Introducing Patterns in Design of Computer-Aided Diagnostic System: a Rheumatologist Application

Marie Beurton-Aimar[1,2], J.Philippe Vernhes[3]
[1] Laboratoire de physiologie mitochondriale,
Universit Victor Segalen - Bordeaux2,
146, rue Lo Saignat, 33076 Bordeaux cedex.
[2] LAboratoire Bordelais de Recherche en Informatique,
Universit Bordeaux1.
351, cours de la Libration, 33405 Talence.
[3] Service de Rhumatologie,
Centre Hospitalier R. Boulin, 33500 Libourne.

Abstract

Architecture of software is a major point of interest today and the use of structuring methods to develop systems become widespread. For an inflammatory rheumatology application, we have worked on the design of a computer-aided diagnostic system including several methods of reasoning. To define such system an Object-Oriented analysis was managed, but the problem to identify the "good" objects is not always solved by this analysis. Some tools, called patterns, can help developers to solve this problem. Patterns give an assistance to refine existing objects and to discover new ones. Catalogs have been published as "Design Patterns" by E. Gamma. We have built a software called LADRI (Logiciel d'Aide au Diagnostic en Rhumatologie Inflammatoire) which integrates 4 of these patterns. The patterns COMPOSITE and STATE are used for knowledge representation; STRATEGY models the diagnostic algorithms and ITERATOR performs generic task encountered in diagnosis making system. Finally, we have designed an Object architecture of medical diagnosis and used it to implement two complementary methods of diagnostic reasoning. LADRI was used about 78 patient cases and obtained more than 92% of correct diagnoses.

Keywords : Design patterns, Object-Oriented development, medical diagnostic support system, rheumatologist application.

1 Introduction

Our purpose is to present some problems linked to the design of medical diagnostic systems. The architecture of such systems takes care two main components: the knowledge base which contains the definitions of all pathologies, and the modules of treatment corresponding to the diagnostic methods used by clinicians.

Nowadays, computer-aided diagnostic system cannot be a *black box* which contains a monolithic process. Systems must be able to store information, to search for a specific disease, to consult validated clinical tables, and to compare the results for several diseases. This suggests the production of strongly inter-connected process modules, sharing a common base of information. Sharing information is made possible through the design of an abstract model of the knowledge base. Then, processes can be specified regardless of the data implementation.

Since the first expert systems, some experimental models had been made in clinical medicine; rheumatologist application systems include Renoir [1], Cadiag2/Rheuma [2] and AI/RHEUM [3]. From these experiences, many authors argued for structured models, see Heathfield [4] or Adlassnig [5] for a more detailed discussion. Object-Oriented methods like OMT[6], HOOD[7] or UML[8] (see [9] for a comparison of the most known methods), supply a framework of such realization [10]. They are powerful tools to translate the concepts of the domain into modules directly implemented with programming language. But they rarely give a response to the essential question: "how to find the good objects?". There are many ways to structure the knowledge into the database, and designers can only rely on their own experience to make the more appropriate choices between the various possibilities. Moreover, significant elements remain hidden during the stage of first analysis. Conception of software requires the definition of objects which do not exist as full entities in a human cognitive system. This includes topics such as algorithm models linked to data structures, or description of generic behaviours. Patterns proceed from these two expectations, they give a new approach of the modeled domain and provide us with some specific solutions to design problems.

Patterns have been introduced by Christopher Alexander, a building architect [11]. Alexander's use of the word "pattern" takes on more meaning than the usual dictionary definition. Alexander's patterns are both a description of a recurring pattern of architectural elements and a rule for how and when to create that pattern [12].

A next stage is accomplished with the Gang of Four's book "Design Patterns" [13]. These authors belonging to the Object community replace patterns into a software context.

> *Design patterns describe software abstractions used by advanced designers and programmers in their software. As abstractions, patterns commonly cut across other common software abstractions like procedures and objects, or combine more common abstractions in powerful ways* [14].

This paper presents the main parts of our Object-Oriented model and the integration of 4 patterns in this model. Then, we present the resulting software called LADRI (Logiciel d'Aide au Diagnostic en Rhumatologie Inflammatoire) and the two reasoning methods included. In conclusion, we discuss the introduction of patterns in the design of a clinical computer-aided diagnostic system.

2 Integration of patterns in the model

One of the aims of this project was to design an architecture with only one knowledge base for several reasoning systems [15]. Two main reasons led to this choice: the consistency of the disease definitions and the maintenance of the application. This problem can be solved by the structuring mode of the knowledge and by the design of an abstract model for the system.

The first basements of our model is the design by L. Steel [16] concerning the "*second generation*" of expert systems. This model is composed of two essential modules. The ***Knowledge Level*** gathers information about the patient and the diseases. The ***Computed Level*** combines the sets of reasoning procedures already defined. The encapsulation of the objects belonging to each module makes it possible to modify diseases (or to add new diseases) without disturbing the operations in the ***Computed Level***. Conversely, the addition of new reasoning procedures does not require major modifications in the knowledge base.

At this step of analysis, the following entities are retained: *Description*, *Signs*, *Value* and *Reasoning methods*. According to the Object-Oriented methods, all of these entities are defined as objects in the model [17]. A *Description* is an agregation of *Signs* like *articular pain*, *articular destruction* etc. The *PatientDescription* and the *DiseaseDescription* inherit from this generic *Description*. This organization ensures the consistency between the patient's and the disease's descriptions and simplifies their comparison. All signs belonging to a disease definition are weighted according to their significance in each diagnosis. This information is modeled by the object *Value*. Each sign used at least in one disease description is stored in *Dictionary*. The patient's description is made from this same vocabulary. A particular object *UserInterface* is defined to store current instances of the disease database, the patient database, the dictionary and the diagnostic engine. Two methods of reasoning are defined: scoring system and criteria list system, they inherit from *ReasoningMethod*. According to graphic conventions of UML [8], figure 1 shows the main objects corresponding to these entities and the links between them.

