1234-37

# Image Indexing by Content : An Object-Oriented Approach

P. Kofakis, A. Karmirantzos, Y. Kavaklis, E. Petrakis, S. Orphanoudakis[1]

Institute of Computer Science - Foundation of Research and Technology
P.O. Box 1385- Heraklion, Crete
GREECE

## 1. ABSTRACT

In this paper we present initial work on issues related to the design and implementation of an image database system for medical images. The objective of this project is the development of an environment that will permit the quick and intelligent access of images using their content. An image database is a complex system consisting of many components. There is an image analysis component that derives a representation of the stored images, we have a database component, and we have a user interface component that must be unified and integrated. The focus is primarily the justification of the design decisions, an illustration of the major features of such a system, and a requirements definition for its implementation. This paper will identify the requirements and propose an object-oriented database approach which meets these requirements. The capabilities of an object-oriented data model will be demonstrated and possible extensions, needed to meet the data modeling requirements will be discussed.

## 2. INTRODUCTION

The different components of an Image Database (IDB) system (i.e. image descriptors, raw image data, image processing procedures that are used to extract the image descriptors, display procedures, the DBMS, the storage and retrieval rules, etc.) are usually ***loosely coupled***. That is, there is a great distinction between data and the procedures that act on this data. An older version of our system [1] used this "loosely coupled" paradigm (fig. 1. ). Raw images were processed by an image processing module and image descriptors were obtained. These image descriptors were handled by a DBMS and used for the indexing of images. Queries were formulated using SQL and could be assisted by an expert system on top of the DBMS in order to define domain specific retrieval strategies (matching).

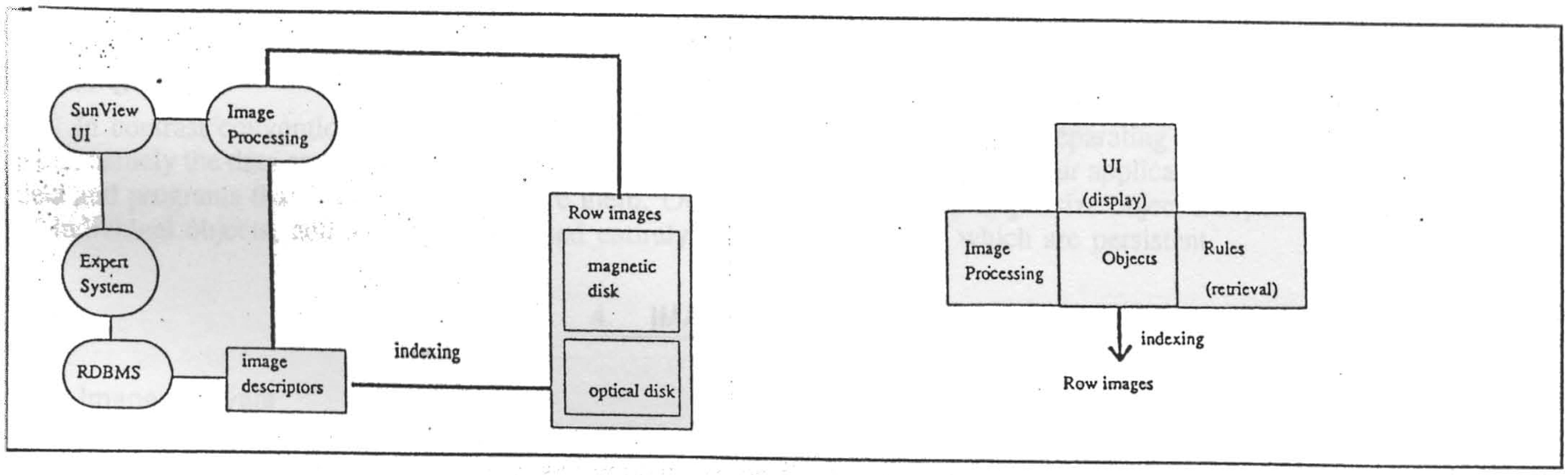

Figure 1. Loosely coupled versus tightly coupled approach

In this paper an object-oriented approach is proposed that will enable the integration of the various components of an IDB system. Thus data, the image analysis component, the database component, and the user interface and display component will be tightly coupled and considered as a single component using the object-oriented programming mechanism [2].

The object-oriented approach seems to be the most attractive one for the design and implementation of an IDB. It provides mechanisms for capturing the semantic content of an image in a highly modular and hierarchical manner at various levels of abstraction (use of abstract data types (ADT's) and principle of encapsulation). It also provides a framework for representing both image content and operations in a uniform way while the property of inheritance can be used to reduce the amount of stored data.

A number of multimedia filing systems have been developed which integrate graphical representation with a DBMS to broaden the bandwidth of information transfer between the computer and the user [3],[4]. A method of storing and manipulating images, based on the object-oriented approach, is discussed in detail in [5].

[1] Also affiliated with the Departments of Diagnostic Imaging and Electrical Engineering, Yale University, New Haven, CT.

The need for an interactive image analysis environment arises from the fact that image analysis methods are not always robust. Sometimes it is difficult to detect automatically interesting regions (i.e. tumors) or adequately segment parts of the image that represent soft tissues. Thus, we can automatically extract the "easy" or reference contours and then add manually interesting or ambiguous contours. Abnormal/pathological regions can then be defined interactively. To edit the results of automatic image segmentation or to define other regions of interest, the physician must be provided with a variety of tools which are easy to use. For example, the physician can interactively draw with a pointing device a curve which is an initial estimate of the contour of the region of interest and this crude estimate can automatically converge toward the correct boundary. He can also use a fluid-fill like tool to choose regions of interest : The user can click with the mouse at a point in a region of interest. This point is the seed for a region growing process that defines the region of interest. A slider and panel can be used for the setup of the parameters depending on the type of image to be processed. Filtering and image processing can also be performed in selected parts of the image (included in a contour).

