KRISYS - A MULTI-LAYERED PROTOTYPE KBMS
SUPPORTING KNOWLEDGE INDEPENDENCE

Nelson Mendonca Mattos

University Kaiserslautern, Department of Computer Science,
P.O. Box 3049, D-6750 Kaiserslautern, West Germany
e-mail: mattos@uklirb.uucp

Abstract

paper discusses architectural issues of Knowledge Base Man-
ent Systems and describes the architecture of KRISYS, a sys-
whose goal is the effective and efficient management of large,
d knowledge bases. Focal points are primarily the design de-
s and the system's features: knowledge independence, object-
tation, mechanisms for knowledge organization, data-driven
utation, inheritance mechanisms, reasoning facilities, etc.
rally, some of these issues which we have been combining to
n a KBMS context, are similar to approaches developed in dif-
t isolated projects. Instead of giving a detailed comparison of
pproaches and those of other projects, we show in this paper
hey can be architecturally combined to build realistic KBMS.
words: AI architectures, Knowledge Base Management Sys-
Database support for Knowledge-Based Systems

1. Introduction

chnology has produced a variety of knowledge-based systems
ranging from simple expert systems to complex natural lan-
e understanding systems. When used for large-scale applica-
KS are faced with problems of managing very large volumes
owledge: virtual memory sizes are not large enough to store
nowledge to be handled, and operations on knowledge (i.e. in-
ce) are computationally intolerable when knowledge bases
are maintained on secondary storage devices.

e problems show that the applicability of KS is limited, since
opriate systems for the efficient knowledge management do
xist. Approaches combining KS with traditional Database
agement Systems (DBMS) for this purpose have failed for sev-
easons (see [1] for a description of the deficiencies of DB sup-
or KS).

olution to KS problem is to develop a new generation of sys-
aimed at the efficient management of large, shared KB. Anal-
to DBMS, these systems are called Knowledge Base Manage-
Systems (KBMS) [2].

observation has motivated our research efforts to identify im-
nt design issues regarding an architecture for KBMS. Our in-
gations have shown that these issues are strongly influenced
ree classes of requirements. Firstly, KBMS should satisfy the
rements of their applications (i.e. KS or end-users). Secondly,
must support the needs of the KB-designer, who plays a very
rtant supporting role in this context. And thirdly, some imple-
ation aspects in terms of data structures and algorithms
d be taken into account, in order to manage the knowledge ef-
tly. Thus, KBMS should provide features obtained from three
ent points of view. So, we believe that KBMS should be archi-
rally divided in three different layers: implementation layer,
eering layer and application layer, which respectively support
of the above classes of requirements.

We feel that the above points are involved with a number of qu
new ideas, which we incorporated in a multi-layered prototypi
KBMS. Clearly, some of these ideas, which we combined to apply
a KBMS context, are related to approaches developed in differ
projects [3,4]. However, these projects handle one or more of th
ideas in an isolated manner, not taking into account the practi
use of them in KBMS. In this paper, we describe our prototypical
chitecture for KBMS, called KRISYS, showing above all how th
and other approaches can be combined in order to build realis
KBMS. To motivate our ideas, we first review the results of our
vestigation: the KBMS architectural issues that arise when
dressing the above mentioned three classes of requirements.

2. Architectural Issues

Traditionally, knowledge representation systems or KS com
nents responsible for the knowledge management have been
signed just to support what we called the engineering layer. Si
the manner of knowledge organization and access is visible at
external interface of those systems, KS possess information ab
the knowledge organization and retrieval possibilities built i
their logic (i.e. embedded in their programs). Any modification
the knowledge structures therefore requires program modifi
tions. Changing the kind of representational framework used,
example from frames to semantic networks, would be even impos
ble, since this would mean throwing the entire KS away and imp
menting a new one.

However, with complex KS and very large KB on the horizon, t
dependence on the framework supported by a knowledge repres
tation system promises to be very problematic. *Knowledge indep
dence* as an analogy to data independence seems to be the key
swer to this problem. Knowledge representation systems must
characterized not in terms of the representational framework th
use, but functionally, in terms of what they know about the KS
main.

This idea of abstraction aimed at the independence of knowle
motivated us to introduce another layer (the application layer) o
the engineering layer in the architecture of KBMS. At the object-
straction interface, KS can, therefore, work independently from
specification of the representational framework supported by
engineering layer. In fact, KS are not interested in things like
complexity of frame structures, the variety of links in a sema
network, the properties of inheritance mechanisms or the powe
reasoning facilities. They are really interested in what they can
or tell the KBMS about the knowledge of their domain, which
stored in the KB managed by the KBMS. This motivated us als
believe that the KS interface is the one supported by the applica
layer (object-abstraction interface). The interface for the KB-
signer is, however, a different one. He (in opposition to the KS
concerned with the aspects of the representational framework.
decides, whether a specific information is to be represented a

