7	généralement
		scalp		5	souvent
		claudication ORL		5	souvent
		céphalées		5	souvent
douleur Rhizomélique	présent			7	généralement
douleur musculaire	présent	*Mot clé : localisation musculaire*			
		fesses, épaules,cuisses		5	souvent
destruction articulaire	absent			8	trés général.
inflammation des vaisseaux	présent	*Mot clé : localisation*			
		artère temporale		8	trés général.
atteinte systémique	absent			7	généralement
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	4	parfois
		Vit. Sédimen. sup.	100	7	généralement
		CRP inf	20	4	parfois
		CRP sup	20	7	généralement
altération état gén.	présent	*Mot clé : température*			
		borne sup. : 39°C le soir		4	parfois
réaction immunitaire	absent			7	généralement
facteur rhumatoïde	absent			7	généralement
réaction au traitement	présent				
		Mot clé : classe thérapeutique			
		corticoïde		7	généralement

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		5	souvent
		Femme		5	ouvent
		Mot clé : Age			
		bornes inf. : 70 ans		7	généralement

A.10 Chondrocalcinose

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		3	rarement
		oligoarticulaire		4	parfois
		monoarticulaire		7	généralement
		Mot clé : localisation articulaire			
		poignet		6	le plus souvent
		genou		6	le plus souvent
		cheville		3	rarement
		InterPhalangienne Distale		4	parfois
		Mot clé : fréquence			
		continu		2	trés rarement
		crise		7	généralement
	absent	gros orteil		7	généralement
destruction articulaire	présent	*Mot clé : localisation articulaire*			
		périphérique		4	parfois
Calcification Articulaire	présent	*Mot clé : localisation articulaire*			
		symphyse pubienne		6	le plus souvent
		poignet		6	le plus souvent
		genou		6	le plus souvent
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	5	souvent
		Vit. Sédimen. sup.	100	6	le plus souvent
		CRP inf	20	5	souvent
		CRP sup	20	6	le plus souvent
atteinte systémique	absent			7	généralement
Réaction Immunitaire	absent			7	généralement
Facteur rhumatoïde	absent			7	généralement
altération état gén.	présent	*Mot clé : température*			
		borne sup. : 39°C le soir		4	parfois
Réaction au traitement	présent	*Mot clé : classe thérapeutique*			
		AINS		7	généralement
		Colchicine		7	généralement
		Corticoïde		6	le plus souvent

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		5	souvent
		Femme		5	souvent
		Mot clé : Age			
		bornes inf. :70 ans		7	généralement

A.11 Goutte

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		2	trés rarement
		oligoarticulaire		4	parfois
		monoarticulaire		7	généralement
		Mot clé : localisation articulaire			
		gros orteil		7	généralement
		Mot clé : fréquence			
		continu		2	trés rarement
		en crise		7	généralement
destruction Articulaire	présent	*Mot clé : localisation Articulaire*			
		périphérique		4	parfois
tophus	présent			7	généralement
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	4	parfois
		Vit. Sédimen. sup.	100	7	généralement
		CRP inf	20	4	parfois
		CRP sup	20	7	généralement
facteur rhumatoïde	absent			7	généralement
réaction Immunitaire	absent			7	généralement
atteinte Biochimique	présent	*Mot clé : marqueurs biochimiques*			
		acide urique		5	souvent
altération état gén.	présent	*Mot clé : température*			
		borne sup. : 39°C le soir		4	parfois
atteinte systémique	absent			7	généralement
réaction au traitement	présent	*Mot clé : classe thérapeutique*			
		AINS		7	généralement
		colchicine		7	généralement

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		5	souvent
		Femme		4	parfois
		Mot clé : Age			
		bornes inf. : 50 ans		6	le plus souvent

A.12 Maladie de Still

Nom de l'événement	Etat	Nom de l'attribut	Val. N.	Valeur	
douleur articulaire	présent	*Mot clé : cardinalité*			
		polyarticulaire		4	parfois
		oligoarticulaire		3	rarement
		monoarticulaire		2	rarement
		Mot clé : localisation articulaire			
		interphalangienne distale		4	parfois
		asymétrique		4	parfois
		symétrique		5	souvent
destruction articulaire	présent	*Mot clé : localisation articulaire*			
		périphérique		4	parfois
		cervicale		4	parfois
douleur musculaire	présent			4	parfois
atteinte cutanée	présent	*Mot clé : forme d'att. cutanée*			
		macules rosées/urticaire		7	généralement
synd. infl. biol.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		Vit. Sédimen. bornes	20 - 100	5	souvent
		Vit. Sédimen. sup.	100	5	souvent
		CRP inf	20	5	souvent
		CRP sup	20	5	souvent
altération état gén.	présent	*Mot clé : température*			
		borne sup. : 39°C le soir		7	généralement
atteinte hémato.	présent	*Mot clé : marqueurs biologiques de l'inflammation*			
		taux Glob. blancs augment.sup 12 000		7	généralement
		taux ferritine augment. sup 2 000		7	généralement
facteur rhumatoïde	absent			8	trés général.
atteinte systémique	présent	*Mot clé : localisation systémique*			
		pleurale		4	parfois
		cardiaque		4	parfois
réaction Immunitaire	absent			8	trés général.