Figure 1 here

Then, this classical organization was improved by addition of new generic objects issued from "Design Patterns" [13]. The patterns COMPOSITE and STATE are used for the management of signs, they belong to the ***Knowledge Level***. The patterns ITERATOR and STRATEGY describe generic classes of treatments, they belong to the ***Computed Level***.

2.1 Signs and syndromes: COMPOSITE

This pattern allows to consider an aggregation of objects like a primitive object. The COMPOSITE object and the primitive objects are placed at the same level in a recursive structure. These objects inherit from an abstract class (*Component*) which is the interface with the other parts of the system. Thus, there is an unique access to these objects whatever their internal structure.

In clinical medicine, the problem is to model in a same way pathologies which have heterogeneous signs. Definitions of diseases are characterized by simple signs like *articular pain* or by complex syndrome like *inflammation*. This *inflammation* is itself characterized by a *pain* and a *swelling* which are two primitive signs. As there is an unique interface for these objects, the search procedure of the different signs of a disease is easier to implement. In the same way, when a new type of symptom is added, the updates only concern the objects belonging to the composition. Figure 2 presents the graph of inheritance for the signs: the object syndrome is a COMPOSITE, it is an aggregation of signs (graphic conventions are those of [13]).

Figure 2 here

Component implements treatments linked to the communication with clients and behaviours common to all elements in the composition. In our application, we want to compare the patients' signs to those of each pathology definition. Whether the sign is primitive or complex, client only wants to know the result of this comparison. Complexity of the structure of signs is managed inside the COMPOSITE.

Note that if this pattern corresponds well to the needs for handling signs, it may be unwize to introduce it in all the hierarchical structures. Thus, the knowledge base contains a list of pathologies which is not a COMPOSITE because only one object Disease exists and we do not need to introduce an interface between this object and its clients. Creation of a composition of diseases is not relevant for our application.

2.2 Presence or absence of sign : STATE

The pattern STATE concerns the design of the object *Description*. In disease descriptions, the clinician specifies whether he is looking for the presence of a sign or in the contrary he is checking the absence of this sign. In most systems, a weight is affected to each sign belonging to the disease definition. The presence or absence is then coded trough this weight, for example by negative value for absence. But a difference exists between these two statements: *"the articular destruction is rarely found in Scleroderma"* (i.e. presence of destruction is not significant) and *"we do not have to find any destruction in a Lupus"* (i.e. absence of destruction is in favor of Lupus). These definitions are difficult to model if "presence" or "absence" of signs are coded in the same weight. By using the pattern STATE, the different possible states of an object are represent explicitly. It is an alternative to the implementation of this state by a value.

There is another argument against the using of value for state representation. As we wish to implement several diagnosis methods, we manage a particular type of value for each method: numeric scale for scoring system and symbolic values for criteria list. If the definition of states is made outside the valuation system, this information is persistent when the type of valuation changes. Thus the implementation of new computing procedures is simplified.

At last, in our model there is a generic object *Description* for disease and patient. So the inheritance mechanism allows to use this concept of state also in the patient description. A sign in a patient description have no value (because value measures the weight of signs in the disease diagnosis). The explicit representation of the state makes it possible to specify if a sign have been searched and found (present) or not found (absent) or not researched (no mentioned) in the patient description.

During the validation stage of the model, some new states could appear. In our application, a state *information* has been detected to characterize, the *past medical history* of the patient (such signs have been present but are not interpretated as present for the current diagnosis). The definition of the pattern facilitates the introduction of a new state; in this case, the pattern performed its task of *"frame"* for the definition of *information*.

Figure 3 here

Figure 3 shows a *Description* which is an aggregation of 3 possible *States* (Present, Absent or Information). For each state, there are the list of signs in this specific state and functions to add signs, to view signs etc. In this design, the signs never know their own state. This is the state which knowns which signs belong to it. This interpretation of the pattern STATE is unusual. Usually, the state is an attribute of the object. We choose this organization because it facilitates the writing of procedures such as: "to search all present signs common to the patient and disease description", "to exclude pathologies for which the patient have no present sign". Moreover, the set of "present" signs concept, is a good reflection of the classification made by a practitioner during the examination of a patient.

2.3 Generic treatment: STRATEGY

The role of the pattern STRATEGY is to define an abstract class of algorithm families. We use it to design a generic class of reasoning method. Such abstraction is classical in software engineering, but in most Expert Systems, software is organised around one unique method of reasoning. The reasoning is the central element of the system and the organization is strongly dependent of the chosen reasoning. If it changes, the system has to be rebuilt. The modularity principles are often used in engineering software but is still rarely observed in software for Articifial Intelligence. Modularity and encapsulation of treatments are necessary due to the sharing of the knowledge base by several methods of reasoning.

Our two main families of treatments, scoring system and criteria list, inherit from the STRATEGY pattern. Each family models a different reasoning principle and implements its own behaviour: the searching weights for all observed signs for the first one, the validation of a restrictive list of signs for the second one. Owing to the inheritance mechanism, the subclasses have the same interface, so switching the methods is user friendly. In addition, we want to be able to define some procedures shared by all methods: to search only present signs, to consider as missing all signs which are not specified during the examination of the patient etc. These procedures are defined into the superclass STRATEGY. With the STRATEGY pattern, a generic model for decision functions is available; by the way, it makes it easier the addition of new diagnostic procedures. This kind of design allows to reduce the complexity of a global system gathering several reasoning methods.

