KYBERNETIKA — VOLUME 40 (2004), NUMBER 3, PAGES 365 - 379

THE DIAMOND TOOL: A WAY OF EFFECTIVE DEVELOPMENT AND UTILIZATION OF KNOWLEDGE

ZDENKO STANÍČEK AND FILIP PROCHÁZKA

This paper presents the Diamond Tool for knowledge management. The main objective of its specification and implementation was to create a universal and easily extendable tool for efficient work with knowledge. One of its extensions is the eTrium technology.

The principal idea behind this technology is to represent explicitly the knowledge used by the information system by means of a knowledge agent built on the Diamond Tool – in contrary to current approaches, where knowledge is present implicitly (in a disintegrated form) in the program code, the database structure and internal regulations of the firm. Explicit representation of knowledge and the method of work with knowledge that is suggested in this article, allows to build information systems of much higher quality in lesser time. Equally important is also the possibility of automatic or semi-automatic verification of knowledge managed by the knowledge agent.

The paper shows how this can change the way applications are created and what it can be used for. Several commercial projects has already been implemented in the Diamond Tool – by means of the eTrium technology.

Keywords: knowledge management, information system architecture, knowledge representation, business rules, business rules engines

AMS Subject Classification: 68N30, 68P20, 68Q55, 68T30, 68U35

1. INTRODUCTION

Despite all the efforts in creating new methods, techniques and technologies of IS development, the problem of inadequate and untimely IT support of the changing business requirements remains topical. In search of a solution to this problem, we aim at improving the quality of the description of required system functions, and, possibly, of processes that should be supported by the system; and also at the description of the required information capability.

A very important fact is that it is not feasible to enumerate all the possible functions of such a system. Today we do not know the form and requirements of business actions that we will urgently need in half a year [13]. Therefore, the entire system has to be flexible enough to *allow support of business activities with new ways of information processing and new types of information records.*

At present, a substantial growth of the importance of rule based systems [1, 2] can be noticed. These systems can be best characterized by the following quotations

from the Business Rules Manifesto [3]:

- Rules, and the ability to change them effectively, are key to improving business adaptability.
- A business rule system is never really finished because it is intentionally built for continuous change.
- Rules define the boundary between acceptable and unacceptable business activity.
- Rules are explicit constraints on behavior and/or provide support to behavior.
- Rules are about business practice and guidance; therefore, rules are motivated by business goals and objectives and are shaped by various influences.
- Rules are basic to what the business knows about itself – that is, to basic business knowledge. Rules need to be nurtured, protected, and managed.
- Rules are essential for, and a discrete part of, business models, system models, and implementation models.
- Terms (of natural language) express business concepts; facts (in a form of connections) make assertions about these concepts; rules constrain and support these facts.

The Gartner Group [5] predicts a dramatic growth of commercial importance of rule based systems and various business rule engines (BREs) for the years after 2003. All these systems use, in fact, work with knowledge that is explicitly stored in the form of rules.

The web pages of Pegasystems, Inc., [7] provide quite a complex range of commercial solutions built on the so-called PegaRULES Process Commander, which is their patented BRE. Solutions include building of enterprise process flows, managing complex works, web services, security model, etc.

In the article "Business RULES-Show Power, Promise" [4], Ellen Gottesdiener describes business rules – what they are, why they are important, how they are used. She also deals with their relations to expert systems and information systems, and with their impact on ICT paradigms. Moreover, she enumerates various examples of applications of the business rule approach, e. g., among others: ObjectPool from Sapiens, Tel Aviv, or Performer, a model-driven application development tool from Texas Instruments, Inc. Both of them are application generators that integrate business rules. Another examples are the Vision Builder from Vision Software, Oakland, or the Usoft Developer from Usoft Corp., Brisbane, Calif. These are repository-driven application generators. The latter use the notion of declarative business rules and a business rules engine to drive the development process and permit an automatic generation of application components based on business rules. Usoft Developer, for example, uses business rules as the central paradigm for all aspects of the application development lifecycle.