## 4.2. Image content description

Upon completion of image analysis, an image description is available based on a collection of attributes and features at different levels. It is not trivial to determine the optimum representation based on which a given object can be compared with other objects in the database. Picture descriptions are generally given in terms of properties of objects contained within the picture and relationships among such objects. A picture description may also include properties or attributes of the picture as a whole. In order to describe image content for storage and retrieval purposes, we don't need a detailed description as we do for interpretation or diagnostic purposes where we need many details. A description that can adequately represent the important structures contained within the picture may be sufficient.

Features such as region, edge etc. and their spatial relations are combined to generate image descriptions (see fig. 2.). Selected features that can be used to represent the content of an image (slots of instances of segment objects) for image database applications are :

1) **Parts of segments** (subsegments): Length, Concavity (or convexity), Orientation, Position.

2) Segments (or objects) : They can describe objects alone (absolute) : Xmin, Xmax, Ymin, Ymax ; Area ; Perimeter ; Position ; Orientation (moments) ; Shape Factor ; Covering with find

its enclosing rectangle ; Compactness ; Elongation ; Shape Factor ; Fourier descriptors ; Texture class ; Gray value ; Color etc.

3) **Relations between segments :** Relative orientation ; Minimum distance ; Relative positions etc.

Images can be regarded as consisting of parts (segments) that in turn are composed of subparts, which have properties etc. Thus an image can be described as an a hierarchical structure using classes of "segments" that contain other classes of segments at a lower level of abstraction. The multilevel description of the image has the advantage of modularity and incremental construction of the system. This means that we can begin from a flat structure and incrementally add more and more attributes and features at different levels of abstraction. It is obvious that this kind of representation of image content uses complex internal structures and may comprise larger numbers of (possibly substructured) properties.

In this project an attempt is made to define and use extensions to the object-oriented model that will enable us to support semantic net like implementations. We must note that the encoding and implementation of "relations" is the major problem, because the complexity of the retrieval (matching) is a combinatorial function of the number of segments. The instantiation and aggregation relationships (Superclass and Subclass links) which a "segment" object may have are usually well formulated and implemented in an object-oriented environment. But there are many other relationships among objects or classes of objects which are needed in modeling an IDB application. Such relationships can be modeled as relationship object nodes. A relationship object may also have relationships with other objects. We limit ourselves to a binary relationship approach to conceptual modeling [11]. A binary relationship is called a link. The binary Relationship Model allows the specification of constraints : identifier constraints (type of link : one-to-one, one-to-many, many-to-many ) kind of objects that can be involved, mutual exclusion, equivalence etc.

To integrate these kinds of objects we can apply AI approaches [12]. A generic relationship may be defined between two generic (the class not the instance) objects as $G(m,i,j)$ which stands for the m-th generic relationship defined between the objects $O(i)$, $O(j)$. We can have directed or undirected relations. In the case of directed relations we can have two slots to represent both directions. Similarly, we can define instance relations between instances of objects.

## 4.3. Problems in retrieval - matching

Matching, i.e. finding a correspondence between a queried image and a set of images in the database images, even with the use of very simple attributes that are easy to compare (simple similarity metrics), [13] and very efficient matching techniques [14] working at the edge segments level, or techniques using hierarchical decision-tree matching, or matching process which works on hierarchical structures that capture increasing levels of detail about the objects of interest, leads to a

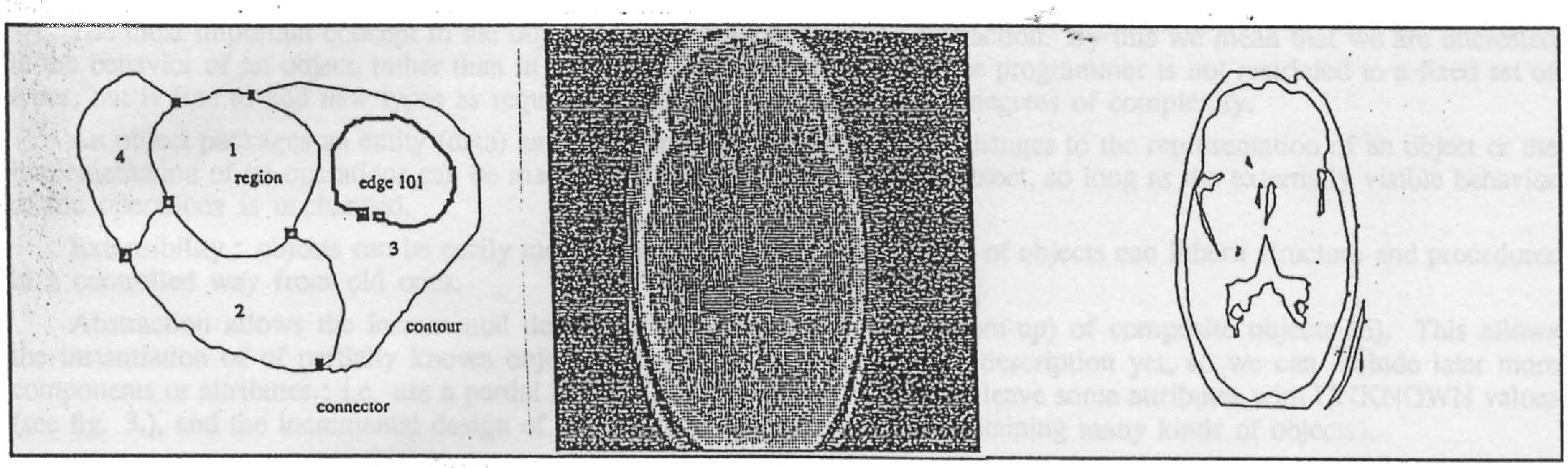

Figure 2. a) Image representation. b) Example image. c) Segments of example image.

combinatorial explosion in the case of large image database (100.000 to 1.000.000 images). The great number of segments can be related in a combinatorially explosive manner. Therefore the organization of the database should be such as to facilitate matching and retrieval operations :

- The number of possible candidate images should be minimized as much as possible. Pictures which are "similar" should be clustered together. Documents will be classified according to a hierarchy (classes) at various levels of specificity.