Figure 4 here

2.4 Inference task: ITERATOR

The pattern ITERATOR concerns a generic task performed by all reasoning methods. The diagnostic task is mainly an enumeration of all patient signs observed, to compare them with signs belonging pathology descriptions. This task is involved whatever the structure of these definitions or the applied operation in the reasoning module. If the access method to the elements is coded inside the computing algorithms, it must be re-written for each algorithm. A first improvement is to implement a module to manage lists, arrays, trees... and to call corresponding functions. An additional level of abstraction is obtained with the introduction of an object ITERATOR. It gives up a generic access to the element independently from the data structures. Thus, the implementation of a new data structure does not imply modifications in diagnostic computing procedures, and conversely a modification or an addition of procedures does not change the access to the definitions. This pattern is a good example of an object which models properly one system task, but it should not have been implemented with this level of abstraction at once.

Figure 5 here

ITERATOR allows communication between data modules and treatment modules. An absolute separation between these two modules, like in the model of "2^{nd} generation of expert systems", is sometimes considered as artificial. Chandrasekaran [18] promotes a strong connection between the choice of the data structures and the treatments to infer. He assumes that if a problem matches the function of a generic task, then a generic task provides a knowledge representation and an inference strategy that can be used to solve the problem. This point of view can be drawn towards the approach of the patterns. But patterns have a higher level of abstraction, for example the *Iterator* proposes the definition of a task without taking into account the specific data structure. In this way the use of patterns allows to preserve the modularity of the system.

Figure 6 here

Using these four patterns we have described elements which are always present in diagnosis system but hidden into the objects identified by the Object analysis. This analysis gives a view linked to the semantic of the data. Using the patterns an organization was designed more directed to the tasks to accomplish and the optimal structures to make them. Patterns do not summarize "what is" a diagnostic system but give a skeleton of such system which can be overlapped with the Object model. Figure 6 shows this overlapping (patterns are shown in grey boxes).

3 A computer-aided diagnostic system in rheumatology: LADRI

From the above model including patterns as objects, a computer-aided diagnostic system in rheumatology was implemented. Two patterns were integrated at strategic position to encapsulate the internal representation of signs (COMPOSITE) or to enable the polymorphism of treatments (STRATEGY). The other ones improved the implementation of the software separating the interpretation of signs from the type of value (STATE) and extracting generic task from treatment algorithms (ITERATOR).

Figure 7 here

The introduction of patterns has induced the creation of some new objects: 4 object classes corresponding to them, 3 subclasses for the states, and the different structures working with (ITERATOR. More than 20 objects compose the core of LADRI, they are shown in figure 7 (GUI modules are not considered here). The software is organised from the object *ExpertSystem* which is the interface between the system and the user. *ExpertSystem* instanciates a disease database, a patient database and a supervisor. Pathologies and patients are described with 3 sets of signs gathered by states. Today, LADRI contains the definition of 12 pathologies obtained from more than one hundred signs. Access to each *Sign* belonging to *Description* is ran by an object *Iterator*. A new object called *Supervisor* is defined to drive the collaboration of the two methods of reasoning. It stores the results produced by the two treatments and a set of procedures which describes how the collaboration works: which procedure is the first one? when is called the next one? *Supervisor* instanciates an object *Diagnosis* for each chosen *Strategy* and defines generic operations of diagnostic computing. These operations are summarized by: to iterate for each definition the comparison, to take into account the result of this comparison, and if it is positive, to apply the operation belonging to the treatment module. The specific treatments of the two methods of diagnosis are implemented in the objects *ScoringSystem* and *CriteriaList*.

3.1 Implementation of reasoning methods

The first method implemented corresponds to a differential diagnosis; it is performed by calculating a score. When the expert in rheumatology defined a disease (afterwards stored in the disease database), he gave a value for all signs belonging to this definition. Each value represents the weigth of the signs in the disease diagnosis. Figure 8 shows for the 12 pathologies the number of signs retained, state by state and in total. The maximum score for a disease corresponds to the maximum value reachable by a patient in the diagnosis of this disease. Even if the number of signs can double for some pathologies (values from 13 to 29), the maximum score variation is less (values from 85 to 148). Moreover, a pathology may have a more large number of signs than another one but with a less maximum score (e.g. Horton disease: 17 signs for 114 points, Still disease: 19 signs for 97 points).

Figure 8 here

With the scoring system, when the patient case is compared to disease definitions, the following rule is apply: $LET\ W_{ij}$ the weight of sign S_i in diagnosis of disease D_j, LET an observed patient O and the following function $O(S_i)$: IF the sign S_i belongs to the set of patient signs $THEN \quad O(S_i) = 1 \quad ELSE \quad O(S_i) = 0$.
Total number T_j for the patient O in diagnosis of disease D_j is obtained by: $T_j = \dfrac{\sum_i O(S_i).W_{ij}}{\sum_i W_{ij}}$

This standardized calculation makes it possible to treat equally pathologies defined with a large number of signs and those defined only with a few, these differences are really observed in rheumatology (see figure 8). For each pathology, a specific total can be computed for each state. During the analysis of a patient case, the scoring module computes the sum using weights of *present, absent* and *information* signs: PW_{ij}, AW_{ij}, IW_{ij}. Then, it is possible to analyze which set of signs (*present, absent* or *information*) leads to the following sum:
$T_j = \dfrac{\sum_i O(S_i).PW_{ij} + \sum_i O(S_i).AW_{ij} + \sum_i O(S_i).IW_{ij}}{\sum_i W_{ij}}$
For a disease D_j, if the patient has no present sign, then $\sum_i O(S_i).PW_{ij} = 0$ and the disease D_j is eliminated from the list of possible pathologies. The result is the list of possible diagnoses ordered by the total T_j.