Another perspective of the state-of-art in business rules driven applications is presented in the article [6]. This article explains the role of business rules in information systems, classifies individual types of business rules, and especially, compares various traditional methodologies of IS design from the point of view of their suitability

for easy business rules specification. The article shows that common methods are insufficient or at least inconvenient for complete and systematic modeling of business rules.

Our solution, that is presented in this paper, has been developed separately from the above cited solutions. Only at the stage of test operation of pilot applications built over the Diamond, we learned about the above quoted results. From what we know about the BRE solutions ([1, 2, 3, 4, 6, 7]), it can be concluded that the general basis of BRE in the form of the Diamond slightly differs from these solutions and, in certain aspects, it seems to provide rather promising solutions. Its advantage is its elaborated formal model (cf. further in this text) and especially the fact that due to a universalistic approach it provides a sufficient and convenient tool for complete and systematic modeling of business rules. The main differences between this approach and the above mentioned solutions could be described by the following statements:

(1) We do not create an application generator where modeling is separated from using the model. Instead, we create one system that can be used for both - the modeling of business and using the models to support the business.

(2) We have no special tools to maintain and handle business rules; business rules are handled as any other object of our interest.

Our approach could be effective if the need for efficient changes is crucial. Traditional BRE solutions could be more efficient if the performance of IS is crucial.

2. THE DIAMOND TOOL DEFINITION

At the beginning of the Diamond Modeling Tool there was the integration of ideas belonging to two, seemingly remote, areas: the area of conceptual modeling and attempts to find a universal modeling tool for the work of analysts and data modelers, [9, 10, 14], and the area of modeling of cognitive processes and attempts to find a universal tool for modeling of cognitive agents [8]. The outcome of this integration is the specification and implementation of the Diamond Modeling Tool. This tool can be used for modeling and simulation of any conceptual system, i. e., for example, for modeling of a conceptual system that an analyst aims to discover in the analysis of a given business area, or for modeling of the conceptual system of an agent simulating the behavior of a particular biological system.

If we want to get to know something, we have to focus first on individual objects (#Object) of the world we are getting to know. This world may be represented by anything: the material world around us, documents containing records about this world, but also the world of ideas, or the world of ants. To focus on an individual object means to differentiate it from everything else, to identify which phenomenal instances represent the same object, and finally, to be able to describe the properties of this object.

Objects alone are, however, usually less interesting for us than their particular connections (#Connection). These connections are, in a certain sense, stable, and may play the role of factual knowledge of these objects. Connections can be determined by a mere grouping of objects that shows visible stability (n-tuples of objects),

or by a function that assigns objects of one type to objects of the same or a different type with a certain meaning, i. e. semantics.

If we want to work with objects and their connections in any way (record them, delete them, assign some properties to them, search for objects of given properties that participate in given connections, etc.), we have to introduce operations (#Operation). They are introduced, particularly, in order to be used. If we could only speak about operations, we could not use them for work. However, to speak about operations is also meaningful, for example, in order to clarify which operations are crucial for us.

But let us return to connections. We frequently speak about connections – we make different assertions about them. Further, we use connections. For example, the answer to the question "Which particular documents have been approved?", is acquired by using the connection "documents that are placed into the category Approved documents".

Knowledge does not concern only individual objects, individual connections or individual operations. Knowledge concerns also our classification of individual items into categories (#Category), and our stating rules (#Rule) about it. Knowledge is that an object belongs to a category, or that a certain rule is valid for a certain category of objects.

When we speak about a category, we speak about it as about any arbitrary object. But when we want to use a category, we have to know by means of which *connection* it was created (defined) and by means of which *operation* we can evaluate what belongs to the category and what does not belong there.

Rules, similarly to categories, are determined by a specifying *connection* and an evaluating *operation*. The specifying connection describes, in the language of objects, what is the test carried out for the given model by a respective evaluating operation about. Further, when we speak about a rule, we do so as in case of categories: as about any arbitrary object. However, if we want to use a rule, then we use specifying connections and evaluating operations.

Anything that we want to mention is an object, i. e. an element of the class (#Object). Class (#Object) forms the centre of the "Diamond Diagram", cf. Figure 1.