Nom de l'événement	Etat	Nom de l'attribut	Mesure	Valeur	
Identité	information	*Mot clé : Sexe*			
		Homme		5	souvent
		Femme		5	souvent
		Mot clé : Age			
		bornes : 20 - 50 ans		5	souvent

Annexe B

Liste des critères utilisés

Liste des critères issus de la brochure de I. Amoura et P. Bourgeois [7] pour les 8 pathologies de la base de connaissances.

B.1 Spondylarthropathies

Pour les spondylarthropathies, 2 listes sont utilisées La première a été proposée par B. Amor en 1989[6], la deuxième par l'European Spondylarthropathy Group en 1991[44].

B.1.1 Critères d'Amor

Si la somme des points des 12 critères est égale ou supérieure à 6 le diagnostic est validé avec une sensibilité de 90% et une spécificité de 86,6%.

Signes cliniques ou histoire clinique	Nb Pts
1. douleurs nocturnes lombaires ou dorsales et/ou raideur matinale lombaire et/ou dorsale	1
2. Oligoarthrite asymétrique	2
3. douleurs fessières	
• uni ou bilatérales	1
• à bascule	2
4. Doigt ou orteil en saucisse	2
5. Tatalgie ou tout autre enthésiopathie	2
6. Iritis	2
7. Urétrite non gonococcique ou cervicite moins d'un mois avant le début de l'arthrite	1
8. Diarrhée de moins d'un mois avant l'arthrite	1
9. Présence ou antécédent de Psoriasis et/ou d'une balanite et/ou d'une entérocolopathie chronique	2
Signes Radiologiques	
10. Sacro-illite ≤ 2 si bilatérale ou ≥ 3 si unilatérale)	3

Terrain Génétique

11. Présence de l'antigène B27 et/ou antécédents familiaux
de pelvispondylite et/ou syndrome de Reiter et/ou de Psoriasis
et/ou d'uvéite et/ou d'entérocolopathie chronique 2

Sensibilité au Traitement

12. Amélioration en 48 heures des douleurs par les AINS
et/ou rechute rapide (48 heures) des douleurs à leur arrêt 2

B.1.2 Critères de l'ESG

Si le malade présente au moins 1 critère majeur et 1 critère mineur le diagnostic est établi avec une sensibilité et une spécificité de 87%.

Critères Majeurs

1. Rachialgies
2. Synovite : asymétrique ou prédominante aux membres inférieurs

Critères Mineurs

1. histoire familiale de spondylarthropathie ou d'uvéite ou d'entérocolopathie,
2. Psoriasis,
3. entérocolopathie inflammatoire,
4. urétrite, cervicite ou diarrhée aiguë de moins d'un mois avant l'arthrite,
5. enthésiopathie,
6. sacro-illite radiologique .

B.2 Polyarthrite Rhumatoïde

Critères de diagnostic de la polyarthrite rhumatoïde proposés par l'ACR[5]. Quatre critères sur sept sont nécessaires pour porter le diagnostic de la Polyarthrite Rhumatoïde avec une sensibilité de 91% et une spécificité de 89%. Pour les critères 1 à 4, les signes ou symptômes articulaires doivent avoir été constants pendant au moins 6 semaines.

1. Raideur matinale :
raideur matinale au niveau et autour des articulations pendant une heure avant l'amélioration maximale.
2. Arthrite d'au moins 3 articulations :
gonflement simultané d'au moins 3 articulations constatés par un médecin et dû à une hypertrophie des tissus mous ou à un épanchement articulaire (et non pas à une saillie osseuse). 14 articulations ou groupes d'articulations sont à prendre en compte : IPP - MCP - poignets - coudes - genoux - chevilles - MTP.
3. Arthrite des mains :
gonflement d'au moins un des groupes articulaires suivants : poignets - MCP - IPP.

4. Arthrite symétrique :
atteinte simultanée et bi-latérale des articulations ou groupes d'articulations définies en 2. L'atteinte bi-latérale sans symétrie exacte des IPP, MCP, MTP est admise dans le critère.
5. Nodules rhumatoïdes :
nodosité sous-cutanés constatées par un médecin sur des crêtes osseuse ou des surfaces d'extension ou en situation péri-articulaire.
6. Présence de facteur rhumatoïde :
7. Signes radiologiques : anomalies radiologiques typiques de la polyarthrite rhumatoïde sur des clichés des mains et des poignets avec érosions osseuse et déminéralisation en bande.

B.3 Lupus Erythémateux Disséminé

Critères de classification proposés par l'ACR [103] 4 critères sont nécessaires pour établir un diagnostic. Les informations sur la sensibilité et la spécificité ne sont pas données.