2.1 The Application Layer

d at knowledge independence, the application layer should knowledge functionally, in terms of only two basic types of op- ons supported at this level: one to enable the KS to ask the S questions to be answered on the basis of the knowledge kept e KB and another one to permit the KS to tell the KBMS new ledge to be maintained in the KB. Therefore, at the object-ab- tion interface the KB can be compared with an abstract data hat interacts with the end users or KS through a set of "ask" tell" operations. Thus, the way the knowledge is captured or ged is hidden from the KS. Because of this, no distinction can de between knowledge that is extensionally stored and that h has to be implied (intensionally). Whether just simple re- al capabilities or inference of some kind is required to answer stion, is not to be decided by the KS, but in the layer.

functional view of the KB has also been introduced in KRYP- [3], where ask and tell operations are split in two different in- ces: a terminological Tbox, and an assertional Abox. However, sues supported by KRYPTON do not meet the idea of knowl- independence. Before making changes in a KB a KRYPTON has to decide for example, whether the KB`s theory of the should imply these changes (tell operation at the Abox) or the vocabulary should include them (tell operation in the Tbox). her words, he has to decide whether these changes should be sionally represented or extensionally. We argue that such a on should not be made by the user but in the application layer, at the way knowledge is represented can be hidden from the m`s users. We believe that this is the only possible way to view as an abstract data type. Furthermore, KRYPTON`s interface ot been developed for a KBMS context and consequently is nei- flexible nor powerful enough to meet the requirements of S applications. (For example, ask statements are restricted to o questions).

query language provided at the object-abstraction interface naturally permit adequate and flexible ways to select the ap- ion's domain knowledge and to cause changes in the KB cor- nding to the many ways knowledge can be accessed and ged. Furthermore, it must be set-oriented. This permits the KS ain several "pieces-of-knowledge" with just one operation, re- g communications overhead between KS and KBMS and offer- n enormous optimization potential for the lower layers. Addi-

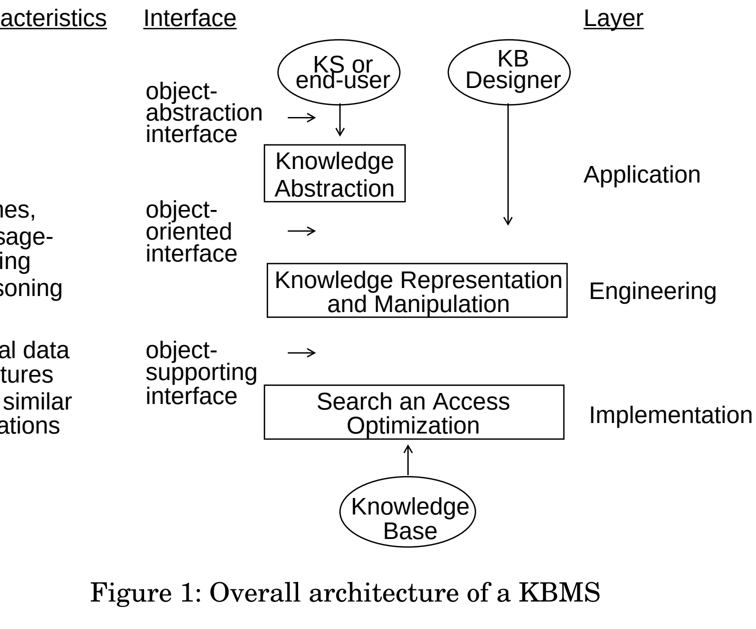


Figure 1: Overall architecture of a KBMS

way knowledge is represented and manipulated).

2.2 The Engineering Layer

The engineering layer focusses on a KBMS from the point of vi of the KB-designer. Here the KBMS is seen in terms of how knowledge of an application can be represented, organized, and m nipulated. In other words, this layer deals with descriptive, orga zational and operational aspects of a knowledge representat model supported for the KB-designer at the engineering layer's terface (object-oriented interface). Concerning the descriptive pr ciples:

- Entities in the application's world should correspond to exac one object in the model.
- Each object should be composed of attributes expressing properties of the entities.
- Attributes are multiple-valued and should be further describ (e.g. data type of the attribute values, etc.).
- Objects are described by object types (i.e. classes), which sho also be objects of the model.
- Classes characterize prototypical instances, i.e., properties scribed in the classes might be contradicted by particular stances.
- Objects can be instances of several classes.
- Relationships between entities should be expressed as propert of objects [5].

Concerning the organizational principles, the abstraction conce [6,7,8], extensively used in AI knowledge representation syster are the best candidates:

- Objects are to be grouped into classes (classification), which, turn, should be organized into class hierarchies providing means for the overall organization of the KB (generalization)
- Inheritance must be supported, in order to eliminate unnec sary definitions. Multiple inheritance should also be allow and the notion of inheritance should be that of default [9,6].
- Similar element objects should be organized into set-element erarchies (association).
- Aggregation of component objects should be supported, a might be applied recursively, so that another organizational mension of the KB can be defined.