Any ideational construct that can be used for arrangement of some elements, where an element can be anything on which we can focus, will be called a *container*. Elements can be inserted into the container as well as removed from it. Containers used in the Diamond will be defined by specification of elements that belong to them. Every element stored in DMT that represents an object of the examined reality (an entity) on which we are able to focus will be called a *dmt-object*. We will say that a dmt-object *represents* an object of the examined reality for the purposes of modeling.

Definition 1. Container (#Object) is defined in such a way that it contains all such dmt-objects which can be mentioned in DMT, i. e. which can be assigned certain properties.

If the object on which we focus turns to be a connection or an operation or a

category or a rule, and we will want to use this connection or operation or category or rule, then we connect it to (#Connection) or (#Operation) or (#Category) or (#Rule), respectively, by an edge R1 or R3 or R2 or R4. Then we say that this object from the class (#Object) represents this connection or operation or category or rule – cf. Figure 1. Note: an object may represent only one of the four mentioned possibilities.

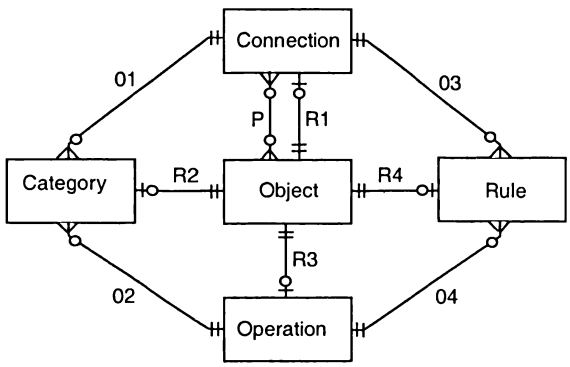

Fig. 1. The Diamond Tool diagram.

Definition 2. Container (#Connection) is defined in such a way that every of its elements is an n-tuple of dmt-objects, where n is some finite natural number. Every element of the container (#Connection) is called a connection.

Definition 3. Container (#Operation) is defined in such a way that every of its elements is an algorithmically computable transformation of one state of DMT to, generally, another state of DMT. By the state of DMT we mean one particular filling of DMT (as a container) by elements-instances, that may dwell in DMT. The elements of the container (#Operation) are called operations.

Definition 4. Container (#Category) is defined in such a way that every of its elements has the following properties:

1. it is a container for dmt-objects,
2. it is one-to-one mapped to the pair $\langle$ Cn, Op $\rangle$, where Cn $\in$ (#Connection), Op $\in$ (#Operation); and,
3. it holds about the operation Op that by means of the connection Cn it can recognize whether a given object is or is not in this container.

The elements of the container (#Category) are called *categories*. The connection Cn is called a *defining connection* of this category. The operation Op is called a *defining operation* or an *evaluator* of this category.

Definition 5. Container (#Rule) is defined in such a way that every of its elements has the following properties:

1. it is a dmt-object,
2. it is one-to-one mapped to the pair ⟨Cn, Op⟩, where Cn ∈ (#Connection), Op ∈ (#Operation); and,
3. operation Op, by means of the connection Cn, carries out a test whether the rule is valid, i. e. the operation returns the value True if the test is successful, and value False in the opposite case.

The elements of the container (#Rule) are called *rules*. The connection Cn is called a *specifying connection* of this rule. The operation Op is called a *specifying (or testing) operation* of this rule.

The edges 01 and 02 in Figure 1 link a category with, respectively, a *defining connection* and an *evaluating operation* of this category. The edges 03 and 04 link a rule with a *specifying connection* and an *testing operation* of this rule. The Diamond Diagram contains, moreover, the so-called *P-edges* (projection edges) that link individual objects with a connection that describes the given relationship between these objects.