- Eruption malaire.
- Lupus discoïde.
- Photosensibilité.
- Ulcérations buccales ou nasopharyngées.
- Polyarthrite non érosive.
- Pleurésie ou péricardite.
- atteinte rénale : protéinurie $> 0.50g/24h$ ou cylindres urinaires.
- atteinte neurologique : convulsions ou psychose.
- atteinte hémolytique ou leucopénie ($< 4000/\mu l$) ou lymphopénie ($1500/\mu l$) ou thrombopénie ($100000/\mu l$).
- Anomalie immunologique caractéristique : cellule LE ou anticorps anti-DNA ou anticorps anti-SM ou sérologie syphilitique faussement positive.
- Présence d'un taux élevé de facteurs antinucléaires.

B.4 Polymyosite et Dermatomyosite

Critères de classification proposés par Tanimoto[129]
Pour la polymyosite, 4 critères de 2 à 9 sont nécessaires le nombre de critères nécessaires pour que le diagnostic soit validé avec une sensibilité de 98,9% et une spécificité de 95,2%.
Pour la dermatomyosite il faut constater en plus de ces 4 critères au moins un critère cutanée. La sensibilité sera alors de 94% et la spécificité de 90,3%.

1. Lésions cutanées :
   (a) Eruption héliothrope : érythème œdémateux rouge violacé des paupières.

   (b) Signe de Gottron : macules ou érythème atrophique rouges violacés kératosiques sur les tendons extenseurs des articualtions des doigts.
   (c) Erythèmes de la face d'extension des articulations des membres : érythèmes rouges violacés légèrement surélevés sur les coudes ou les genoux.
2. Faiblesse musculaire proximale des memebres supérieurs et inférieurs et du tronc.
3. Elévation du taux sérique des CPK (créatine phosphonikase) ou aldolases.
4. douleur musculaire à la pression ou douleur spontanée.
5. atteinte myogène à l'électromyogramme : potentiel polyphasiques myogènes de courte durée avec potentiels de fibrillation spontanés.
6. Anticorps anti-jo 1 positifs (histadyl tRNA synthétase).
7. Arthralgies ou arthrites non-destructrices.
8. Signes inflammatoires systèmiques : fièvre supérieure à 37° prise au creux axillaire, élévation sérique du taux de CRP ou VS supérieure à $20mm/h$ par la méthode de Westergren.
9. Examens anatomo-pathologiques compatibles avec le diagnostic de myosite inflammatoire : infiltration inflammatoire des muscles squelettiques avec dégénérescence ou nécrose des fibres musculaires.
   Phagocytose active, noyaux centraux ou évidence de régénération active.

B.5 Sclérodermie

Critères de classification proposés par l'ACR[93]. 1 critère majeur ou 2 mineurs sont nécessaires pour établir un diagnostic. Les informations sur la sensibilité et la spécificité ne sont pas données.

Critères Majeurs

1. Sclérodermie proximale :
   Modification sclérodermique typique de la peau (tendue, épaissie, indurée, ne prenant pas le godet), touchant la face, le cou, le tronc, ou la partie proximale des membres supérieurs ou inférieurs.

Critères Mineurs

1. Sclérodactylie.
2. Cicatrice déprimée d'un doigt ou ulcération de l'extrémité d'un doigt.
3. Fibrose pulmonaire des bases.

B.6 Maladie de Horton

Critères de classification proposés par l'ACR[74].
3 critères sont nécessaires pour que le diagnostic soit validé avec une sensibilité de 93,5% et une spécificité de 91,2%.

1. Début de la maladie aprés 50 ans.

2. Céphalées récentes.
3. Sensibilité ou diminution des battements d'une artère temporale.
4. VS $> 50mm$
5. Biopsie d'artère temporale montrant des lésions de vascularite avec infiltrats lymphocitaires ou à polynucléaires et habituellement présence de cellules géantes.

B.7 Maladie de Still de l'adulte

Critères de classification proposés par Kahn [79] et Yamaguchi [142]. 5 critères dont au moins 2 majeurs (et aucun d'exclusion) sont nécessaires pour établir un diagnostic avec une sensibilité de 96,2% et une spécificité de 92,1%.

Critères Majeurs

1. Fièvre supérieure ou égale à $39°C$ le soir depuis au moins une semaine.
2. Arthralgies depuis au moins 2 semaines.
3. Rash maculaire ou maculo-papuleux rosé non prurugineux habituellement pendant les épisodes fébriles.
4. Hyperleucocytose supérieure ou égale à $10000/mm^3$ avec au moins 80% de polynucléaires neutrophiles.

Critères Mineurs

1. pharyngite.
2. Adénopathies d'apparition récente, et/ou splénomégalie.
3. Anomaies hépatique : transaminases et/ou LDH élevées, en l'absence d'autre cause (toxique, médicamenteuse, allergique).
4. Latex Waaler Rose et facteurs anti-nucléaires négatifs.

Critères d'Exclusion

1. Infection (notamment septicémie, mononucléose infectieuse).
2. Cancer (dont lymphome).
3. Affections rhumatologiques (notamment Peri-arthrite noueuse, polyarthrite rhumatoïde avec signes extra-articulaire).