Operational issues should primarily support object orientati This will allow the attributes of the objects to store either extensi al or intensional data, which are accessed in a uniform way (i.e. sending object messages). Moreover, object orientation makes possible to extend the functionality of the layer interface very ea ly, since it permits the KB-designer to define his own functio which as attributes of objects can also be directly addressed at interface by message passing. Reasoning facilities should also provided. The knowledge representation model should offer alrea implemented general purpose reasoning functions, not prohibiti however, that these functions may be substituted by special or developed by the designer. Mechanisms for checking integrity c straints are to be provided too. To define constraints this la should enable the KB-designer to define procedures that are link to attributes of objects and are automatically activated before or ter such attributes are accessed or updated. These procedures tached to data will then perform the necessary checks, and the c responding actions to keep the KB in a consistent state.

ts supply to the engineering and application layers. For this n, many of the issues here are related to traditional DB prob-

requisite to identify the specific requirements to be supported is the investigation of the behavior of KS running on second-torage environments. In order to pursue this investigation, we adapted some available KS to work on such environments [10]. bservations may be summarized as follows:

- e accesses made at the implementation layer interface (object-pporting interface) are mainly to tiny granules refering to in-vidual attributes of objects rather then to objects as a whole.
- e attributes' access frequencies differ very much. Dynamic at-butes (i.e., the specific knowledge of a consultation) are access-with very high frequencies; static attributes (i.e., the expert owledge of the KS) on the other hand with very low frequen-s.
- e dynamic knowledge can be kept temporarily, since it only ex-esses information of a particular consultation.
- multi-user environments, the KBMS must maintain as many rsions as the number of users working with the KB for the dy-mic knowledge. These versions are then accessed individually, that synchronization should only be controlled for the static owledge.
- each phase of the problem solving process, accesses concen-te on just some objects of the KB. These objects build a context ich contains the knowledge needed to infer the specific goal of at phase. The context needed in each phase can be established some information that the KS deduces during the preceding ases (i.e., dynamic knowledge), so that it is possible to deter-ne the next context needed at the end of a phase. This enables ynamic preplanning of the accesses to the KB that should be ed at this layer for optimization purposes.
- orage structures for the KB should give more priority to the op-nization of the retrieval operations, since modifications in the uctures of the KB (e.g. changing object types) are very seldom.

3. The KRISYS-Prototype

3.1 Overview of the System Architecture

YS is architecturally divided in three different modules, which t the aspects of the previously discussed layers (Figure 1). The cation layer corresponds to the Query-Processing Manager re 2). Here, knowledge independence is supported by the YS Object-Abstraction Language (KOALA), which character-he object-abstraction interface. To implement the engineering we choose a Frame-System (KRISYS Frame System KFS) n, in turn, supports the interface to the KB-Designer and to uery-Processing Manager, i.e., the object-oriented interface. mplementation layer requirements are fulfilled by the Work-lemory and Context Manager (WMCM) and by a DBMS Ker-VMCM embodies the nearby application locality concept, i.e., reservation of the locality of the application's object processing

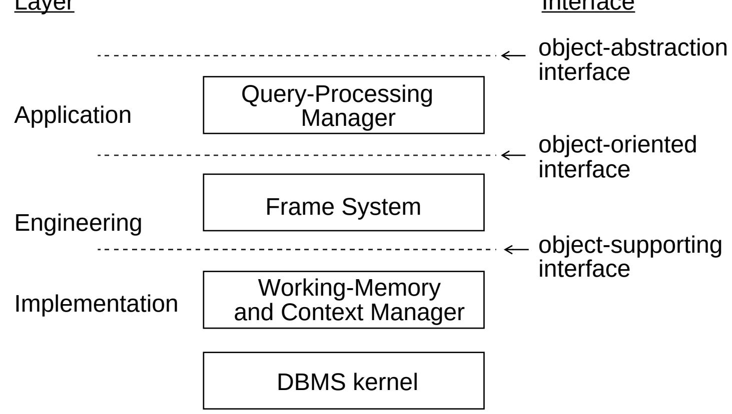


Figure 2: The KRISYS Architecture

3.2 Query-Processing Manager

The application layer of KRISYS can be better characterized by description of its object-abstraction interface, the language KOA which defines all interactions between a user or application and system, which occur with two types of operations:

ASK -	to ask KRISYS,	whether an "expression" is true, retrieving the qualified knowledge, a
TELL -	to tell KRISYS,	that an "expression" is true, changing the KB appropriately

The Ask-Statement

An Ask-statement contains two parts; a projection and a select (i.e. qualification) part.

ASK[<projection>][<selection>]

The last one specifies the expression to be proved, which is str tured in accordance to the first-order predicate calculus. The sel tion is evaluated following the model-theoretic approach [11]. other words, ask-statements are evaluated as true or false with spect to the KB. During evaluation, the Closed World Assumpt (CWA) [12], the Unique Name Assumption (UNA), and the Dom Closure Assumption (DCA) are taken as granted. For example, query (ASK (IS_INSTANCE tweety penguin)) would be true onl there exists both tweety and penguin as objects of the KB related the abstraction concept of classification.