- The number of segments that would be checked for correspondence between the query image and the set of possible candidate images should be kept to a minimum. That means that we don't try to much every segment in the image, but using some knowledge either we restrict the matching in high level of abstraction (not full depth matching) for some parts of the image, either we try to match a limited subset of the image segments (try to match "differences"). Models should be hierarchical and object knowledge could be used to spatially constrain the search in the image, and limit the number of image abstractions. We can also limit the number of attributes for some specific segments (only relevant and discriminant attributes are used).

Modules that use specific techniques for matching could be implemented using the OO paradigm and activated using message passing at different levels of abstraction (levels of the hierarchy). A constraint-Based Graph Matcher [15],[16] that takes as input a data graph whose nodes are objects and whose arcs are attributes of the objects, and relations between these objects, and a pattern graph which describes a part of a given image and verifies if there is a match or not. The matcher could be invoked when both graphs are small.

**Query : possible scenarios :**

- The object-oriented model is associated with an interactive user-friendly environment. The model itself encourages and simplifies the concept of pointing at an object to find out the allowable operations on the object and then select a desired operation. A number of different modes of access are outlined below :
- Access to an object through its class.
- Access through the aggregation hierarchy. (eg. requests for descendants ).
- Access through relationships (relationships can be used to move from one object to another).
- Access through specific features of intrinsic data (eg. search for specific patterns).
- Combination of above.

In most cases users don't know what they are looking for. When a user is opening a document ("object") he can look at it at various levels of abstraction (resolution) and open iteratively lower levels of abstraction, navigate to other objects etc. A user wants the query language to assist with "opening up" complex objects and searching for qualifying subobjects. He does not want an operator for each particular search. Various searching techniques have been proposed and developed to improve the performance of the systems.

***Indexing*** collections of objects provides extremely fast retrieval by avoiding long, sequential searches.

***Clustering*** allows objects to be physically placed on disk in a way that minimizes retrieval time for specific data access patterns.

## 5. MAJOR OBJECT ORIENTED CONCEPTS/CAPABILITIES

We describe the most significant Object Oriented concepts/capabilities for the specific application :

***Data abstraction*** :

The most important concept in the object-oriented-approach is data abstraction. By this we mean that we are interested in the behavior of an object, rather than in its representation. Further more the programmer is not restricted to a fixed set of types, but is free to add new types as required and define objects at various degrees of complexity.

An object packages an entity (data) and the operations that apply to it. Changes to the representation of an object or the implementation of its operations can be made without affecting users of the object, so long as the externally visible behavior of the operations is unchanged.

Extensibility : objects can be easily modified or specialized. New classes of objects can inherit structure and procedures in a controlled way from old ones.

Abstraction allows the incremental design (possibly top-down and bottom-up) of composite objects [6]. This allows the instantiation of of partially known objects (we do not know their entire description yet, so we can include later more components or attributes : i.e. use a partial list of all the possible attributes, or leave some attributes with UNKNOWN values (see fig. 3.), and the incremental design of large objects (complex images containing many kinds of objects).

We can define a set of basic objects (edges, curves, regions, arrays, quad trees etc) and then combine these basic objects to define more complex objects. A region can be defined as a complex object consisting of other regions or other curves. When a representation is chosen the object inherits from its superclasses all procedures and relevant messages and can immediately be manipulated by other objects of the system. The message passing mechanism can be used for the control of the system.

***Encapsulation*** : Data and procedures (called methods) are packaged together in a single, autonomous structure called an object. Data and operations are modeled at the same time. For each object there is an interface part and an implementation part. The interface part is the only part of the object visible outside it and includes the specification of the operations that are allowed on the object. The implementation part, includes a data part that describes the structure and the properties of the object and an operation part which consists of the implementation of the operations. Encapsulation of data with procedures significantly reduces the complexity of large systems by minimizing the interaction between objects.

***Independence (object identity)*** : Every object has a unique identity, independent of its value, that uniquely characterizes it. Two objects can be identical only if they are the same object. They are equal (not identical) if they have the same value.

***Schema evolution*** :

Traditional databases allow very little flexibility for evolution of their schema (classes). Relational systems sometimes permit adding attributes. However, dropping attributes or moving them to other relations is seldom permitted. I.e. the schema is static

Schema change operations fall into three categories :

- changing class definitions, i.e. instance variables or methods.
- modifying the class lattice by changing the relationships between classes.
- adding or deleting classes in the lattice.
- incremental design of objects and database schema :

***Types and Classes (Classification)*** : If every object is to carry its own attribute names and methods, the amount of information to be specified and stored can become unmanageably large. For this reason 'similar' objects are grouped together into a class. Objects that belong to a class are called instances. The concept of class hierarchy reduces information redundancy. A class can have more than one superclass (generalization).

Databases traditionally have very few classes, with large numbers of instances per class. Classification in (relational) databases serves mainly to provide means to efficiently manage large amounts of data by differentiation between entities by attribute contents. Classification in an object-oriented system supports instantiation, encapsulation and class inheritance.

***Inheritance*** : A class inherits properties and methods from other classes (single or multiple inheritance). Further, the inheritance mechanism makes it possible for applications to define new classes and have them inherit properties (data and operations) from existing classes; this makes the application easily extendible.

***Passive/Active objects, message passing etc.*** : Conventional databases have always viewed data objects as being passive. (Active objects can be considered as processes, each with its own script of actions waiting to be activated). Operations are performed from outside. One of the most interesting aspects of OODB is that the objects can be viewed as active agents. They can respond to requests (messages) from other objects, but they can also trigger themselves (eg. constraint checking when we enter a value etc.).

***Navigation*** : OODBs can support point-to-point navigation between instances in the database. Thus it is easy to access objects with many instance-to-instance relationships when they are stored in a database. This can help queries and access of objects (browsing, iterative query etc.).