Bibliographie

[1] R.J. Abbott. Program design by informal english descriptions. *Comm ACM*, 26(11):882–894, 1983.

[2] A. Aho, R. Sethi, and J. Ullman. *Compilateurs Principes, techniques et outils.* InterEditions, 1989.

[3] C. Alexander. *Timeless way of building.* Oxford University Press, 1979.

[4] C. Alexander, A. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. *A Pattern Language.* Oxford University Press, 1977.

[5] American College of Rhumatology. Critères de diagnostic de la Polyarthrite Rhumatoïde et critères d'exclusion pour une sélection correcte des malades. *Arthritis and Rheumatology*, 31(3):315–324, 1987.

[6] B. Amor. Le concept des spondylarthropathies. *Rev. Rhum.*, 57:85–89, 1990.

[7] I. Amoura and P. Bourgeois. *Maladies systémiques - Le livre des critères.* Pfizer International Inc, 1996.

[8] J. G. Anderson. Clearing the way for physicians' use of clinical information systems. *Communications of the ACM*, 40(8):83–90, 1997.

[9] J. Arsac. *La science et le sens de la vie.* Fayard, 1993.

[10] E. J. Berk. *Java-Lex: A lexical analyzer generator for Java.* http://spectral.mscs.mu.edu/javadev/parsers/JavaLex/manual.html, Aout 1996(v 1.1).

[11] H. J. Bernelot Moens. Validation of the AI/RHEUM Knowledge Base with Data from Consecutive Rheumatological Outpatients. *Methods of Information in Medicine*, 31:175–181, 1992.

[12] H. J. Bernelot Moens and J. K. Van Der Korst. Development and validation of a computer program using Bayes's theorem to support diagnosis of rheumatic disorders. *Annals of the Rheumatic Diseases*, 51:266–271, 1992.

[13] M. Beurton-Aimar. Medical KBS design with patterns. In A Rashid, D Parsons, A Telea , editor, *Object-Oriented Technology, ECOOP'99 Workshops, Panels, and Posters*, volume 1743 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[14] M. Beurton-Aimar and B. Le Blanc. Designing Medical Knowledge-based System from Patterns. In *Proceedings of the 14th International Conference on Computers And Their Applications*, pages 218–221, 7-9, April 1999.

[15] M. Beurton-Aimar, B. Le Blanc, and J.P. Vernhes. LADRI: a decision making system in inflammatory rheumatology. In *Proceedings of the 7th International Conference on Intelligent Systems*, pages 51–56, July 1-2 1998.

[16] M. Beurton-Aimar, B. Le Blanc, and J.P. Vernhes. A decision-making system in inflammatory rheumatology using different reasoning methods. *International Journal of Computers and their Applications*, 6(2):73–77, june 1999.

[17] M. Beurton-Aimar and B. Le Blanc. Using Patterns into the Conception of a Medical KBS: LADRI, a Software for Diagnosis in Rheumatology. In *Object-Oriented Technology, ECOOP'99 Workshops, Panels, and Posters*, volume 1743 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[18] M. Beurton-Aimar, J.P. Vernhes, B. Le Blanc, J. Dehais, and R. Salamon. Creating a knowledge base for diagnosis making in rheumatology. *Revue du rhumatisme édition française*, 11:762, Novembre 1997.

[19] G. Booch. Object oriented development. *IEEE Trans. on Software Eng.*, 12(2), 1986.

[20] G. Booch. *Conception orientée objets et applications.* Addison-Wesley France, 1992.

[21] G. Booch. *Analyse et conception orientées objet.* Addison-Wesley France, 1994.

[22] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User guide.* Addison Wesley, 1997.

[23] G. Botti, M. Fieschi, D. Fieschi, and M. Joubert. Presentation of a system for computer-aided decision in diabetes therapeutics. *Pathol Biol*, 33(2):101–106, Feb 1985.

[24] R.J. Brachman. What's in a Concept : Structural Foundations for Semantics Networks . *International Journal of Man-Machine Studies*, 9:127–152, 1977.

[25] J.H. Breasted. *The Edwin Smith Surgical Papyrus.* Chicago press, 1930.

[26] B.G. Buchanan and E.H. Shortliffe. *Rule-Based Experts Systems: The Mycin Experiments of the Stanford Heuristic Programming Project* . Addison-Wesley Publishing Compagny, 1984.

[27] T. Bylander and B. Chandrasekaran. Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition. *Knowledge-based systems*, 1:65–77, 1988.

[28] B. Chandrasekaran and T. Johnson. Generic tasks and Tasks Structures: history, critique and new directions. In *Second Generation Experts Systems.* Springer Verlag, 1993.

[29] W.J. Clancey. The Epistemology of a rule based system - a framework for explanation. *Artificial Intelligence*, 20:215–251, 1983.

[30] W.J. Clancey. Heuristic Classification. *Artificial Intelligence*, 27:289–350, 1985.

[31] W.J. Clancey and E.H. Shortliffe. *Readings in Medical Artificial Intelligence.* Addison-Wesley, 1984.