The logical formulas are built with predicates (e.g. IS_INSTAN in the above example). These use either constants, variables functions as their terms. The last one can, in turn, use variables constants as terms again, in order to address the objects of the KOALA supports functions and predicates to be applied to obje (schemas), attributes (slots), attribute values (slot values) and tribute descriptions (aspects). Some important object predica are those used to express the abstraction concepts of classificati instantiation (IS_ INSTANCE) generalization/specialization (SUBCLASS) and association (IS_ELEMENT, IS_SUBSET). next example shows a query which asks whether or not elephan a specialization of mammal and an element of the animals_from_africa:

```
(ASK (AND (IS_SUBCLASS elephant mammal)
          (IS_ELEMENT elephant animals_from_africa)))
```

To manipulate attributes, KOALA offers functions to select all sl of an object, only the inherited ones, the inherited ones from a p ticular object, the slots defined for an object (i.e. not inherited), instance_slots, its own_slots, its standard_slots or its us defined_slots. These functions have as their result a set of slo which are to be interpreted as the components of the specified obj (abstraction concept of aggregation). If someone would like to a

((IS_IN nutriment (INHERITED_SLOTS camel
(SUPERS mammal))))

tions about slot values can be expressed by a lot of predicates, n use functions to access the slot value itself (e.g. SLOTVALUE _name> <schema_name>). The equality predicate can be used for example, to express that a slot value is equal to a specific :

(EQUAL milk (SLOTVALUE nutriment camel)))

erical comparison predicates are also provided. Furthermore, other functions, it is possible to evaluate arithmetic expres- such as, sum, average, minimum and maximum value of a set mbers or even to count the number of elements of a set.

bute description predicates allow the application to formulate tions about slot value restrictions, cardinality specifications, or example, the query

((IS_POSSIBLE_VALUE fly (ASPECTVAL
POSSIBLE_VALUE
form_of_locomotion mammal)))

s, whether the value fly is allowed in the slot _of_locomotion of the object mammal.

rientation is achieved by using variables in the logical formu- Variables express either objects, attributes, attribute values or bute descriptions of the KB. During evaluation they are instan- d with all values, which satisfy the logical formulas. Because of CA, formulas, which contain variables that cannot be instan- d, are assigned 'false'. To ask whether there exists any bird cannot fly, one can use variables to create the following ques-

((EXIST ?X (AND (IS_INSTANCE ?X bird)
(NOT(IS_IN fly (SLOTVALUE
form_of_locomotion ?X))))))

bles can also be used to construct questions involving more one object (roughly analogous to a database join). The user can build queries combining variables that represent different s (i.e. objects, attributes, attribute values or attribute descrip-). For example, to know whether there is any object in the KB has stored a specific value in any of its attributes, someone d pose the question

(EXIST ?X ?Y (AND (IS_SCHEMA ?X)
(IS_IN ?Y (ALL_SLOTS ?X))
(EQUAL specific_value
(SLOTVALUE ?Y ?X)))))

uently, the user is also interested to see the instances, which y his logical formulas (e.g. to identify the dogs descending from ame parents), in addition to the boolean value of these formu- He can specify this in the projection part of the ask-statement. xample, if someone would like to know which mammals can , he should ask

(?X) (AND (IS_INSTANCE ?X mammal)
(IS_IN swim (SLOTVALUE
form_of_locomotion ?X))))

rojection also enables the user to express exactly which parts accessed information (objects, attributes etc.) he would like to This can be achieved by using specific projection clauses or by ining different variables (e.g. variables that represent objects those that represent attributes). For example, if someone s to see the dogs stored in a KB together with the value of the they have in common with cats, one would ask KRISYS the fol- g query:

Recursive queries may be expressed using the PATH predica with which a special class of recursion equations, called generali transitive closure (GTC) [13], can be specified. Unfortunately, to space limitations it is not possible to discuss these aspects h (For a detailed description see [1]).

The Tell-Statement

The tell-statement takes a sentence and asserts that it is true. effect is to change the KB into one whose contents imply that s tence. Naturally, it can also occur that the contents of the KB already imply the sentence asserted. In this case, no changes wil made, since they are not required. For example, if the KS asse that penguin is a subclass of bird, expressing "(TE (IS_DIRECT_SUBCLASS penguin bird))" two situations can occ Either the KB already contains this information, requiring changes, or the KB does not contain it and consequently chan must be made. Here, many things can happen. If neither bird penguin exist as objects in the KB, both will be created and rela to each other by the abstraction concept of generalization. If o one of them exists, the other one will then be created and relate the first one as specified above. And if both of them exist only generalization relationship will be built.