The scoring system is mainly an expert's report, but other information about the definitions of the pathologies is available. Standard clinical tables for some pathologies are published by the international rheumatologist associations; e.g. Kahn's criteria for *Still Disease* [19] or the ESSG criteria [20] for the *Spondylarthropathies*. These tables contain the signs most often observed in the study of these pathologies during epidemiological investigations. The second reasoning method uses these lists of signs and compares them to the patient's profile. It is the criteria list system.

The values of signs used in the criteria list method are *major, minor* and *exclusion*. The operation is a rule about the number of signs needed to validate the diagnosis. This rule is established according to the values of signs, e.g. to confirm *Still Disease* using Kahn's criteria, 5 signs are required with at least 2 major signs and no exclusion sign. If a patient meets only some of the criteria list, not even a partial diagnosis can be made. Of course, the criteria list method eliminates some pathologies for which the patient exhibits the exclusion criteria.

3.2 Collaboration of reasoning methods and final diagnosis

Out of respect for modularity and encapsulation principles, each treatment does not know anything about the other ones. The mechanism of inheritance and the definition of a generic class of treatment *Strategy* keep hidden the real coding of each procedure of reasoning. But it needs to define an object to deal with each of them. It is the role of the object *Supervisor*. It drives all stages of the diagnostic session. Afterwards practitioner have selected from dictionaries the signs best corresponding to the observed case and built the description of the patient, the *Supervisor* activates the scoring system. Then, it stores the results and launches the criteria list treatment. At least, it applies a merging algorithm on the different results.

Figure 9 shows the results which were obtained with LADRI for 78 patients. For each pathology, the third column shows the number of cases for which this pathology was classified at the head of the list by the scoring system. The fourth column shows the number of cases presenting a complete clinical table. Two main problems with the criteria list system may be seen: a common list for the three *Spondylarthropathies* (*Ankylosing Spondylitis, Psoriatic Arthritis* and *Reactive Arthritis*); and the missing list for *Rhyzomelic PseudoPolyarthritis*, *Chondrocalcinosis*, and *Gout*.

The scoring system classified the right disease at the head of the list for 70 cases and the criteria system validated 44 cases. After running the two procedures, the following algorithm is applied. The diseases which present exclusion signs are eliminated from the ordered list given by the scoring system. Then, if there is no validated pathology, the final diagnosis is the head of the list. If there are one (or several) disease which are validated, we must aggregate the different results. We take into account two cases. The first one is when a pathology D is validated by a criteria list and is at the head of the ordered list (produced by the scoring method); in this case D

is definitely diagnosed. The second one is when D is validated and is not at the head of the list. In this case, we examine all pathologies which have a best rank. If those do not have a criteria list the disease D is validated and these pathologies are added as possible diagnoses. If they have a criteria list (and it is not validated of course), it is a misdiagnosis. The column 5 shows the final diagnosis given by this procedure. In total, 72 cases (more than 92%) are definitively diagnosed by the system.

Figure 9 here

4 Discussion

During the analysis and conception stages, Object methods describe principles of Object design and diagrams for system organization. But these methods do not reply to the essential question which have been asked by Booch [10]: how to discover the good objects or the good classes? Rumbaugh [21] remarks that software designers have had a great handicap in learning to design: "There are no *program museums* where you can examine great programs to emulate". Patterns catalogs, as [13], are a kind of those museums where the *works of masters* are shown. Rather than formulating a global theory about "how to do", patterns suggest some examples, as achieved as possible, and propose them "ready to use".

In our application, we have identified 4 patterns present at different level in the Object model. COMPOSITE encapsulates the signs encoding and keeps inside the complexity of their structure. STATE gives a new approach to specify signs in descriptions. It carries out design which allows to access to the information about the state of the signs without taking into account the type of value used by the treatment module. STRATEGY and ITERATOR improve modularity of treatments. It is important to observe that patterns are applied in the design phase and are not overly concerned with specific implementation issues [22].

Patterns make up the Object-Oriented analysis and give a different point of view of the system, more efficiency and more oriented towards the architecture of the software. With patterns, the Object model was improved with the addition of abstractions relevant to the problem: to build a flexible and evolutive diagnosis support system. In a response to Heathfield about the professionalism in medical informatics [23], Giuse have remarked the lack of handbooks of precise, predictive rules for software engineering, and well documented software modules [24]. The approach of pattern designers is directly linked to this concern. They try to give a common vocabulary for some specific modules, well identified and corresponding to precise tasks. In current practice, the use of these tools is only starting because this supposes a sufficient background in software engineering and in Object design. Even if Doug Lea explains that when patterns are integrated, things are so evident that *even your grandmother could think of it* [25], the purpose is not to claim that an engineer inexperienced with Objet methods, could perform the design of a system only from the reading of existing patterns. Pattern catalogs are a source of mood and an opportunity to share the experience of the "experts" in Object design. Patterns do not replace the Object methods, they complemente them.

Some criticism about patterns and Object methods evoke an over-cost of development due to an excessive modelling and a trend to get more and more the number of modules to implement. In fact, we had created additional level of abstraction for STATE, ITERATOR and STRATEGY. But the quantity of the effective coding for these classes is low and this added cost is make up for the realized benefits on maintenance.

In a system like LADRI, one of the major issues is to keep the model and the application up to date by additions or modifications of the data structures and algorithms. This experience joins the preoccupation of some actors of the domain [26]. Heathfield [23] has noted that the implementations of systems are often lacking a coherent design and development strategy. They failed to facilitate flexibility and portability to new sites. In the same way, Musen [27] claims that basic principles of software engineering are not yet applied universally in the development of clinical applications; contemporary work on participatory design of computer systems are not yet widely recognized in the medical informatics community. In this aim, patterns had been an essential contribution to the quality of this kind of softwares and to their evolution.

Acknowledgements

The authors warmly thank the members of the rheumatology ward of Professor J. Dehais in Bordeaux university hospital for their assistance in data collection.