[32] M.P. Cline. The pros and cons of adopting and applying Design patterns in the real world. *Communication of the ACM*, 39(10):47–49, october 1996.

[33] P. Coad. Object-Oriented Patterns. *Communication of the ACM*, 35(9):152–159, september 1992.

[34] J.O. Coplien and D.C. Schmidt. *Patterns Languages of Program Design.* Addison-Wesley, Reading, MA, 1995.

[35] J.C. Courbon, D. Dubois, B. Roy, and J.C. Pomerol. Autour de l'aide à la décision et de l'intelligence artificielle. In *1er Congrès Biennal de l'association française des sciences et technologie de l'information et des systèmes*. AFCET, Juin 1993.

[36] D. Crevier. *The Tumultuous History of the search for Artificial Intelligence* . Basic Books, a division of HarperCollins Publishers, Inc., NewYork., 1993.

[37] J. M. David. Les systèmes experts de seconde génération ou de l'importance de la modélisation dans la construction de systèmes à base de connaissances. *Techniques et sciences informatiques*, 14(4):435–471, 1995.

[38] J.M. David. Second Generation Experts Systems; a step forward in knowledge engineering. In *Second Generation Experts Systems*. Springer Verlag, 1993.

[39] FT De Dombal, DJ Leaper, JR Staniland, AP McCann, and JC Horrocks. Computer-aided diagnosis of abdominal pain. *BrMed J*, 2:9–13, 1972.

[40] J. de Kleer and B.C. Williams. Diagnosis multiple fault. *Artificial Intelligence*, 32:97–130, 1987.

[41] P. Degoulet and M. Fieschi. *Informatique Médicale*. Masson, 1998.

[42] P. Degoulet, M. Fieschi, and G. Chatellier. Decision Support Systems from the Standpoint of Knowledge Representation. *Methods of Information in Medicine*, 34(1):202–208, 1995.

[43] W.M. Detmer and E.H. Shortliffe. Using the Internet to Improve Knowledge Diffusion in Medicine. *Communications of the ACM*, 40(8):101–110, 1997.

[44] M. Dougados, S. Van Der Linden, R. Juhlin, B. Huitfeldt, B. Amor, A. Cats, G. Pasero, E. Veyse, A. Calin, B. Dijmans, I. Olivieri, and H. Zeidler. The European Spondylarthropathy Study Group Preliminary criteria for the classification of spondylarthropathy. *Arthritis Rheum.*, 34, 1991.

[45] M. Ensing, R. Paton, P. Speel, and R. Rada. An Object-Oriented Approach to knowledge representation in a biomedical domain. *Artificial Intelligence in Medicine*, 6:459–482, december 1994.

[46] J.L. Ermine. *Génie Logiciel & Génie Cognitif*. Technique et Documentation, Lavoisier, Paris, 1993.

[47] S. Falasconi, G. Lanzola, and M. Stefanelli. An Ontology-based Multi-agent Architecture for Distributed Health-care Information Systems. *Methods of Information in Medicine*, 36:20–29, 1997.

[48] S. Falasconi, G. Lanzola, and M. Stefanelli. Ontology and Terminology Servers in Agent-based Health-care Information Systems. *Methods of Information in Medicine*, 36:30–43, 1997.

[49] E.A. Feigenbaum, B.G. Buchanan, and J. Lederberg. On Generality and Problem Solving: A Case Study Using the DENDRAL Program. In *Machine Intelligence, 6*. Edimburgh University Press, Edimburgh, 1971.

[50] M. Fieschi and J. Gouvernet. Reasoning Foundations of Medical Diagnosis Revisited. In *The Promise of Medical Informatics*, Yearbook of Medical Informatics - 99, pages 78–82. IMIA, 1999.

[51] M. Fieschi and M. Joubert. Intégration de bases d'informations dans des systèmes d'information médicaux : apports de l'Intelligence Artificielle. In *Informatique et Santé*, volume 7. Springler-Verlag, Paris, 1994.

[52] G. Frege. *Conceptual Notation and Related Articles.* traduit et édité par T.W. Bynum, Oxford, Clarendon Press, 1972.

[53] R.D. Frost and M.L. Gillenson. Integrated Clinical Decision Support Using an Object-Oriented Database Management System . *Methods of Information in Medicine*, 32(2):154–160, 1993.

[54] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns Elements of Reusable Object-Oriented Software.* Addison-Wesley Publishing Compagny, 1995.

[55] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns catalogue de modèles de conception réutilisables.* Vuibert, Paris, 1999.

[56] M. Garnier, V. Delamare, and J. Delamare. *Dictionnaire des termes de médecine.* Maloine, 24e edition, 1997.

[57] D. A. Giuse and R. A. Miller. Potholes in the road to professionalism in medical informatics. *Methods of Information in Medicine*, 34:434–440, 1995.

[58] B. Goldfedder and L. Rising. A training experience with patterns. *Communication of the ACM*, 39(10):60–64, october 1996.

[59] I. Graham. *Object-Oriented methods.* Addison-Wesley, 1994.