TELL<assertion>[WITH<selection>]

The sentences to be asserted are specified in the assertion par the tell-statement. An assertion is syntactically similar to the se tion part (see ask-statement), however, much more restrictive. I for example, not possible to specify formulas combined with log connectors, in order to avoid ambiguities (note, that if someone serts p v q, it is impossible to know whether p, q or both are tr Nevertheless, several assertions can be specified, which will the interpreted independently, i.e., KRISYS will make each one tru

(TELL (IS_DIRECT_SUBCLASS penguin bird)
(IS_DIRECT_INSTANCE tweety penguin)
(EQUAL frankfurt_zoo (SLOTVALUE address tweety))

If the same assertions should be made valid for several objects, a can specify this requirement in just one tell-statement by us variables to represent these objects. The variables would then be stantiated during the evaluation of the selection part, which has same syntax and works just like the selection of the ask-stateme After this evaluation, the assertions will then be applied for each stance of the variables that satisfy the selection. The following ample shows a query expressing the assertion that every anim that lives in water (here represented as elements of the set wat animal) can swim:

(TELL (IS_IN swim (SLOTVALUE locomotion ?X))
WHERE (IS_ELEMENT ?X water_animal)).

Another example for the use of variables shows the following qu which expresses that every owner of a dog of the KB should also an object of the KB:

(TELL (IS_SCHEMA ?X)
WHERE (FOR ALL ?Y WITH (IS_INSTANCE ?Y dog)
(IS_IN ?X (SLOTVALUE owner ?Y)))).

Summarizing this chapter, we can say that the object-abstract interface is completely specified by the above two operations. "T takes a sentence and asserts that it is true, changing the KB i one whose contents imply the assertion. "Ask" takes a sentence a checks on the basis of the current contents of the KB whether i true or not. Schematically, they can be described as

TELL : KB x sentence ==> KB Sentence is true
ASK : KB x sentence ==> T/F Is sentence true

idden from the user. The KB is characterized as an abstract type, specified only by these two operations rather than by a n implementation structure.

3.3 The Frame-System

hoose a frame system [4] to implement our engineering layer, se we believe that it offers the necessary framework to repre- he descriptive, organizational and operational aspects of the in knowledge of any KS.

t-Oriented Representation

e KRISYS Frame-System (KFS) the three aspects mentioned e are incorporated in its basic concept: the schema. A schema symbolic representation of a real world entity. It is composed chema name and of a set of attributes. Attributes can be of two :

ts, used to describe the descriptive and organizational aspects he schema, and

ethods, used to describe its operational aspects.

efore, KFS could be named object oriented. It allows both at- tive characteristics and procedural properties of the real world es to be integrated into a schema.

ns in KFS occur by sending messages to the objects, which can, rn, communicate with other objects by message passing. This pt supports the important principle of data abstraction. That order to request the methods of some objects to be performed, sumptions have to be made about the implementation and in- l representation of the objects (i.e. slots and methods).

and methods have the same structure. Both possess name, , type and a schema-name, that specifies where the attribute efined (origin). The type indicates whether an attribute repre- characteristics of the schema itself, or of its instances. Char- istics of a schema are described by the types OWNSLOT and METHOD, whereas the instance ones by INSTANCESLOT NSTANCEMETHOD, respectively.

ciated with the attributes there can be aspects, which are used scribe the attribute more exactly. Five aspects are predefined S for slots: possible-values, cardinality, comment, default and ns. Figure 3 shows the structure of the schema elephant. Since tributes shown in the example describe characteristics of ele- t itself, they have type OWNSLOT or OWNMETHOD (the ing of the slots INSTANCE-OF and ELEMENT-OF will be de- ed later).

```
phant
NSTANCE_OF          mammal              OWNSLOT  GLOBAL
LEMENT_OF           savanna-animal      OWNSLOT  GLOBAL

rm-of-locomotion    (walk swim)         OWNSLOT  animal
   POSSIBLE_VALUES     (walk fly swim)
   CARDINALITY         [1,3]
   DEFAULT             (walk)

abitat              (savanna)           OWNSLOT  animal
   POSSIBLE_VALUES  INSTANCE_OF habitat-class)
   CARDINALITY      [O, ∞]
   COMMENT          (This slot contains the places, where this
                      animal  is found in the nature)
at                  (<code>)            OWNMETHOD  animal
   PARAMETER        (food)
```

Figure 3: Example Definition of the Schema Elephant

slots, that are automatically activated when these slots are acc ed. This concept of data-oriented computation is very useful to r resent intensional data or to check complex integrity constrai For example, if someone wants to represent some informat whose value changes with respect to other data (e.g. the exact of a person, which changes every day), he needs a mechanism t generates the extensional value of this information automatica each time this information is accessed. This can be realized in K by using demons.

Demons are stored in a KFS schema (like any other), which h however, the predefined schema DEMON as its most superclass. an instance of DEMON this schema inherits the method attribu GET, PUT, ADD and RETRACT, where the code for the respect demons are stored (represented by schema D1 in Figure 4).