***Overriding and late binding*** : A name can be used for different operations. In an object-oriented system, an operation is defined for the most general type. The body of the operation is defined for each instance of the generic type (overriding).

***Virtual objects*** : A virtual object looks and acts like any other object in the environment. However, the procedures underlying its access messages modify the descriptions of other objects without duplicating their underlying representation. In other words a virtual object is just a filter around the messages of other objects. A simple example is a virtual image (G_GRAD) which is the the smoothed gradient of an existing image and is used in the interactive contour detection. Virtual objects can also map over localized neighborhoods in the source object (gradients, convolution, and median filtering are all possible without creating intermediate images). There is a speed disadvantage, however there is a tremendous space efficiency. Virtual objects are also very important in a display system where minor modifications of existing objects are desired to tune the appearance of the display. Here virtual images could be used as filters between images coded in different formats that are not directly displayable on the screen. Virtual objects can also provide an excellent interface to various types of display hardware. For example display of an 8-bit image to a monochrome display. Virtual curves and regions can also be created. For example the zoomed part of a contour or the polygonal approximation of a contour could be virtual objects.

**Dynamic classification,** i.e. mechanisms for the propagation the changes performed in the schema. Once schema modifications are performed, their impact on the relevant classes are characterized. For each modified class, the changes are defined and tested on its relevant classes. Should all these changes be correct with respect to the model and the semantic rules (background knowledge), the corresponding sets of instances are modified. This means that modifications can be propagated on whole sets of instances belonging to relevant classes as atomic operations. Changing a constraint in a class implies that instances of this specific class may also become instances of another class. (eg. assume that the perimeter constraints on various contour subclasses are as follows : SMALL : per<= 20, MEDIUM : 20<per<100, LARGE : 100<=per<=200, VERY_LARGE : per>200. Changing the constraint in the VERY_LARGE class from 200 to 180 implies that some instances of LARGE will also be instances of VERY_LARGE ).

## 6. KNOWLEDGE REPRESENTATION

### 6.1. Data representation

The data representation model used in our work is the Frame model, which is one of the essential knowledge representation techniques used in AI [17],[18],[19]. The primitive element is called unit. Each unit has a name and consists of a number of slots, which in turn consist of a list of aspects called facets. Slots are used for describing the unit they belong to (attributes). Facets are needed for proper specification of the slot and its value. Thus units are formed by an aggregation of slots and facets. A unit can be a class-object or a member-object. Relations (links) relate member-objects to class-objects. As data structures frames benefit of slot access procedures (active values in KEE), and value inheritance. The frame is a control mechanism as well as a data representation. The slots may have constraints of their own (cardinality, value class, etc), or constraints that reference other slot instances or other objects implemented in the form of active values or rules (applicable rules facet). Figure 4. shows some slots of a segment instance description in KEE.

### 6.2. Rules

The system should allow the integration of background knowledge (knowledge about the specific domain), like constraints, assertions that describe a class, procedures and production rules that change and manipulate the schema (eg. if the attrM = X then it belongs to class N) etc.). The use of existing knowledge will help us to eliminate many search paths, so the response time during retrievals will be shorter.

Knowledge is encapsulated in ***if-then*** rules which interact with the body of data [20]. Rules are also consider as objects.

With rules, many complicated pieces of knowledge can be expressed in a flexible way by splitting them to many small and simple rules. Because of the large number of rules additional control has been added by structuring the rule base into classes, and attaching methods to specific classes of the system via the APPLICABLE.RULES facet. Using rules, we can build several models (like the brain has one large cancer or two small cancers). After the initialization of the models, we can examine them, and decide which model fits best to our situation. Thus, we can finally find the exact model of the brain.

**Initialization - Classification rules :** They use apriori knowledge to classify segments of the image [21].

**Consistency rules :** One of the basic function of the knowledge is to check the consistency of our data. This class of rules applies spatial and contextual constraints to update or modify values of slots for the segments of a specific image. They also can check consistency of the database schema and used for constraints propagation or dynamic classification (database schema evolution). For instance, update rules can be provided for the creation, modification and deletion of objects to allow semantic integrity maintenance.

```
If ((?segment1 contains ?segment2)
    and (?segment2 contains ?segment3))
  then (?segment1 contains ?segment3)
```

```
Unit : S1.1
Superclasses : SEGMENTS
Member of : SKULL
-------------------------
Own Slot : HISTORY from SEGMENTS
 Inheritance : OVERRIDE.VALUES
 Value Class : (UNION IMAGE.PROCESSING IMAGE.DISPLAY)
 Cardinality.Min : 1
 Cardinality.Max : 100
 AvUnits : UNKNOWN
 Variable Type : MULTIPLE.ARGUMENTS
 Applicable Rules : UNKNOWN
 Values : FMOY (FFIL 20) (TEXT 5)

Own Slot : TEXTURE from SEGMENTS
 Inheritance : OVERRIDE.VALUES
 Value Class : (ONE.OF BLANK SHADED STRIPED CROSSED WAVY)
 Cardinality.Min : 1
 Cardinality.Max : 1
 AvUnits : TEXTURE.AV
 Confidence : .9
 Time Stamp : Nov 1 09:20:03
 Variable Type : NOMINAL
 Applicable Rules : TEXTURE.RULES
 Values : UNKNOWN

Own Slot : DISPLAY from SEGMENTS
 Inheritance : OVERRIDE.VALUES
 Value Class : (UNION IMAGE.PROCESSING IMAGE.DISPLAY)
 AvUnits : UNKNOWN
 Variable Type : MULTIPLE.ARGUMENTS
 Applicable Rules : UNKNOWN
 Values : UNKNOWN
```

Figure 3. Some slots of a segment instance description in KEE.

**Strategies :** Strategies are control structures that will be implemented in the form of rules. They will be used to encode knowledge about which kind of methods to apply and in what order (eg. sequence of image analysis tasks)..