[60] I. Graham. *Méthodes Orientées Objet.* International Thomson Publishing France, Paris, 2e edition, 1997.

[61] F. Gremy. *Informatique médicale. Introduction à la méthodolgie en médecine et en santé publique.* Médecine sciences Flammarion, Paris, 1987.

[62] B. Grenier. *Evaluation de la décision médicale.* Masson, 2ième edition, 1996.

[63] M.C. Halker and U.I. Welz. Software life cycle for knowledge based system. In *Avignon89 EC2*, 1989.

[64] J.P. Haton, N. Bouzid, F. Charpillet, M.C. Haton, B. Lâasri, H. Lâasri, P. Marquis, T. Mondot, and A. Napoli. *Le raisonnment en intelligence artificielle.* InterEditions, 1991.

[65] F. Hayes, D. Waterman, and D. Lenat. *Building Expert Systems.* Addison-Wesley, 1983.

[66] H. A. Heathfield and N. Kirkham. Object-oriented design and programming in medical decision support. *Computer Methods and Programs in Biomedicine*, 36:239–251, 1991.

[67] H. A. Heathfield and J. Wyatt. Philosophies for the Design and Development of Clinical Decision-Support System. *Methods of Information in Medicine*, 32:1–8, february 1993.

[68] H. A. Heathfield and J. Wyatt. The road to professionalism in medical informatics: A proposal for debate. *Methods of Information in Medicine*, 34:426–433, 1995.

[69] C. Hernandez, J. Sancho, M. Belmonte, C. Sierra, and F. Sanz. Validation of the Medical Expert System RENOIR. *Computers and Biomedical Research*, 27:456–471, 1994.

[70] C. Hoare. Monitors : An Operating System Structuring Concept. *Communications of the ACM*, 17(10):549–557, 1974.

[71] R. Hodgson. Finding,building and reusing objects. In *Object Oriented Design, Unicom seminars*. Uxbridge, 1990.

[72] D. Hofstadter. *Gödel Escher Bach : les brins d'une guirlande éternelle*. InterEditions, 1985.

[73] S. Hudson. *LALR Parser Generator for Java*. http://www.cs.princeton.edu/ appel/modern/java/CUP/, July 1999 (v0.10j).

[74] G.G. Hunder, B.A. Bloch, and B.A. Michel. The American College of Rheumatology 1990 criteria classification of giant cell arteritis. *Arthritis and Rheumatology*, 33:1122–1128, 1990.

[75] D.H.H. Ingalls. The Smalltalk-76 programming system: Design and implementation. In *5th Annual ACM Symposium on Principles of Programming Languages*, Tucson, Arizona, 1978.

[76] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. *Object-Oriented Software Engineering, second edition*. Addison-Wesley, ACM Press, 1993.

[77] M. Jubola, Y. Autamo, E. Kentala, and I. Pyykkö. An essay on power of expert systems versus human expertise. *Medical Informatic*, 20(2):133–138, april-june 1995.

[78] Ch.E. Kahn, L.M. Roberts, K.A. Shapper, and P. Haddway. Construction of a bayesian network for mammographic diagnosis of breast cancer. *Computers in Biolology and Medicine*, 27(1):19–29, 1997.

[79] M. F. Kahn, A. P. Peltier, O. Meyer, and J. C. Piette. *Les Maladies Systémiques*. Médecine-Sciences. Flammarion, 3ème edition, 1991.

[80] D. Kayser. *La représentation des connaissances*. Hermès, Paris, 1997.

[81] D.G. Kilman and D. W. Forslund. Virtual Patient Record. *Communications of the ACM*, 40(8):111–117, 1997.

[82] G.J. Kupermann, R.M. Gardner, and T.A. Pryor. *HELP : a Dynamic Hospital Information System*. Springer-Verlag, New York, 1991.

[83] M. Lai. *UML La notation unifiée de modélisation objet*. InterEditions, 1997.

[84] P. Le Beux, A. Burgun, and M. Cléret. De la méconnaissance à l'expertise. In *Informatique et Santé*, volume 7. Springler-Verlag, Paris, 1994.

[85] B. Le Blanc and G. Palmer. AntibES : an expert system for antibiotic prescription based on fuzzy logic . *13th Int. conf. on Expert Systems, Avignon*, pages 123–132, 25-28 Mai 1993.

[86] D. Lea. Patterns-Discussion FAQ. *http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html*, december 1997.

[87] R. S. Ledley and L. B. Lusted. Reasoning Foundations of Medical Diagnosis. *Science*, 130:9–21, 1959.

[88] E. Lepage, M. Fieschi, R. Traineau, J. Gouvernet, and C. Chastang. Système d'aide à la décision fondé sur un modèle de réseau bayesien : application à la surveillance transfusionnelle. In *Informatique et Santé*, volume 5, pages 76–97. Springler-Verlag, 1992.

[89] J. Levine, T. Mason, and D. Brown. *lex & yacc*. O'Reilly, 2nd edition, 1992.