The linkage between slot and corresponding demons is done by st ing in the demons aspect of the slot (S1) the schema name of the spective demons (D1). Whenever an access to the slot is made K checks if there is a demons aspect defined in this slot, activatin by sending a message to the schema specified there, demanding evaluation of the corresponding demon. That is, the method spe fied in GET will be invoked if a get-access has been issued to K the method PUT is invoked by issuing a put-access etc. By not st ing the demon-code directly in the aspect of the slot, as is done many systems, KFS allows many slots to use the same dem without having to introduce redundancy in the representation. K additionally allows the KB-designer to specify when the den should be activated (i.e. before or after the access to the slot). T flexibility allows the use of demons for many different purposes. mons applied to check integrity constraints should be activated fore updating the value of a slot, whereas those used to trigger tions when particular slots are accessed should be activated a the access itself.

```
schema-x
   :
   S1 <S1-value> OWNSLOT <S1-origin>
       POSSIBLE-VALUES (...)
       :
       DEMONS D1
   :

D1
   INSTANCE_OF   (DEMON)
   GET          <get-code>        OWNMETHOD  DEMON
   PUT          <put-code>        OWNMETHOD  DEMON
   ADD          <add-code>        OWNMETHOD  DEMON
   RETRACT      <retract-code>    OWNMETHOD  DEMON
```

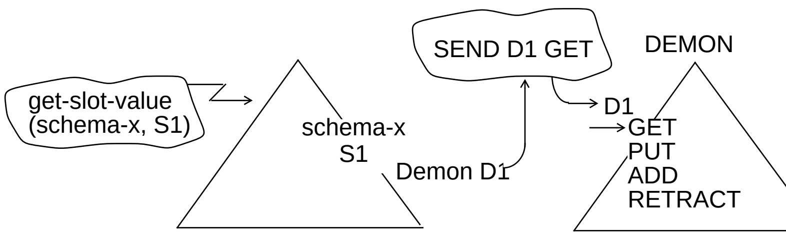


Figure 4: Activation of Demons

Knowledge Organization

In order to enable the KB-designer to structure the KB adequat KFS supports various abstraction concepts. These are represen as relationships between objects specified by predefined sl which are contained within every schema. The abstraction conc of generalization is represented by the slots SUBCLASS_OF a HAS_SUBCLASSES. INSTANCE_OF and HAS_INSTANCES used to specify the classification relationship. The association c any special slot for its representation.

lating objects to another, hierarchies are built, where a partic- schema can play the role of a class (i.e. object type), an in- e, a set and/or an element. Since a schema is able to reflect an nce and a class as well, we need two different slot and method to know whether a slot represents characteristics of itself NSLOT and OWNMETHOD) or of its instances (INSTANC- OT and INSTANCEMETHOD).

itance

of the advantages of organizing the schemas into abstraction rchies are the built-in reasoning facilities they provide [6]. By ng objects with abstraction relationships, special "automatic" ning facilities are supported as part of each modification and val operation. The most usual and important of these reason- cilities is the one built-in the is-a hierarchy i.e. inheritance. In only instanceslots and instancemethods are inherited, since wnslots and ownmethods represent characteristics of their has. Between schemas linked with association relationships no itance occurs. Here, as well as over aggregation relationships, kinds of built-in reasoning facilities are provided. For a de- l description of these facilities see [6].

oning

also supports general reasoning facilities. Rules in KFS are sented as schemas, which are also organized in special hierar- having two predefined schemas as superclasses. One of these has specifies the structure of the rules, i.e. this schema ES) contains some instanceslots (condition, action, etc.) which herited by each rule. In order to construct a rule base, the KB- ner has only to store the particular contents of the condition, n, etc. in the respective slot of each rule. The other top level ma (RULE-SETS) contains methods corresponding to the stan- inference strategies, which are then inherited by each set of

rticular rule of KFS can, therefore, be used in both reasoning tions (i.e. forward-reasoning and backward-reasoning) without g to be stored redundantly. The reasoning direction of the are dynamically defined when the KS requests the activation e of the inference strategies. Since these are implemented as ods, the activation of reasoning mechanisms is made by send- message to a rule set. This will then activate its corresponding od (forward-reasoning or backward-reasoning) evaluating of its rule elements.

narizing this chapter, we can argue that KFS supports all is- needed at the engineering layer. Descriptive aspects are satis- y the rich declarative part of our frame model, i.e. the struc- of schemas, slots, slot values and aspects. Organizational as- are supported by the abstraction hierarchies and inheritance. ational requirements are met by allowing the KB-designer to fy his own methods (object orientation), by supporting at- d procedures (data-oriented computation) and by offering rea- g mechanisms.

3.4 Working-Memory and Context Manager

ready mentioned, the goal of this component of KRISYS is to de a framework for the exploitation of the application's locali- is, therefore, desirable to have a mechanism that enables re- on of DBMS calls and the reduction of the path length when ac- ng the objects of the KB, allowing the application to reference ts almost directly. This is reached by storing needed objects orarily in a special main memory structure called working-

to the stored objects.