References

[1] 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.

[2] K. Adlassnig, H. Leitich, and G. Kolarz. On the applicability of the diagnosis criteria for the rheumatoid arthritis in an expert system. *Expert Systems Applications*, 6:441–448, 1993.

[3] J.F. Porter, L.C. Kingsland, D. Lindberg, I. Shah, J. Benge, and S. Hazelwood. The AI/RHEUM knowlegde-based computer consultant system in rheumatology. *Arthritis and Rheumatism*, pages 219–226, 1988.

[4] 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.

[5] K. Adlassnig, H. Leitich, and G. Kolarz. Systmes experts en rhumatologie. *Rhumatologie en Europe*, 25(3):104–108, 1996.

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

[7] Hood Technical Group. *HOOD reference manual 3.1*, masson edition, 1993.

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

[9] I. Graham. *Object-Oriented methods, second edition*. Addison-Wesley, 1994.

[10] G. Booch. *Object oriented analysis and design with applications*. Benjamin/Cummings, Redwood city, CA, second edition, 1994.

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

[12] K. Beck and R. Johnson. Patterns generate architectures. In *ECOOP'94*, 1994.

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

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

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

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

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

[18] 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.

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

[20] 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.

[21] J. Rumbaugh. What is a method? Technical report, rational software corporation, 1995.

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

[23] 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.

[24] 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.

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

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

[27] 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.