**Classification rules :** These methods could be used for the classification of objects, We can use rules of a rule-based system to propagate changes and operations on some entities to other entities to which they are related (for example propagate operations from a complex object to its components, and vice versa ), rules for specifying how and when to check constraints, rules for invoking exception handlers

**Retrieval rules :** These rules encode knowledge and strategies for horizontal and vertical access, the retrieval strategies for a specific class of images (e.g in order to find images matching a specific image of the brain we should first identify some reference contours (skull) and then use for the retrieval relations of a segment that represents a pathological case (UNKNOWN class) and a reference segment).

### 6.3. Active values

Active values are pieces of code which are executed when a specific change in data (slot value ;, see fig. 3.) takes place. We can use this code to check for consistency, update something else in the database or simply for debugging purposes.

The combination of active values with rules can facilitate a lot the construction of the system's knowledge-base (see fig. 4.). We can attach an active value to specific data of our system, and when these data change the necessary rules are triggered so that the state of our knowledge will be always consistent.

## 7. DESCRIPTION OF THE SYSTEM

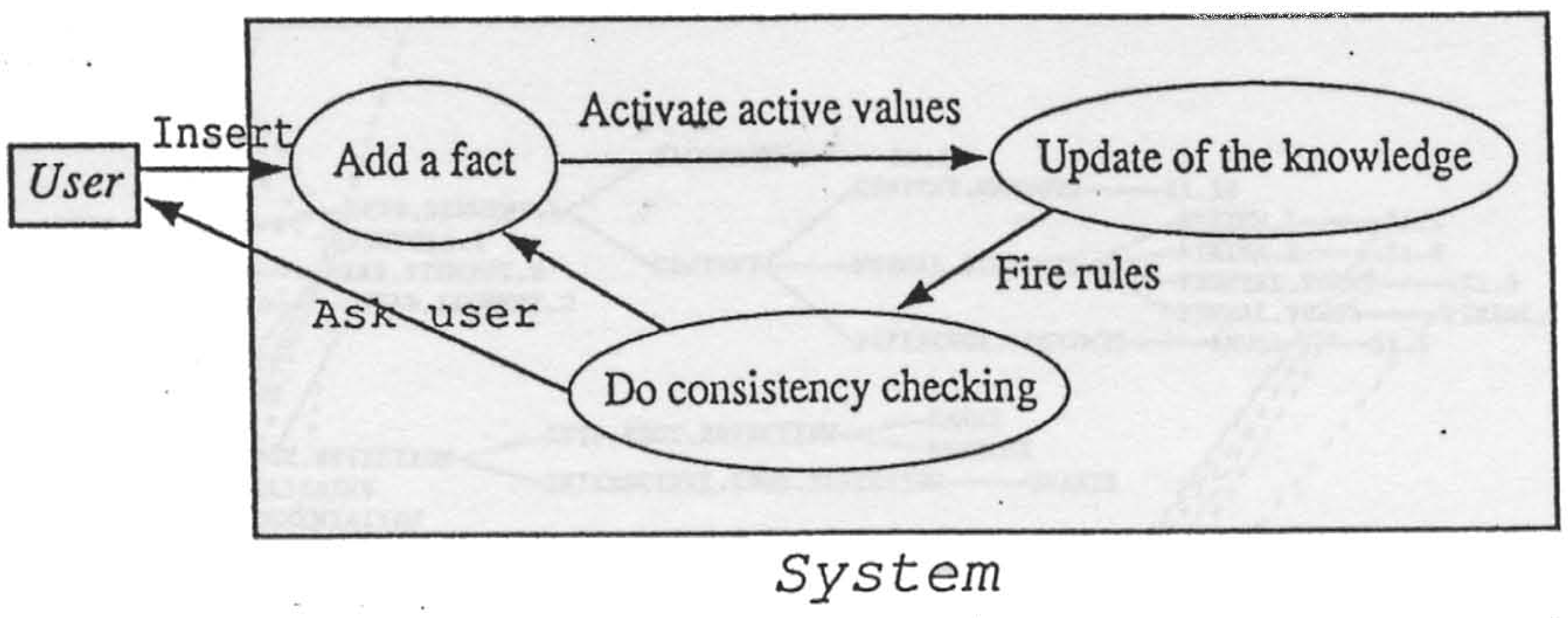

Figure 4. The actions taken for the consistency checking.

## 7.1. Logical Organization

The database is organized hierarchically, with only large-grained or high level of abstraction objects at the top level. Figure 5. shows the organization of the system. The low level of abstraction is at the right part of the figure.

ACCESS : defines the different classes of users, time and location organization : data, procedures (methods, rules etc) and parameters (for display, processing and retrieval) can be inherited to a specific class of image content.

IMAGE.PLANE : the top levels are directly obtained from the image header. The lower represent classes based on the image content, and classes of image segments for a specific class of images.

IMAGE.TOOLS : procedures for the display and processing (interactive and automatic) of images are considered as complex objects (methods + parameters + rules).

MODALITY, OPERATIONS : like ACCESS the data, procedures and parameters can be inherited to a specific class of image content.

RULES : see previous paragraph on "knowledge representation".

SEGMENTS : The instances of this class represent image segments (refer to previous paragraph on "image representation" (attributes : slots).

SEGMENTS.RELATIONS : representation of the image segments relations (see previous paragraph"image content description").

**Memory Organization :**

*Long term :* This part of the system contains descriptions of all permanent objects (definitions of classes, rules, methods etc) and is the "static" part of the system. For efficiency it is resident in virtual memory [8]

*Short term :* This part of the system includes the large amount of objects that are instances of a class (leaves) and is the part of the system that changes dynamically. For space efficiency a DB implementation is needed.

## 7.2. User Interface

The system will be utilized by clearly different stereotypical categories of users (e.g. radiologists, referring physicians, physicists, technicians, etc.) with diverging user-interface requirements. The same holds for diverse categories of goal-oriented tasks which range from simple (routine) to complex (problem solving) procedures. From the system designer's point of view, user interfaces have to hold and access explicit, formalized models of users and and tasks which define major parts in the interaction system.