The Diamond Tool has to be provided with a set of operations in order to permit its utilization and modeling of business rules by means of it. These operations are divided, according to their semantics, into families of operations. We need a family of operations for creating, deleting, updating, making accessible and recognizing dmt-objects (CREATE, DELETE, WRITE, READ, OBTAIN, LABEL), a family of operations for "traveling" around the circumference of the Diamond (GET_CONNECTION, GET_OPERATION, GET_CATEGORY, GET_RULE), and a family of operations for work with P-edges (projection of a static connection onto its component, adding and removing an element to/from a dynamic connection, testing if an element belongs to a connection, etc.).

In the following text, we will discuss in greater detail those operations that make the Diamond Tool a universal modeling tool. Edges R1, R2, R3 and R4 are jointly called *R-edges*. R-edges are used for execution of transition between mentioning (operation MENTION) and using (operation USE) and vice versa. If we stand on some particular object of the class (#Object), which we have mentioned somehow, then by the operation USE we cross, along the respective R-edge, to the represented connection (#Connection) or operation (#Operation) or category (#Category) or rule (#Rule), which we can then use directly. The particular operation is specified as follows (+ stands for an input parameter, and − for an output parameter):

2.1. use_obj(+Idobj,-Idnode)

Used for acquiring a dmt-object – an instance of a Diamond vertex that is represented by the given object, i. e. for transition from the centre of the Diamond to the respective vertex. The operation fails if the given object does not represent anything.

Operation USE may fail on objects belonging to (#Object) that are a "mere" object and do not represent any category, operation, rule or connection. An example of such an object can be a concrete document.

2.2. mention(+Idnode,-Idobj)

If we are on one of the vertices of the Diamond Diagram, then the operation MENTION takes us, along the respective R-edge, to the representing object which we can then mention directly. Formally:

Used for acquiring a representative object of a given dmt-object that is an instance of one of the Diamond vertices, i. e. for transition from the given vertex to the centre of the Diamond. If *Idnode* is an instance of a dmt-entity (#Object), then *Idobj* will be unified with *Idnode*.

2.3. use_con(+Idcon,-Idobj_list)

Operation USE applied to a connection belonging to (#Connection) permits to use this connection (to find out what is connected to what, or possibly, to have the possibility to complete a record about a particular connection). Formally:

It returns a list of objects – its components *Idobj_list* – to the given connection *Idcon*.

2.4. use_cat(+Idcat,+Arglist)

Similarly, the operation USE applied to a category belonging to (#Category) permits to use this category (to find out about its current content, whether a given object belongs to it, or possibly, to add further elements to this category). Formally:

Using a category means to run an operation, linked with the category by the relation 02. The arguments of this operation are the category used *Idcat*, a connection, linked with this category by relation 01, and arguments listed in the list *Arglist*.

2.5. use_ope(+Idope,+Arglist)

Operation USE applied to an operation belonging to (#Operation) simply executes this operation. Formally:

It runs a given operation *Idope* and gives it as its arguments the content of the list *Arglist*. Arglist may contain both instantiated as well as free variables. What happens in case of free variables in *Arglist* is determined by how the used operation was defined within the respective operation CREATE.

2.6. use_rul(+Idrul,+Arglist)

Finally, operation USE applied to a rule belonging to (#Rule) runs execution of a test that checks the validity of the given rule. Formally:

Using a rule means to run an operation linked with the rule by the relation 04. The arguments of this operation are the rule used *Idrul*, a connection, linked with this rule by relation 03, and arguments listed in the list *Arglist*.

Thanks to the universal construction of categories we can work with different types of categories. For example, we can work with categories specified by the list of their elements (the specifying connection of a category then contains directly the elements of the modeled category), or with categories that use their relationships

to other categories for the evaluation of their own content, or fuzzy categories, or categories that are evaluated by means of neural networks. Similar freedom can be applied in rules creation (#Rule).

Thanks to representative edges R1, R2, R3 and R4, we can mention any of the entities in Figure 1 – i. e. to relate it with other entities, categorize it and fill its attributes. We can create connections, e. g. between categories (more accurately, between representative objects of these categories), or between categories and rules, we can categorize the categories, etc. This is the major difference when compared to the majority of modeling tools where the modeled world is strictly separated from the world by means of which we model. There is only one world for DMT. Further, in DMT it is possible to integrate models of various extent of abstraction and diverse purposes. Thus, we can create one model, and another above it which is used for tuning of the first model; and still another model above these two that learns from their interaction.