The approach used to support locality, is to dynamically extract context (see 2.3) needed in each phase of the application's probl solving process from the DBS. Therefore, during changes of proce ing phases the old context would be discarded from the worki memory and the new one loaded into it. Since the application's cesses will then be concentrated on the objects of the loaded conte only a few DBS calls will be made during the next processing pha Due to the set-oriented specification of the required objects, DBS can use its optimization potential, reducing I/O and trans overheads. Furthermore, the tiny granules of the application's cesses will not bring any inefficiency, since the path length when cessing the objects is now very short. (Details about the inter structure and implementation aspects of the working-memory provided by [1]).

	duration of consultation (CPU-sec)	ratio to main memory approac
main mamory approach	146	1
direct coupling	28105	~ 170
coupling with application buffer	2946	~ 18
WMCM	643	~ 4

Figure 5: Performance Comparison of Different Coupling Approaches between KFS andDBS

The performance of WMCM has been compared with the per mance of other coupling approaches between KFS and DBS (Fig 5). In this case we use a main memory-based form (i.e. when whole KB is stored in main memory) as basis for this comparisi Coupling KFS and DBS directly seems to be the most inefficient ternative, due to the long path length (from application to DB-b er) by the application's very tiny access granules. This problen eliminated by using an application buffer, which uses LRU as a cation/deallocation strategy, to keep the most recently used obje This approach needs, however, to perform very many DBS ca since individual objects are frequently being extracted from DBS and stored into it. This is particularly critical after change processing phases, when most of the needed objects are not founc the buffer. We solved this problem by a set-oriented fetch, as scribed above, which accomplishes an efficiency very close to main memory approach. Indicative performance figures of a par ular example as illustrated in Figure 5 are derived in [10].

3.5 DBMS Kernel

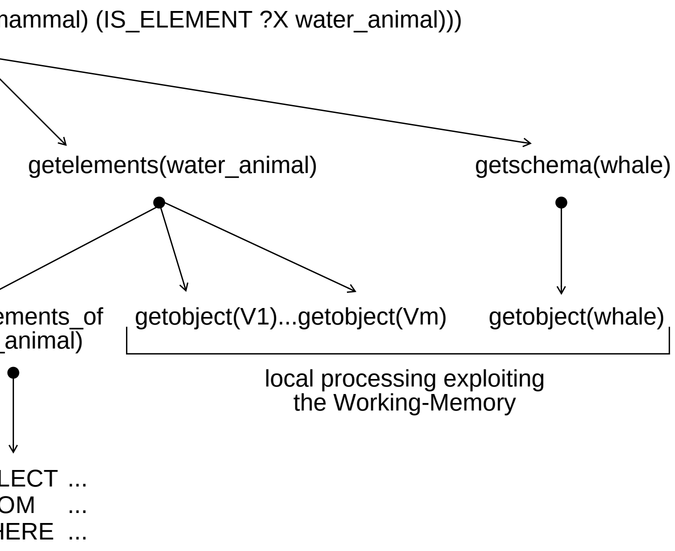
Due to space limitations it is not possible to discuss the aspect the DBMS kernel and the mapping of frames with it in suffici depth.

The DBMS kernel architecture chosen for KRISYS [14, 15,16] v developed for the support of the so called non-standard appli tions. For this reason, our kernel, named PRIMA, offers neut yet powerful mechanisms for managing KB of all types of KS e ciently: storage techniques for a variety of object sizes, flexible r resentation and access techniques, basic integrity features, lock and recovery mechanisms, etc. [17]. PRIMA is a *PR*ototype *I*mp mentation of the *MA*D model. MAD provides dynamic definit and handling of objects, based on direct and symmetric mana process are found in [18,19, 20].

3.6 Summarizing Example

s chapter, we illustrate the control flow through the KRISYS s by giving a simplified example of the operations needed to a KS query at each of the system interfaces (we consider the ing-Memory as been empty at the beginning of an evalua- Since a complete description is too space-consuming, we re- ourselves to essential properties expressing the flavor of the nal operations.

ree structure of Figure 6 indicates the calling hierarchy need- r the evaluation of an ASK-statement (where control opera- etc. are dropped for simplicity reasons). Exploitation of infor- on encountered in the Working-Memory would save (expen- calls to the DBMS kernel interface (with accesses to the S buffer or even to the disks). Perfect locality preservation d avoid any kernel calls for repeated object references.



on of an ASK-statement

presented the design of our prototype, named KRISYS. We h advocated the division of KBMS in three different layers, which spectively support the requirements of the application, the ne of the KB-designer and the requirements of an efficient mana ment of the KB. The main philosophy of the system is the ide abstraction aimed at the independence of knowledge. At the s tem interface, knowledge is seen not in terms of the flexible rep sentational framework supported by the engineering layer functionally, in terms of the application's beliefs about its doma In order to achieve this functional view of the KB, it is necess to restrict the communication between application and KBMS ASK and TELL operations, however, providing:

- multiple ways to select the application`s domain knowledge,
- user defined projections,
- flexible forms to cause changes in the KB,
- set-orientation, and
- recursion.