Object-Oriented techniques facilitate the implementation of specific display and user interface methods for different classes of users and images which will be inherited accordingly. It also supports extensibility. That means that the system can support new devices and functions on the images and pictures of the database. For example a new color display device may be added to the system with relative ease, if at a high level of abstraction the color display can be viewed as a more specialized presentation device than a more general display device already supported by the system.

**Display :**

The display system can be build around a family of messages. Each method (routine) may contain two main pieces : an iteration scheme for scanning and accessing the object using the basic messages, and the code needed to display the object(s) on the screen(s). A small set of display functions can be used on a wide variety of objects. The use of virtual objects enhances

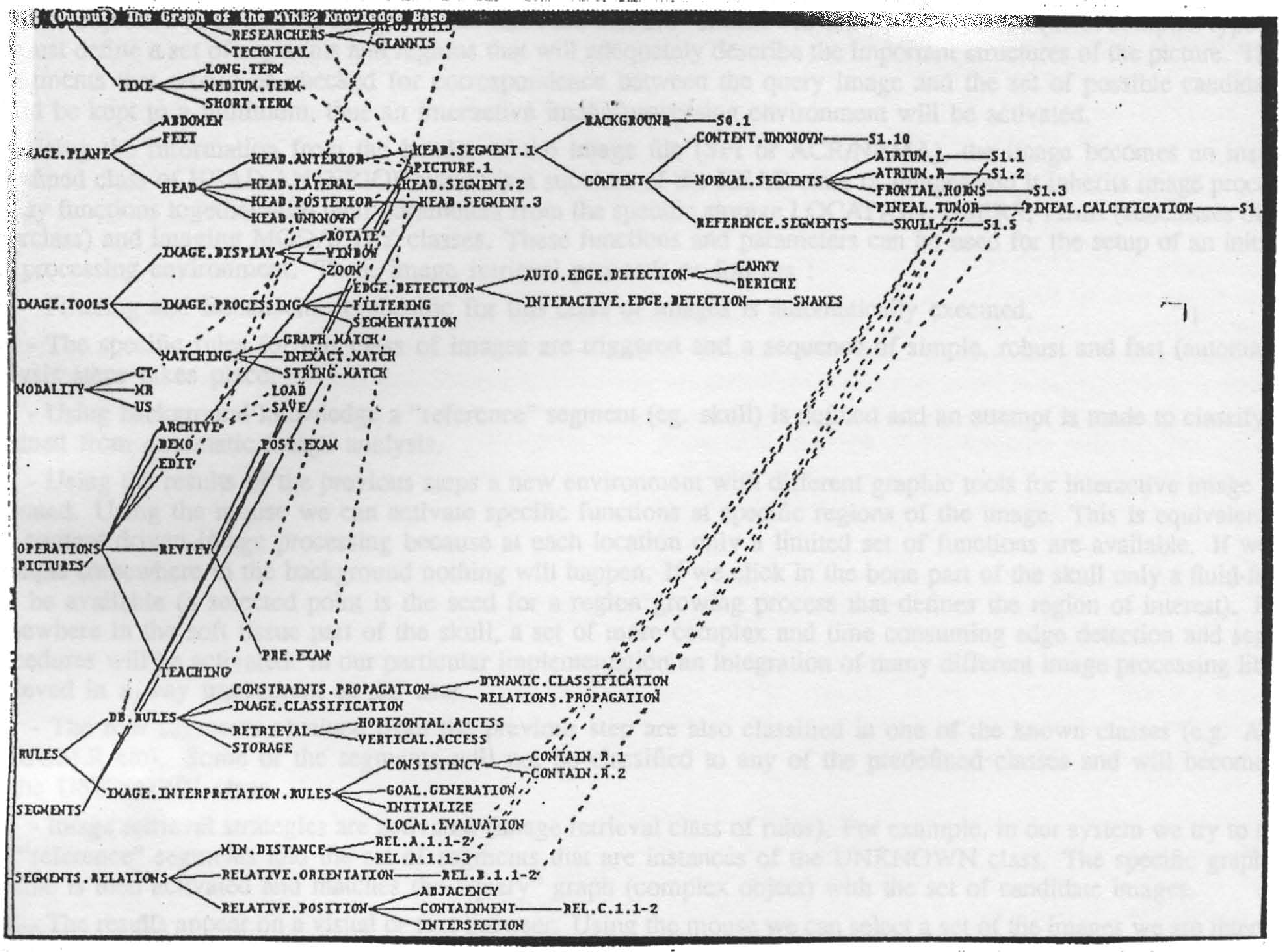

Figure 5. Organization of the system.

and simplifies the display system. Rather than having options in the display routine, a virtual object can be created that captures the desired functionality. The different parameters of the display routines could be stored in the database hierarchy.

**Browsers :**

Browsers are an interactive query mechanism for finding and exploring objects and relationships between objects [22]. Three different types of browsers can be used for the access and retrieval of images: text oriented browsers, image browsers, and graph browsers. All of the browsers maximize the amount of information (vertical or horizontal) presented at one time and provide a means to interactively apply functions to select objects.

*Text browser :* the text oriented browser works with any object and provides an easy interface to interactively apply functions. The display can be done horizontally : display one object at a time with one field per line, maximizing the depth of information displayed about a particular object, or vertically when we are looking for a specific set of objects.

*Image browser :* The image browser displays reduced resolution pictures of objects. Usually a text browser is used to narrow down the number of images to browse with an image browser.

*Graph Browser :* the graph browser displays objects and the relationships between these objects (e.g. parent and child links, relationship objects etc). The KEE system has built in text and graph browsers.

## 7.3. Retrieval Example

To illustrate the possible use of the system under development consider a CT image of the brain (refer to fig. 5. ). Its description can be given in terms of a complex object consisting of other complex objects (directed tree graph). At the lower level of abstraction, the description consists of parts of segments with their attributes. Relationship objects represent the different relationships between the different structures in the picture. Some parts of the image (eg. skull) are relatively stable (detected without ambiguity) and can be used as "reference" segments. These "reference" segments can have their own display methods and may activate special methods for image analysis for the part of the image they enclose.