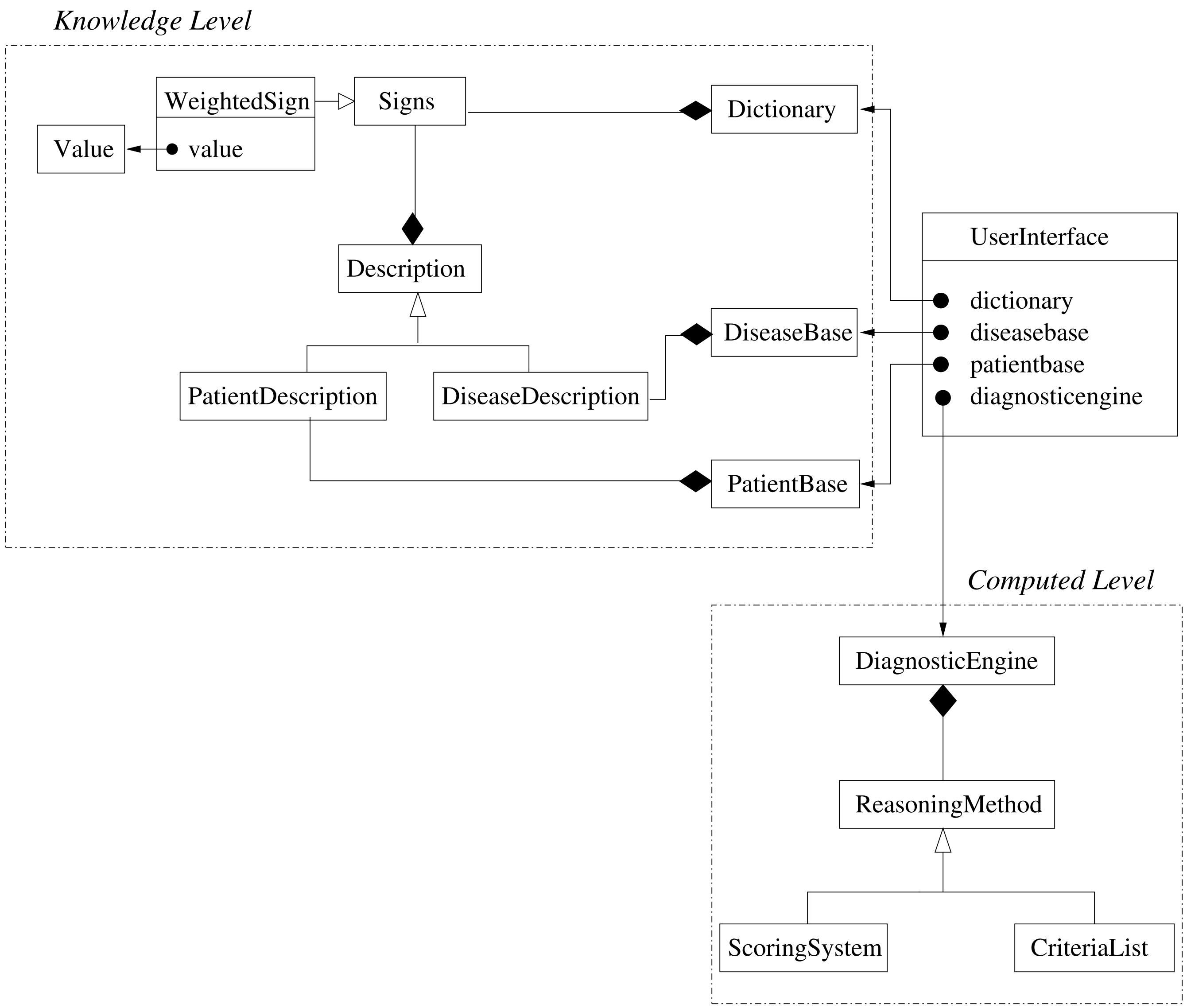

Figure 1: Global architecture of the system

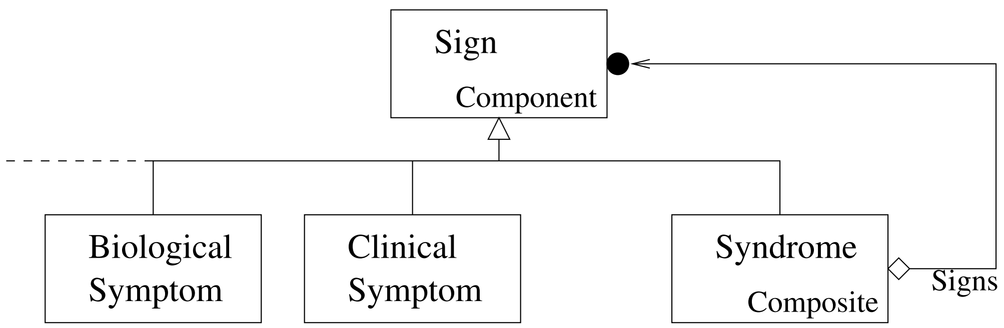

Figure 2: Inheritance graph of signs

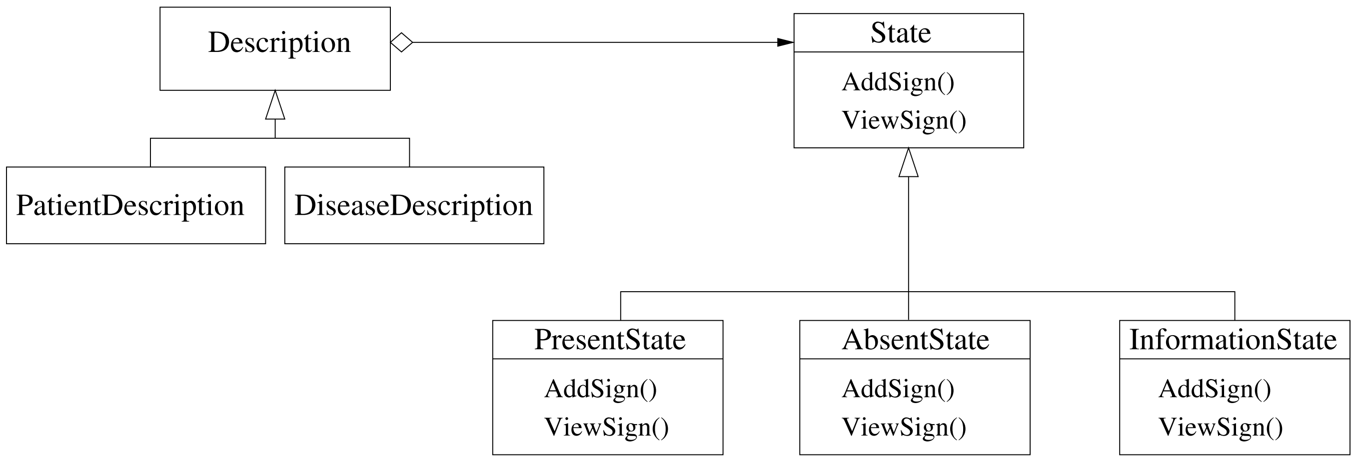

Figure 3: State of signs

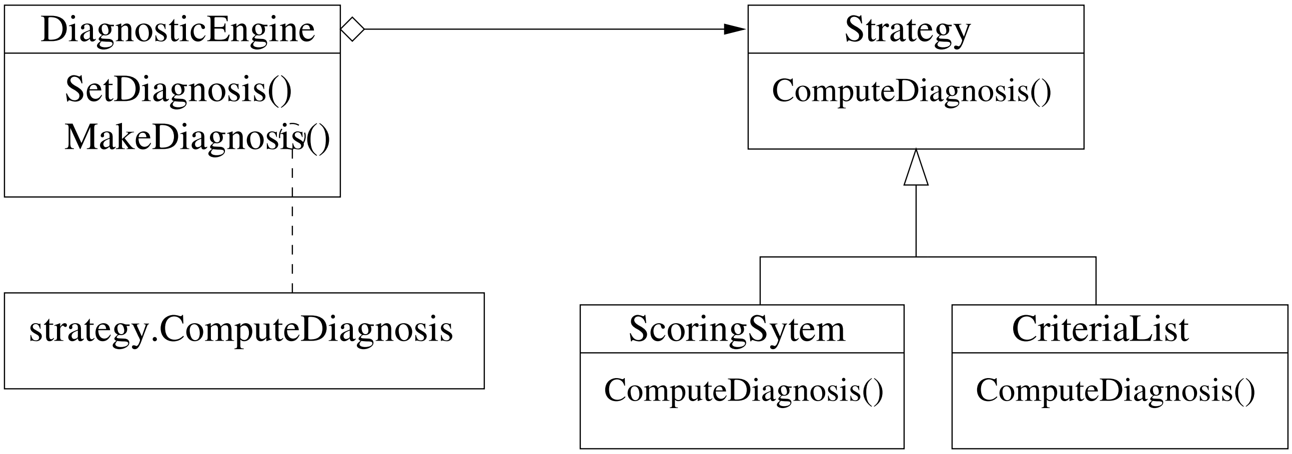

Figure 4: Scoring and criteria list strategies

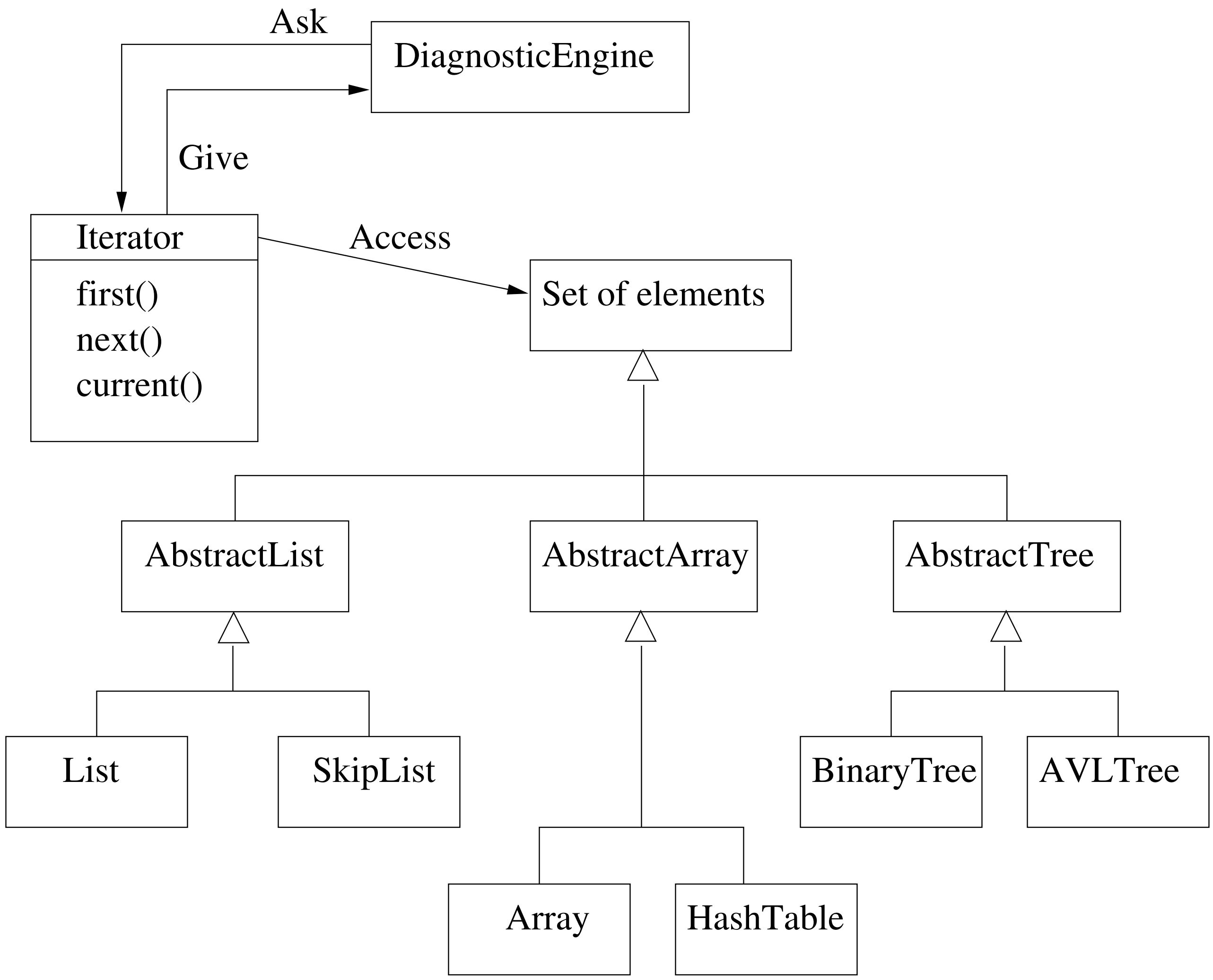

Figure 5: Use of Iterator

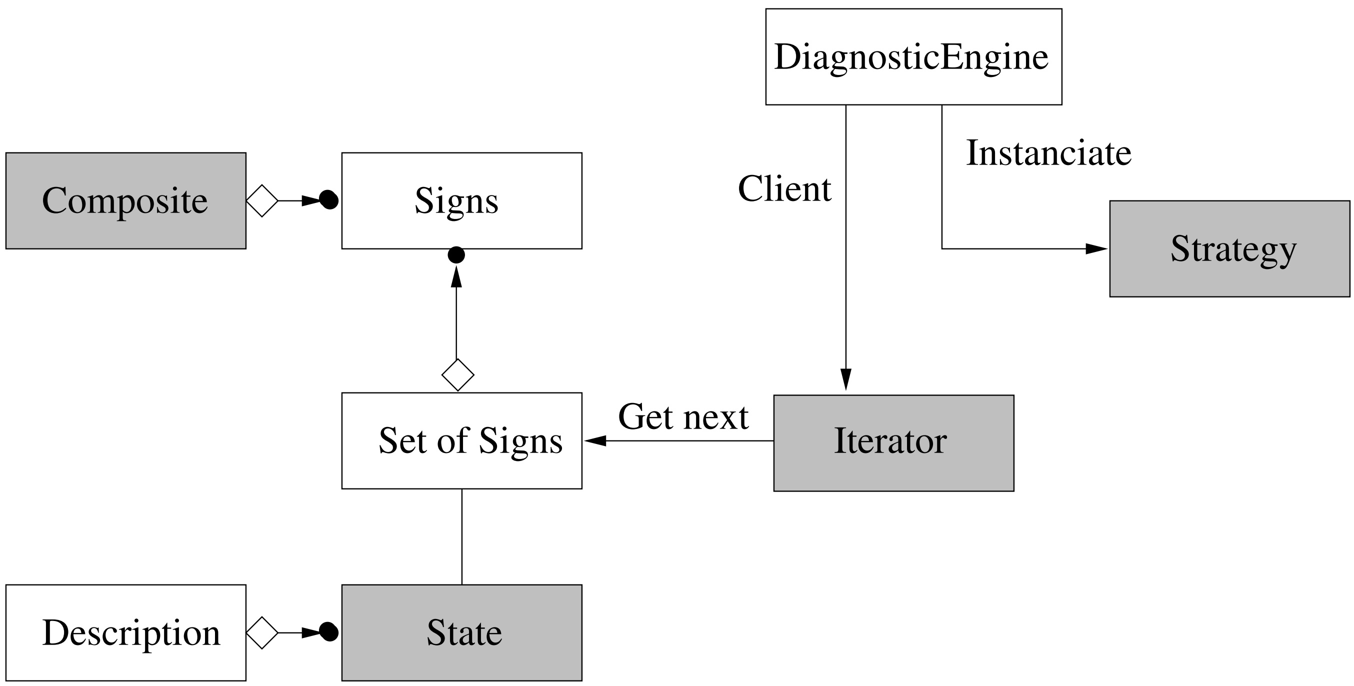

Figure 6: Association of the Object model and patterns

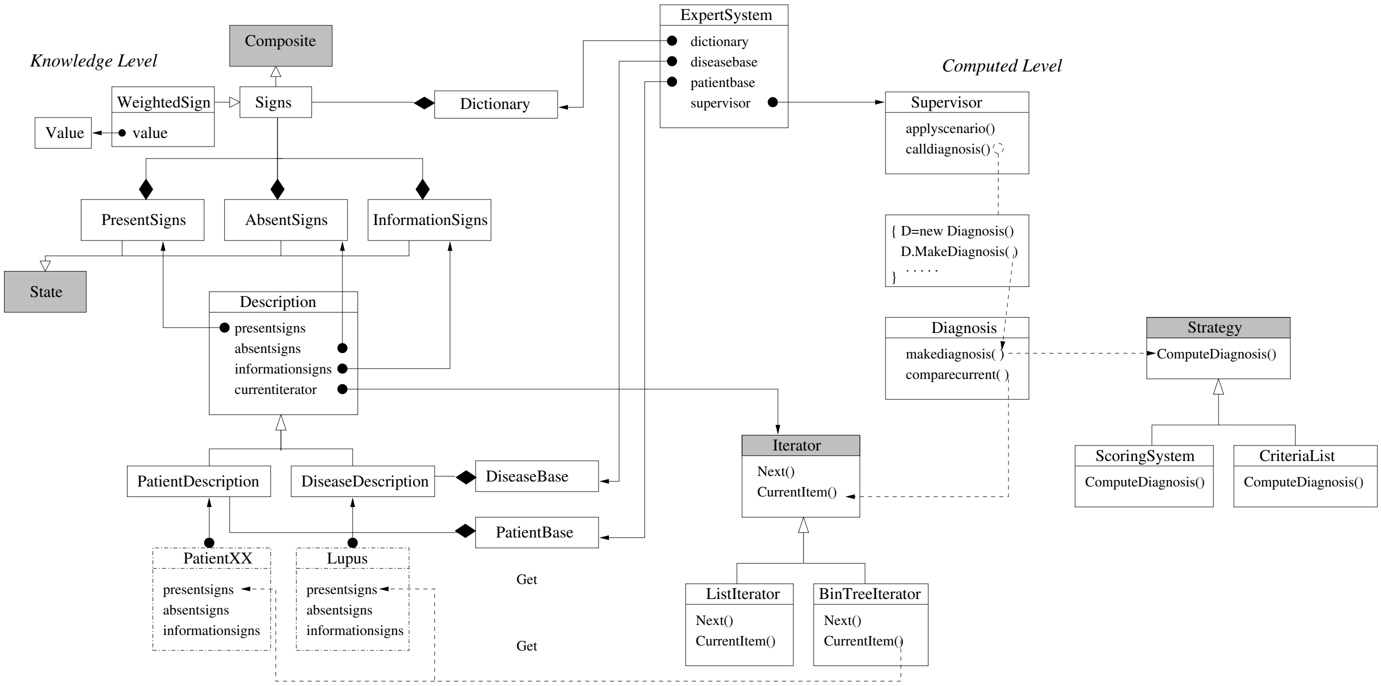

Figure 7: The architecture of LADRI

Pathologies	Nb. of signs retained				Maximum Score			