[90] P. Levine and J.Ch. Pomerol. *Systèmes interactifs d'aide à la décision et systèmes experts*. Hermès, 1989.

[91] M. Linster and M.A. Musen. Use of KADS to create a conceptual model of the ONCOCIN task. *Knowledge Acquisition*, 4:55–87, 1992.

[92] R. C. Martin, D. Riehle, and F. Buschmann. *Pattern Languages of Program Design 3*. Software Patterns Series. Addison-Wesley, 1997.

[93] A.T. Masi, G.P. Rodnan, and T.A. Medsger. Preliminary criteria for the classification of systemic sclerosis (scleroderma). *Arthritis and Rheumatology*, 23:581, 1980.

[94] T. Menzies. Object-Oriented Patterns: Lessons from Expert Systems. *Software-practice and experience*, 1:1–22, 1997.

[95] R.A. Miller, H.E. Pople, and J.D. Myers. INTERNIST-1, An Experimental Computer-Based Diagnostic Consultant for General Internal Medicine. *New England Journal of Medicine*, 307:468–476, 1982.

[96] G. Molino, F. Molino, D. Furia, F.Bar, S. Battista, and N. Cappello. Computer-Aided Diagnosis in Jaundice : Comparison of Knowledge ans Probabilistic Approaches. *Methods of Information in Medicine*, 35:41–51, 1996.

[97] M. A. Musen and E. A. Barnett. The road to professionalism in medical informatics: the role of training. *Methods of Information in Medicine*, 35:148–151, 1996.

[98] M. A. Musen and A. Th. Schreiber. Architectures for intelligent systems based on reusable components. *Artificial Intelligence in Medecine*, 7:189–199, 1995.

[99] M.A. Musen. Dimensions of knowledge sharing and reuse. *computers and biomedical research*, 25:435–467, 1992.

[100] G. Myers. *Composite/Structured Design*. Van Nostrand Reinhold, 1978.

[101] A. Newell. The Knowledge Level. *Artificial Intelligence*, 18(1):87–127, 1982.

[102] A. Newell and H.A. Simon. *Human Problem Solving*. Englewoods Cliffs, Prentice Hall, 1972.

[103] The American College of Rheumatology. The 1982 revised criteria for the classification of systemic lupus erythematosus . *Arthritis Rheum*, 25:1271–1277, 1982.

[104] W.F. Opdyke and R. Johnson. Refactoring: An aid in designing application framworks and evolving object-oriented systems . In *Proceedings of the 21st Annual Computer Science Conference(ACM CSC'93)*, pages 66–73, Marist College, Poughkeepsie, NY, september 1990.

[105] J. Parsons and Y. Wand. Choosing Classes In Conceptual Modeling. *Communications of the ACM*, 40(6):63–69, 1997.

[106] J. Pitrat. *Métaconnaissance, futur de l'Intelligence Artificielle*. Hermès, 1990.

[107] H.E. Pople. *Artificial Intelligence in Medicine*, chapter 5. Heuristics Methods for Imposing Structure on Ill-Structured Problems : The Structuring of Medical Diagnosis. AAAS Selected Symposium, 1982.

[108] H.U. Prokosch, F. Amiri, D. Krause, G. Neek, and J. Dudeck. A SEmantic Network Model for the Medical Record of a Rheumatology Clinic. In R.A. Greenes, H.E. Peterson, and D.J. Protti, editors, *Proceedings of the 8th World Congress on Medical Informatics*, pages 240–244, 1995.

[109] M.R. Quillian. Semantic Memory. In *Semantic Information Processing*. M.I.T. Press, 1968.

[110] W. Raghupathi. Health Care Information Systems. *Communications of the ACM*, 40(8):81–82, August 1997.

[111] J.A. Reggia, D.S. Nau, and P.Y Wang. Diagnostic expert systems based on a set covering model. *Int. J. Man Mach. Stud.*, 18:437–460, 1983.

[112] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[113] J.L. Renaud-Salis, F. Bonichon, M. Durand, C. Lagarde, P. Mendiboure, and J.P. Serre. SENEX : un système d'aide à la décision et à la conduite des essais thérapeutiques dans les cancers du sein, intégrant systèmes experts et système de gestion de base de données . In *Systèmes Experts en Médecine*, pages 107–118. EC2, 1988.

[114] J. Rumbaugh, M. Blaha, and W. Premerlani. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[115] B. Russel and A.N. Whitehead. *Principia Mathematica*. Cambridge University Press, 8th edition, 1962.

[116] R. Sacile, C. Ruggiero, and R. Dieng. Using CommonKads to create a conceptuel model of a guideline system for breast cancer prognosis. *Medical Informatics*, pages 45–59, 1996.

[117] R. Salamon. *Statistique médicale*. Masson, 1987.

[118] R. Salamon, V. Leroy, S. Maurice-Tison, and B. Le Blanc. Health Informatics : Handle with Caution. *Methods of Information in Medicine*, 36:79–81, 1997.

[119] A. Th. Schreiber, P. Terpstra, P. Magni, and M. Van Velzen. Analysing and implementing VT using COMMON-KADS. In *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge Based Systems Workshop*, 1994.