A second very important philosophy is the effective support of needs of the KB-designer. Clearly, this philosophy has also b the goal of most existing knowledge representation systems wh try to support these needs by focusing on the improvement of expressiveness (i.e. descriptive aspects) of their representat model. However, in a KBMS context expressiveness is just on the three aspects to be considered at the engineering layer. H organizational and operational aspects are equally as importan descriptive ones. Because of this, an effective support of the ne of the KB-designer is to be achieved by a mixed knowledge rep sentation framework offering:

- object-orientation,
- data-driven computation,
- mechanisms for knowledge organization (i.e. all abstract concepts),
- inheritance and other built-in facilities, and
- reasoning mechanisms,

all of them uniformly integrated with a very rich descriptive pa

A number of concepts used in the KRISYS implementation pay tention to performance requirements. Most important is framework provided by the implementation layer for the explo tion of the application locality. A single-user version of KRISY now complete. KRISYS is to be considered a research vehicle fo variety of KS. It is intended, therefore, to run as a tool for build KS and as a generic system, aimed at the effective and effici management of large knowledge bases.

Acknowledgement

I should like to thank T. Härder and B. Mitschang for the help comments on the revision of this paper, and H. Neu and I. Litt for preparing the manuscript.

References

[1] Mattos, N.M.: KRISYS - A Multi-layered Protot KBMS Supporting Knowledge Independence, Resea Report ZRI No. 2/88, University of Kaiserslautern, 19

[2] Mylopoulos, J.: On Knowledge Base Management S tems, in: [21], pp. 3-8.

[3] Brachman, R.J., Fikes, R.E., Levesque, H.J.: Krypton Functional Approach to Knowledge Representation, Computer, Vol. 16, No. 10, 1983, pp. 67-73.

[4] Fikes, R., Kehler, T.: The Role of Frame-based Repres tation in Rsoning, in: Commications of ACM, Vol. 28, 9, September 1985, pp. 904-920.

[5] Borgida, A.: Survey of Conceptual Modeling of Infor tion Systems, in: [21], pp. 461-470.

Schrefl, M., Tjoa, A.M., Wagner, R.R.: Comparison-Criteria for Semantic Data Models, in: Proc. IEEE 1st Int. Conf. on Data Engineering, Los Angeles, 1984, pp. 120-125.

Brodie, M.L., Mylopoulos, J., Schmidt, J.W. (eds.): On Conceptual Modelling (Perspectives from Artificial Intelligence, Databases, and Programming Languages), Topics in Information Systems, Springer, New York, 1984.

Borgida, A., Mylopoulos, J., Wong, H.K.T.: Generalization/Specialization as a Basis for Software Specification, in: [8], pp. 87-114.

Härder, T., Mattos, N., Puppe, P.: On Coupling Database and Expert Systems (in German), in: State of the Art, Vol. 1, No. 3, 1987, pp. 23-34.

Gallaire, H., Minker, J., Nicolas, J.-M.: Logic and Databases: A Deductive Approach, in: ACM Comp. Surveys, Vol. 16, No. 2, June 1984, pp. 153-185.

Reiter, R.: On Closed World Databases, in: Logic and Databases, (eds. Gallaire, H. and Minker, J.), Plenum, New York, pp. 56-76.

Dayal, U., Smith, J.M.: PROBE: A Knowledge-Oriented Database Management System, in: [21], pp. 227-257.

Härder, T., Reuter, A.: Architecture of Database Systems for Non-Standard Applications (in German), in: Proc. of the GI-Conf. on Database Systems for Office, Engineering and Science Environments, 1985, pp. 253-286.

Dadam, P., et al.: A DBMS Prototype to Support Extended NF2-Relations: An Integrated View on Flat Tables and Hierarchies, in: Proc. ACM SIGMOD Conf., Washington, D.C., 1986, pp. 356-367.

Paul, H.-B., Schek, H.-J., Scholl, M.H., Weikum, G., Deppisch, U.: Architecture and Implementation of the Darmstadt Database Kernel System, in: ACM SIGMOD Conf., San Francisco, 1987, pp. 196-207.

Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - A DBMS Prototype Supporting Engineering Applications, in: Proc. 13th VLDB Conf., Brighton, 1987, pp. 433-442.

Mitschang, B.: A Molecule-Atom Data Model for Non-Standard Applications - requirements, data-model design, and implementation concepts (in german), PhD Thesis, University of Kaiserslautern, 1988.

Mattos, N.M.: Mapping Frames with the MAD model (in German), Research Report No. 164/86, Univ. Kaiserslautern, 1986.

Härder, T., Mattos, N., Mitschang, B.: Mapping Frames with New Data Models (in German), in: Proc. GWAI'87, Springer, 1987, pp. 396-405.

Brodie, M.L., Mylopoulos, J. (eds.): On Knowledge Base Management Systems, Springer, New York, 1986.