The Diamond is used within business as follows: when we support execution of business processes, we focus on the centre of the Diamond. We record and evaluate information about objects on which we focus, and we do so by means of the Diamond vertices. When we improve the model for support of business process execution, we focus just on categories, connections, rules and operations, i. e. on the diamond vertices. Thus we may imagine that alternation of utilizing knowledge and developing knowledge is represented by regular "throbbing" of the Diamond, as if it were a heart of a living creature.

DMT is implemented in Prolog (current version is 1.4.3) and it is used as the core for building of other systems in the form of its plug-ins.

3. IS ARCHITECTURE

One of such plug-ins is the knowledge agent (KM agent) that is the key component of information systems built in the eTrium technology. In this chapter we will present further components of the eTrium technology and the way they communicate.

Design of the architecture is shown in the diagram in Figure 2. The meaning of its three key components (Communication and Interface – Database – KM Agent) is the following:

Communication and Interface means a web solution, i. e. a company intranet; the logic of display and communication of information (in the directions system-user and system-system) is programmed there.

Database is a store of facts prepared in such a way that it is easy to report them from the database according to arbitrary requests. Technologically, it is solved as a relational database.

KMA (Knowledge Management Agent) is a program, by its nature belonging to the sphere of AI, that implements the whole business logic with the help of a centralized knowledge base; the knowledge (management) agent can provide for work

with knowledge in the rhythm of the Diamond's throbbing, because technologically it is a plug-in to DMT.

The basic principle underlying the co-operation between these components is: The component Database serves as a data store. It records not only facts explicitly entered by users, but all other facts that are inferred from the entered facts by the KM agent (contrary to deductive databases, where facts are inferred during queries). The component Communication and Interface then benefits from the fact that all necessary facts have been already prepared and inferred in such a way that it can read them directly from the Database. Its program code therefore does not include the knowledge of how to infer facts (for example, facts about the approval procedure of these documents) from another kind of facts (for example, facts concerning the matter-of-fact content of documents).

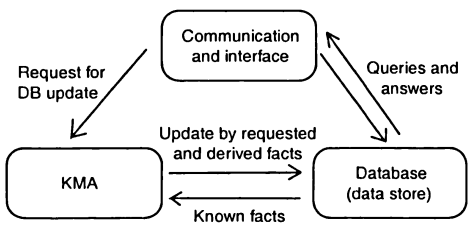

Fig. 2. Architecture of IS in the eTrium technology.

We will show now that a system built in this architecture can be used both as an IS for the support of operative execution of business processes, and as a modeling tool in which it is possible to develop and modify its way of functioning (in the role of an IS).

The description of logic of communication (co-operation) between components is divided into four cases [13]. Case A describes routine queries for information during the support of operative execution of business processes in a company/organization. In case B, the same system is used for a necessary update of the state of the information database. In both cases the system is used in a situation when the information process is *used*. Case C describes tuning of business and the model. Here, the same system is used in a situation when it is necessary to *mention* (i. e. to analyze and develop) changes in operative processes. Case D describes revisions of strategy and creation of a model. Again, it is a situation when it is necessary to *mention* methods of process execution.

Case A. A user or another system requires an answer to a query: The pair of edges "Queries about the state of the database, and answers" represents internal communication between technological components in the process of answering queries. The component "Communication and Interface" provides that the system environment is able to formulate a query and send it to the component "Database". The component "Database" interprets the query and returns an answer (in the form of a list of facts) to the component "Communication and Interface". This component then provides

display of information for users or another system in the required form. Correctness of the inquiry action (no interruptions caused by an update, etc.) is guaranteed by the component "Database" by its inner mechanism-which is a common function of every reasonably usable database engine (DBMS). The KM agent therefore does not participate when facts are read from the data store.