	Pres.	Abs.	Info.	**Total**	Pres.	Abs.	Info.	**Total**
Ankylosing Spondylitis	*19*	*3*	*7*	**29**	*88*	*21*	*39*	**148**
Psoriatic Arthritis	*18*	*3*	*7*	**28**	*77*	*22*	*40*	**139**
Reactive Arthritis	*18*	*3*	*6*	**27**	*72*	*21*	*31*	**124**
Rheumatoid Arthritis	*16*	*2*	*2*	**20**	*79*	*15*	*10*	**104**
Lupus	*19*	*1*	*2*	**22**	*91*	*7*	*12*	**110**
Dermatomyositis	*22*	*1*	*2*	**25**	*113*	*7*	*10*	**130**
Scleroderma	*21*	*0*	*2*	**23**	*110*	*0*	*10*	**120**
Rhyzom. PseudoPolyarth.	*7*	*4*	*2*	**13**	*44*	*29*	*12*	**85**
Horton Disease	*12*	*3*	*2*	**17**	*73*	*29*	*12*	**114**
Chondrocalcinosis	*15*	*4*	*2*	**21**	*81*	*28*	*12*	**121**
Gout	*12*	*3*	*2*	**17**	*69*	*21*	*11*	**101**
Still Disease	*16*	*1*	*2*	**19**	*71*	*16*	*10*	**97**

Figure 8: Scoring system

Pathologies	*Nb of cases*	*Reviewing (Score)*	*Confirmation (Criteria)*	*Definitely diagnosed by LADRI*
Ankylosing Spondylitis	10	9	for those 17,	10
Psoriatic Arthritis	5	3	10 cases are	2
Reactive Arthritis	2	2	validated	2
Rheumatoid Arthritis	22	21	19	21
Lupus	4	4	3	4
Dermatomyositis	4	3	2	3
Scleroderma	4	4	4	4
Rhyzom. PseudoPolyarth.	11	11	missing	11
Horton Disease	6	4	6	6
Chondrocalcinosis	5	4	missing	4
Gout	4	4	missing	4
Still Disease	1	1	0	1
Total	78	70	44	72

Figure 9: Diagnoses for 78 patients