Our objective is to find all images in the database that are "similar" to this specific image (most complex type of query). We must define a set of segments and regions that will adequately describe the important structures of the picture. The number of segments that would be checked for correspondence between the query image and the set of possible candidate images should be kept to a minimum, thus an interactive image processing environment will be activated.

Using the information from the header of the image file (SPI or ACR/NEMA), the image becomes an instance of a predefined class of HEAD.ANTERIOR which is a subclass of the HEAD class of images and it inherits image processing and display functions together with their parameters from the specific storage LOCATION, USERS, TIME (subclasses of ACCESS superclass) and imaging MODALITY classes. These functions and parameters can be used for the setup of an initial display and processing environment. Then, image retrieval proceeds as follows :

- Filtering and Enhancement, specific for this class of images is automatically executed.

- The specific rules for this class of images are triggered and a sequence of simple, robust and fast (automatic) image analysis steps takes place.

- Using background knowledge a "reference" segment (eg. skull) is defined and an attempt is made to classify segments obtained from automatic image analysis.

- Using the results of the previous steps a new environment with different graphic tools for interactive image analysis is activated. Using the mouse we can activate specific functions at specific regions of the image. This is equivalent to model and content driven image processing because at each location only a limited set of functions are available. If we click for example somewhere in the background nothing will happen. If we click in the bone part of the skull only a fluid-fill function will be available (a selected point is the seed for a region growing process that defines the region of interest). If we click somewhere in the soft tissue part of the skull, a set of more complex and time consuming edge detection and segmentation procedures will be activated. In our particular implementation an integration of many different image processing libraries was achieved in a way transparent to the user.

- The new segments obtained from the previous step are also classified in one of the known classes (e.g. ATRIUM.L, ATRIUM.R etc). Some of the segments will not be classified to any of the predefined classes and will become instances of the UNKNOWN class.

- Image retrieval strategies are activated (image retrieval class of rules). For example, in our system we try to match only the "reference" segments and the set of segments that are instances of the UNKNOWN class. The specific graph matching module is then activated and matches the "query" graph (complex object) with the set of candidate images.

- The results appear on a visual or text browser. Using the mouse we can select a set of the images we are interested in. If we are not satisfied we can repeat the image analysis step to define a more complete description of our image for iterative query.

## 8. DISCUSSION

Possible enhancements of this work include :

**Use of Machine Learning :**

1) Information from the image header can be used for the classification of images. This information is in an unstructured form, thus classical classification techniques [23] could be used.

2) Classification of segments for a specific class of images (e.g. head tomograms). The objective is to obtain a decision tree or rules for the classification of segments (e.g. ventricle segment). The attributes that describe a segment are in an unstructured form, thus classical classification techniques could be used.

3) Derive automatically structural descriptions of complex image classes. The existence of relational and structural attributes requires the use of an inductive learning, INDUCE, AQ15 [24][25] or similar type of algorithm, to obtain the characteristic and discriminant description of a limited number of image classes (some classes defined by the user plus one UNKNOWN).

4) Construct automatically meaningful classifications of images using their content. Conceptual clustering techniques in order to generate classes together with their taxonomic descriptions seem appropriate for this application.

**Attribute selection :**

The determination of the relevant attributes, their proper encoding [26], representation and structuring is a major problem. Various methods of attribute selection [23] such as factor analysis, multidimensional scaling, factor analysis, data standardization and data (linear) transformation have been used to approach the problem of selecting the most relevant set of attributes to describe objects. These methods are most effective for numeric attributes when the size of the event set is statistically significant. When attributes are symbolic and the event set is small, these methods are not adequate. Some novel methods [27] can deal /handle adequately the many-valued, nominal variables which occur often in human classifications.

**Data Modeling for 3D applications :**

The modeling and representation of 3D objects in a database could be a future issue of an image database. (instead of 2D tomograms we can have a set of successive tomograms (slices) and a 3D representation of the body).

We can use the Delaunay triangulation structure for the description of a 3D object. In this representation the object is known by the 3-coordinates of a set of points selected on the boundary (surface) of the object. The structure between these surface points is the Delaunay triangulation (graph of 3- polyhedrons with the measured points as vertices), and it is a minimal one. So we obtain a representation of the object's surface *S* as a set of triangular facets (polyhedron which is the boundary of the set of tetrahedra). Details on the Delaunay triangulation can be found in [28].

The object-oriented system could support the modeling of such complex objects [18],[29]. These objects are seen and manipulated at different levels of abstraction. At higher levels they are treated as atomic units of data which are described by several attributes. At lower levels they reveal their internal structure. Their components may be complex objects, or just primitive objects without internal structure. Different complex objects may share components, for instance a 3D object may share part of its surface (glued to) with another object (here we may have a many-to-many relationship), or a complex object may be composed of objects of the same type (recursive decomposition of a 3D object into subparts). Complex objects are thus represented by a connected, directed, and acyclic type graph as a subgraph of the database schema. This graph becomes cyclic in the case of recursive objects [30]. It is obvious that we have to cope with objects that change dynamically depending on the actual view of the application. For example an object that represents a 3D object can use a boundary (surface) or a volume representation [31].

## 9. IMPLEMENTATION ENVIRONMENT

Programming constructs and environments for an image database application have two conflicting requirements : they must be as much general and offer many levels of abstraction and be efficient in the same time. Having a programming environment that satisfies the first requirement (eg. KEE) maximizes the productivity and allows the implementation at a conceptual level without concern for its efficient implementation. Object-oriented languages like C++ generate efficient code which is also portable on many machines.