Case B. A user or another system wants to update the content of the facts database: When a user or another system expresses a "Request for update of the database" (see the respective edge in the diagram in Figure 2), then:

1. The component "Communication interface" allows to formulate this request (provides the necessary form) and sends the update request to the KM agent.
2. For each update request the KM agent asks the database for known facts that concern the record being updated (relevant context).
3. The KM agent, on the basis of knowledge base and relevant contexts, infers further update actions that have to be done.
4. The KM agent executes both requested and inferred update actions over the database in the component "Database".

Case C. Users find out that it is necessary to modify business rules or the method of their IT support:

There is a range of facts and rules of the KM agent that can be changed by the user through forms offered by the component "Communication interface". This range of facts and rules constitutes the "tunable" part of the model. The change of the KM agent is carried out as follows:

1. the knowledge base is updated
2. facts that are stored in the database are made consistent with the new knowledge base.

This procedure may have the following form: in the database, all the facts that were inferred by the previous version of the KM agent are forgotten and they are inferred anew by the new version of the KM agent. The whole process has the form of an update transaction, analogically to case B.

Case D. Users find out that it is necessary to revise the whole business logic or methods of its IT support:

When an agreement on the changes has been reached and new (modified) sets of rules, facts and categories accepted, a change in the system is carried out in the following steps:

1. The state of the KM agent before the change is archived. The data base of the "Database" is archived.
2. A new version of the KM agent, enriched with new categories, facts and rules, is created.

3. New business rules and all their implications are tested. Debugging rules, completeness tests, etc., are used for this operation.
4. Review of the results of testing of the new business logic is carried out with the client.
5. The data in the component "Database" are made consistent with the current state of the knowledge base by the KM agent.

The system is ready to be used with the new "business logic", i. e. new business rules and rules of business support by means of ICT.

4. THE KNOWLEDGE BASE OF KMA

Now we will deal in greater detail with the structure of the conceptual system of the KM agent and its realization in the Diamond Tool.

First of all we need to separate the participating objects classify them into types. This means to create a set of categories with such a characteristic that each object belongs to only one category of this set (a set of categories is modeled as a category the elements of which are categories). For example, the KM agent for a concrete application of the eTrium technology that is called eDialog (cf. [12]) has the following primary types: "Document", "Publication Project", "Editor". The fact that a particular object is of a certain type is expressed by a fact in the form: **object belongs to a category**.

Further, it is necessary to classify objects more finely according to their properties. Again, we will use categories for this classification. It generally holds that we are interested in those categories the instances of which we have to treat differently than other objects in the field of business or its IT support. In case of documents, we will need for example the categories "Document to be approved," "Document not to be approved," "Document is displayed in the upper right window," "Document is displayed in the News window," etc. Concrete knowledge then may have the following form: "Price list of courses 2004" belongs to the category "Document to be approved."

In order to be able to find our way, we need to focus also on the classification of categories. Categories therefore will now become the objects of our focus, knowledge will be expressed in the form: category is in a category (in the sense: "is an element of a category"). Consider for example the categories "Possible approval states" or "Document displaying category". Concrete knowledge then may have the form: the category "Document to be approved" belongs to the category "Possible approval states".

Another kind of knowledge is expressed by means of relationships between categories. Frequently, we have to express knowledge of the type: if an object is in category X, then it has to belong to category Y – e. g. if an object is in the category "Schedule", it has to belong to the category "Document is displayed in the upper right window". These relationships between categories are expressed by means of rules (cf. the container #Rule in Section 2).

5. TWO CONCEPTUAL SYSTEMS

One of the most significant ideas of using the Diamond within the eTrium technology is that the KM agent has two conceptual systems at its disposal: 1) the conceptual system of the given business area, 2) the conceptual system of IT support.

Definition 6. A category that is used for the classification of objects focused on from the point of view of some business activity, and that is not used for controlling the IS behavior is called a *business category.* A category that is used for the control of IS behavior is called an *IT-category.*

The crucial role of the agent is to manage the transition from the conceptual system of business to the conceptual system of IT support. This actually means: 1) To carry out an analysis of the examined record and its relevant context by means of the conceptual apparatus of the given business area (cf. step 3 in case B, Section 3). 2) Based on this analysis, to describe the examined record and its relevant context by means of the conceptual system of IT support (ibid.).