[120] G. Schreiber, B. Wielinga, H. Akkermans, W. Van de Velde, and A. Anjewierden. CML: the CommonKADS Conceptual Modelling Language. In *Proceedings of the 8th European Knowledge Acquisition Workshop, EKAW'94*, A future for knowledge acquisition, pages 1–25. LNA867 (Heidelberg: Springer-Verlag), 1994.

[121] G. Schreiber, B. Wielinga, and J. Breuker, editors. *KADS: A Principled Approach to Knowledge-based System Development*. Academic Press, London, 1993.

[122] K. Shankar. *Data Design: Types, Structures and Abstractions*. C. Vick and C. Ramamoorthy New York : Van Nostrand Reinhold, 1984.

[123] E. Shortliffe, A. Scott, M. Bischoff, C. Van Melleand, and W. Jacobs. ONCOCIN: An expert system for oncology protocol management. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 876–881, Vancouver, B.C.,Canada, 1981.

[124] E.H. Shortliffe. *MYCIN : Computer Based Medical Consultation*. Elsevier, 1976.

[125] E.H. Shortliffe. The Adolescence of AI in Medicine: Will the field come of age in the '90s? *Artificial Intelligence in Medicine*, 5:93–106, 1993.

[126] J.F. Sowa. *Conceptual Structures. Information Processing in Mind and Machine.* Addison Wesley, 1984.

[127] L. Steel. Second Generation Expert Systems: Future Generation. *Computer Systems*, 1(4), 1985.

[128] P. Szolovits and S.G. Pauker. Categorical and probabilistic reasoning in medecine revisited. *Artificial Intelligence*, 59:167–180, 1993.

[129] K. Tanimoto, K. Nakano, S. Kano, S. Mori, H. Ueki, H. Nisihitani, T. Sato, T. Kiuchi, and Y. Ohashi. Classification criteria for polymiositis and dermatomyositis. *Journal of Rheumatology*, 22:668–674, 1995.

[130] A. Thayse, A. Bruffaerts, P. Dupont, E. Henin, Y. Kamp, J.P. Müller, J.L. Binot, and D. Snyers. *Approche logique de l'intelligence artificielle. 1. De la logique classique à la programmation logique.* Dunod Informatique, 1990.

[131] A. Thayse, P. Delsarte, J. Hagelstein, G. Louis, A. Rifaut, M. Vauclair, E. Dubois, A. Van Lamswweerde, and F. Van Der Linden. *Approche logique de l'intelligence artificielle. 4. De l'apprentissage artificiel aux frontières de l'IA.* Dunod Informatique, 1991.

[132] S.W. Tu, H. Eriksson, J.H. Gennari, Y. Shahar, and M.A. Musen. Ontology-based configuration of problem-solving methods and generation of knowledge-acquisition tools: application of PROTEGE-II to protocol-based decision support. *Artificial Intelligence in Medicine*, 7:257–289, 1995.

[133] Unified Modeling Language, version 1.1 . Rational Software Corporation, 1997.

[134] J.H. Van Bemmel, M.A. Musen, and R.A. Miller. *Handbook of Medical Informatics*, chapter Methods for Decision Support, pages 233–260. Springer, 1997.

[135] J.H. Van Bemmel, M.A. Musen, and R.A. Miller. *Handbook of Medical Informatics*, chapter Methodology for Information Processing, pages 438–239. Springer, 1997.

[136] W. Van Melle. A domain-independant production-rule system for consultation programs. In *Proceedings of Sixth International Joint Conference on Artificial Intelligence*, pages 923–925, Standford, CA, 1979. Standford University, Departement of Computer Science.

[137] J.P. Vernhes, M. Beurton-Aimar, B. Le Blanc, J. Dehais, and R. Salamon. Software for diagnosis making in inflammatory rheumatology. *Revue du rhumatisme édition française*, 11:763, novembre 1997.

[138] J. M. Vlissides, J. O. Coplien, and N. L. Kerth. *Pattern Languages of Program Design 2*. Software Patterns Serie. Addison-Wesley, 1996.

[139] C. Vogel. *Génie Cognitif.* Masson, 1988.

[140] C. Vogel. Comment qualifier un système à base de connaissances ? *Génie Logiciel & Systèmes Experts*, 16:4–9, 1989.

[141] S.M. Weiss and C.A. Kulikowski. EXPERT: a system for developing consultation models. In *Proceedings of Sixth International Joint Conference on Artificial Intelligence*, pages 942–947, Standford, CA, 1979. Standford University, Departement of Computer Science.

[142] M. Yamaguchi, A. Ohta, and T. Tsunematsu. Preliminary Criteria for classification of Adult Still's Disease. *Journal of Rheumatology*, 19(3):424–430, 1992.

[143] W. Zimmer. Experiences using Design Patterns to Reorganize an Object-Oriented Application. *Architectures and Processes for Systematic Software Construction, URL http://www.fzi.de/prost/publications/*, 1, 1995.

Table des figures