We decided to implement our system using the C programming language (for the image processing software), Common-Lisp and Knowledge Engineering Environment (KEE) under the Unix operating system. Advanced software tools, such as KEE [19], provide an appropriate environment for the development of such a system. In KEE the fundamental representation scheme is frames, actually called units, which unify the procedural and declarative expression of knowledge. KEE is an example of object-oriented programming. The whole knowledge base is represented in a set of units. Facts and rules in KEE are represented as objects or frames that have labeled slots containing either values or means for obtaining values (methods written in LISP). Similarly, a slot may contain a set of rules that conclude values for other slots. I.e. even rules and Lisp procedures are units. KEE integrates frame-based and rule-based reasoning techniques to describe structures and behaviors.

## 10. ACKNOWLEDGEMENTS

This research was supported in part by the Commission of the European Communities under project AIM-1008. We would like to thank the members of the PRIMIS group (AZ-VUB Brussels) who provided most of the medical images.

## References

[1] S. C. Orphanoudakis, E. G. Petrakis, and P. Kofakis. A medical image database system for tomographic images. In *Proceedings of the International Symposium CAR'89*, pages 618–622, Berlin, June 25-28 1989. Springer - Verlag.

[2] D.C. Tsichritzis. Fitting round objects into square databases. In *Proceedings ECOOP '88 (Oslo)*. Springer-Verlag., 1988.

[3] D. Woelk, W. Kim, and W. Luther. An object-oriented approach to multimedia database. In *Proceedings ACM - SIGMOD '86, International Conference on Management of Data*, pages 311–325, Washington, D.C., May 1986.

[4] Kim W. Woelk D. Multimedia information management in an object-oriented database system. In *Proc. of the 13th VLDB Conference*, pages 319–329, Brighton, 1987.

[5] L. Mohan and R.L. Kashyap. An Object-Oriented Knowledge Representation for Spatial Information. *IEEE Trans. on Software Eng.*, 14, No 5:675–681, 1988.

[6] Nguyen G. and Rieu D. Schema evolution in object-oriented database systems. In *INRIA research report No 947*, December 1988.

[7] I.R. Young D.A. Bell, D.H.O Ling. Medical image databases : State of the art. In *Internal document , HIPACS AIM project deliverable 01*, August 1989.

[8] Thatte S.M. Persistent memory : A storage architecture for object-oriented database systems. In *1986 Int. workshop on Object-Oriented Database Systems*, pages 148–158, Asilomar-California, Sept. 1986.

[9] Gupta R., Cheng W.H., Gupta R., Hardonag I., and Breuer M. An object-oriented vlsi cad framework : A case study in rapid prototyping. In *IEEE-Computer*, pages 28–36, May 1989.

[10] J. Johnson, T. Roberts, W. Verplank, D. Smith, C. Irby, M. Beard, and K. Mackey. The xerox star : A retrospective. In *IEEE Computer,Vol. 22, No 9*, pages 11–29, September 1989.

[11] O. De Troyer, J. Keustermans, and R. Meersman. How helpful is an object-oriented language for an object-oriented database model ? In *1986 Int. Workshop on Object-Oriented Database Systems*, pages 124–132, Asilomar-California, Sept 1986.

[12] Hiromichi F., Atsushi H., and Jun'ichi H. A personal universal filing system based on the concept-relation model. In *Proc. of 1st Int. Conf. on Expert Database Systems*, pages 31–40, Charleston, April 1986.

[13] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice Hall, 1982.

[14] Ayache N. Efficient registration of stereo images by matching graph descriptionms of edge segments. In *INRIA research report No 559*, August 1986.

[15] M.A. Eshera and K.S. Fu. An image understanding system using attributed symbolic representation and inexact graph-matching. In *IEEE - Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 5*, September 1986.

[16] Wen-Hsiang Tsai and Shiaw-Shian Yu. Attributed string matching with merging for shape recognition. In *IEEE - Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No.4*, July 1985.

[17] Minsky M. A framework for representing knowledge. In P.H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill Book Company, 1975.

[18] M. Mmitschang. Towards a unified view of design data and knowledge representation. In *Proc. of 2nd Int. Conf. on Expert Database Systems*, pages 33–49, Sheraton Premiere at Tysons Corner, VA, 1988.

[19] IntelliCorp. *KEE Software Development System - User's Manual*, 1988.

[20] B.A. Draper, R.T. Collins, J. Brolio, A.R. Hanson, and E.M. Riseman. The schema system. In *International Journal of Computer Vision, 2*, pages 209–250, 1989.

[21] D.M. McKeown, W.A. Harvey, and J. McDermott. Rule-based interpretation of aerial imagery. In *IEEE Transactions on PAMI, Vol. PAMI-7, No. 5*, September 1985.

[22] C. McConnell, P. Nelson, and D. Lawton. Constructs for cooperative image understanding environments. In *Proceedings : Image Understanding Workshop*, pages 497–506, February 1987.

[23] B. Everitt. Cluster analysis. In *Gower, Halsted Press*, 1986.

[24] R.S. Michalski, I. Mozetic, and N. Navrac I. Hong. The AQ15 inductive learning system: an overview and experiments. In *Intelligent System Group, Dept. of Comp. Science, Un. of Illinois*, 1986.

[25] J.B. Larson. Inductive inference in the variable-valued predicate logic system VL21. In *PhD dissertation, Un. of Illinois*, 1977.

[26] Connell J. and Brady M. Generating and generalizing models of visual objects. *Artificial Intelligence Journal*, 31:159–183, 1987.

[27] P. Baim. A method for attribute selection in inductive learning systems. In *IEEE Transactions on PAMI, Vol. PAMI-10, No 6*, Nov. 1988.

[28] J. D. Boissonnat. Representing 2D and 3D shapes with the Delaunay triangulation. In *7th International Conference on Pattern Recognition*, pages 745–748, Montreal, July 1984.

[29] Batory D.S. and Buchmann A.P. Molecular objects, abstract data types and data models : a framework. In *Proc. of 10th VLDB Conf., Singapore*, pages 172–184, 1984.

[30] B. Mitschang (G. Goos C. Hubel and J. Hartmanis eds.). *Object Orientation within the PRIMA-NDBS*, pages 98–103. Springer-Verlag, 1988.

[31] M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1988.