An example may be the examination of business properties of a document, and, finding out that it is a "price list" that has already been "approved" (concepts from the business area), this document is placed into "documents that should be displayed on the web" (an IT-support concept).

The main reason for the existence of two conceptual systems of the KM agent is the attempt to simplify and streamline the design and maintenance of an IS. The reason for the existence of some category in the conceptual system of the IS is the requirement that IS should treat the elements (objects) of this category differently than other objects. Identification of objects that should be treated uniformly by the IS must be straight and free of business properties.

The advantages of this approach can be shown in the following example: When a programmer asks: "How can the program recognize those documents that can be already displayed on the web?", there is an immediate answer to this: Those, that belong to the category "documents that should be displayed on the web". Applying a traditional solution, we would have to explain to the programmer something like: Those, that have to undergo an approval procedure and are approved of, or those that do not need to undergo the approval procedure and has been prepared. Then the programmer would encode this knowledge somewhere into the program (in other words: he would represent the knowledge by a code of an imperative programming language, which is one of the least suitable methods of representing knowledge whatsoever – and it is really surprising that it is still the most used method).

When using the eTrium technology, we do not have to explain anything to the programmer. It is sufficient to tell him the category in which he can find objects of required properties (in our example it is the category "documents that should be displayed on the web"). Elements are placed into the category by the KM agent according to rules in the knowledge base. If it is necessary to change these rules (i. e. to adjust the information system to current requirements of the client), we do not need the programmer for this. It is enough to change the knowledge base and the information system can immediately start working according to it (cf. Chapter 3,

case B).

In this way the eTrium technology differs from traditional approaches, where it is necessary to find the respective programmer, explain to him what we want, then he has to identify the place where this particular knowledge is encoded, he has to re-code it and, finally, he has to verify that this re-coding has no side effects. In the eTrium technology, programmers need not (and must not) be interested in business properties of objects. Only IT properties are relevant for them. As a result, the testing and verifying of the whole information system is also simplified.

6. KNOWLEDGE BASE DESIGN AND MAINTENANCE

It is clear that conceptual systems of knowledge agents managing real life information systems are rather extensive, similarly to rules systems in BRE based systems [2]. Therefore, it is necessary to support somehow the design and maintenance of these conceptual systems. We will describe two possible methods:

The first method is based on using categories as it has been described above. The objects of focus are now categories and rules used for the specification of a KM agent. We introduce new "tuning/testing" categories, such as, for example, "Rules not verified yet", "Business categories that are not mapped to IT categories", "Business categories not approved yet", "IT categories yet not supported by SW", etc. In other words, the "conceptual scaffolding" by means of which we "build" the KM agent can be built in the same way as the KM agent is built.

The other method is based on the method of automatic verification. We want to have verified that something cannot happen (e. g. that a document that has not been approved yet will not be published on the web) or when something happens, that it happens in a particular way (e. g. that in the news column only the news will be displayed and nothing else). To find this out, we can use some of the algorithms of artificial intelligence for state space searching, or use some of the methods for automatic verification of knowledge systems.

7. CONCLUSION

Efficient handling of knowledge is the key that opens the way to an innovative organization or firm. By means of knowledge we control processes. If we want to improve our business processes, we have to utilize efficiently the knowledge that controls these processes. The Diamond Tool was introduced as a means for knowledge management together with its universal modeling capability. Its MENTION and USE operations bring a consistent and easy way of the development and utilization of business and IT knowledge. However, though this solution seems to be sufficient for handling of knowledge, it does not meet the most important requirement – efficiency in practical utilization. Therefore, a further step is necessary – the eTrium architecture. As we have described above, the eTrium architecture is based on the combination of three different technologies: a Java object-oriented code for the communication interface, a database engine for universal data store, and an AI approach to the knowledge management agent. The KM agent is built as an extension of the Diamond Tool.

The structure of the database (a universal data store) is an extension of the Diamond structure. The world view as defined by the Diamond is also used within the communication interface; and this particular solution presents also the easiest way of achieving universality of this interface. Therefore, we expect that information systems built within the eTrium technology will have many fundamental advantages, compared to traditional solutions. Some of them could be:

- Substantial shortening of the time necessary for the implementation of an information system – the programmers do not have to encode knowledge into programs, the creation of the knowledge base of the KM agent is just about conceptual mastering of the given business area.
- Possibility of continuous and smooth adjustment of the running information system to the changing needs of the client by means of changing the knowledge base, without lengthy and risky re-programming.
- Substantial increase in quality of the information system – the design can use methods of automatic verification of the knowledge base.
- Existence of a constantly up-to-date description of business and IS (knowledge base). The business description does not lie idle in a drawer, but it determines the run of IS. At the same time, the logic of the information system is not enchanted in any program code or potentially out-dated documentation.

In this technology, the key role is played by the knowledge agent that manages knowledge of the information system. The motivation was not a futuristic vision of intelligent information systems of the future, but the still more apparent necessity to build information systems in lesser time and of higher quality, and especially, to make them continuously adjustable to changes in the supported business area without constant expensive, lengthy and risky re-programming.

(Received November 2, 2003.)

REFERENCES

[1] Business Rules Group (R. G. Ross and K. A. Healy, eds.): Organizing Business Plans – The Standard Model for Business Rule Motivation. www. BusinessRulesGroup.org.

[2] Business Rules Group (D. Hay and K. A. Healy, eds.): Defining Business Rules – What Are They Really? www.BusinessRulesGroup.org.

[3] Business Rules Group (R. G. Ross, ed.): Business Rules Manifesto. www. BusinessRulesGroup.org.

[4] E. Gottesdiener: Business RULES – Show Power, Promise. EBG Consulting, Inc. Application Development Trends *4* (1997), 3. www.ebgconsulting.com/powerpromise_article.htm.

[5] J. Sinur: The Business Rule Engine 2003 Magic Quadrant. Gartner, Research Note Markets, 2003. www.gartner.com/reprints/fairisaac/114166.html.

[6] H. Herbst, G. Knolmayer, T. Myrach, and M. Schlesinger: The specification of business rules: A comparison of selected methodologies. In: Methods and Associated Tools for the Information System Life Cycle (A. A. Verijn-Stuart and T. W. Olle, eds.), Elsevier, Amsterdam 1994, pp. 29–46.

[7] www.pega.com

[8] F. Procházka: Modeling of Cognitive Processes. Diploma Thesis. Masaryk University, Faculty of Informatics, Brno 2002.
[9] Z. Staníček, M. Benešovský, and K. G. Jeffery: Formal specification and tools for strategic planning (An introductory discussion on the StraDiWare project). In: Proc. Systems Integration 97, Prague 1997.
[10] Z. Staníček: Universální modelování a jeho vliv na tvorbu IS (Universal modeling and its influence on IS creation). In: Proc. Conference DATAKON'2001, Masaryk University, Brno 2001.
[11] Z. Staníček: Knowledge management: The words and the actions. J. Per Partes on Knowledge Management, Knowledge in Action, Per Partes Consulting s.r.o. 2002, 119–125.
[12] Running applications based on eTrium architecture: eDialog. Expertis s.r.o., Praha, www.expertis.cz, EuroSet, Clever Office s.r.o., www.euroset.cz.
[13] Z. Staníček and F. Procházka: Technologie eTrium a její aplikace (eTrium technology and its applications). In: Proc. Conference DATAKON 2002, Masaryk University, Brno 2002, pp. 113–132.
[14] Project STRADIWARE, contract No. COPERNICUS 977132, web site: www.itd.clrc.ac.uk/activity/stradiware/.

Zdenko Staníček and Filip Procházka, Faculty of Informatics, Masaryk University Brno, Botanická 68a, 602 00 Brno, and eTrium Corporation, Inc., Technological Incubator of VUT, U Vodárny 2, 616 00 Brno. Czech Republic.
e-mails: stanicek, xprocha3@fi.muni.cz, zdenko.stanicek, filip.prochazka@etriumgroup.com