ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

B Novikov, University of St Petersburg, Russia and J W Schmidt, University of Hamburg, Germany. (Eds)

Advances in Databases and Information Systems, Moscow 1996

Proceedings of the International Workshop on Advances in Databases and Information Systems (ADBIS '96). Moscow, 10-13 September 1996

Designing Databases Using the Activity Paradigm

Y. Amghar and M. Meziane

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

Designing Databases using the Activity Paradigm

Y. Amghar

LISI - INSA, 20 Avenue A. Einstein

F-69621 Villeurbanne - FRANCE

M. Meziane

LISI - INSA, 20 Avenue A. Einstein

F-69621 Villeurbanne - FRANCE

Abstract

The aim of this work concerns the specification of a methodology for designing databases using the active object concept to best represent the dynamic of an information system. Two strong notions are introduced: the sensitivity and the reaction of an active object. We propose to classify real world objects following two categories: i) the structural active object that describes an object offering services when requested and reacts to database events produced in its environment and ii) the service active object that represents an activity of the information system. These two categories correspond to structuring, communication and processing aspects that constitute fundamental features of an information system. This paper also presents the characterization and the semantic of these objects through a model. Later a methodology to design new databases starting from the message flow of an organization is presented.

1 Introduction

Object oriented methods have been a dynamic sector in recent years. This is explained by a need of renewal in design methodology imposed by new enterprise requirements or the temptation to correct traditional method insufficiencies. Generally, an Object oriented method presents a basic concept : the object. It offers, in addition, structuring mechanisms of an object schema as the generalisation (specialisation) and the aggregation (composition). However, with regards to the dynamic aspects that are, partly, modelled by procedures associated with objects, method mechanism suffers from an absence of representation because the reaction part of an object (reaction to an event or to a constraint violation) is completely ignored. The active object paradigm is a natural conceptual extension of the object concept. This extension is a semantic improvement based on notions of sensitivity, reaction and behaviour that contribute to the definition of the object as an active entity in the real word [1, 5, 11, 9]. These notions allow the object to take into account its environment and to possibly react with a spontaneous manner. The active object, as all objects, has a static aspect including a structure (or descriptive) and a function set that we will call basic functions representing what the object knows to do independently from any context. The object becomes active when it has a dynamic aspect that allows it to achieve simple or complex activities in response to events or to an internal decision. The active object is also sensitive to its environment and consequently has to have different activities. Each active object is described by a structure and basic functions that are necessary to its classification, to its functioning and to its identification. The whole constitutes the static aspect that represents what we call the *genetic* of an active object, that is all it has at the creation moment. The different states by which the active object will pass during its existence constitute an enrichment to lead increasingly complex activities. It is through basic functions and to the knowledge of its states that the active object can lead its activities. The activity of an active object is a non-determinist process allowing the distribution of results, the achievement of a service or the change of state. The activity is translated into a sequence (non determined before) of basic functions. It is triggered by events (internals or externals) to which the active object is sensitive. An event is an information coming from any object to communicate (to emit a result) or to inform its evolution (change of state). The event does not inevitably address a precise object. Various works deal with the notion of events in objects [7, 10]. Without being nominal, the event has to be identified by its name and by the name of the transmitter object. The event is responsible for the object action triggering. It can be also responsible for triggering

several actions on different objects. The message and the event are at the origin of any communication in object world. These two notions are complementary but not similar. Indeed, an event, can participate in the triggering of an activity : alone, with other events, several times before it is not taken in account or in correlation with time. When an object produces an event, it does not necessarily know the name of functions of the object to be reached. While a message, in the senses of the object approach, is described by the name of the method to execute (and arguments). The sensitivity of an active object is its aptitude to know the state of the universe that surrounds it and to take necessary decisions according to the state of this universe. It represents final ability to react to events that "gravitate" around it. The active object therefore has a reaction ability. Only events that can influence the object are taken in account. One can simply say that the sensitivity of an object represents all events influencing it.

In the classical object approaches supporting object oriented DBMS [20, 13], often one means with the object behaviour, a set of methods which are executed when adequate messages are transmitted. These methods are considered as procedures. The term of behaviour, does not cover all the reality since : the object does not react when it does not receive explicit message even if its environment has changed, the object does not inform the other objects of its possible state and evolution changes and finally the object does not self-trigger.

Our proposal can be considered as a contribution for an extension of the object-oriented methods to active objects. This extension includes all features of an active object (sensitivity, reaction, ...). Thus, the introduction of object behaviour and notably active aspect during the first phase of modelling has the great advantage of preserving the isomorphism between reality and modelling result. The object that one describes is then an active object. Many objects extracted from the real world are active by nature and therefore have a reaction that must be taken into account during the modelling process.

This paper is organised as follow: in section 2 we present the way to model an active object in an information system. We also discuss how objects communicate amongst themselves. Then, in section 3, we define a knowledge model based on an active object that embeds its sensitivity and reaction. Activity is also taken into account and considered as an active object. The section 4 presents the methodology steps for designing an information system based on this model. This same section discuss the needs of an active DBMS for implementing such a model. The section 5 concerns a general framework for active database development. The related work are presented in the section 6. Lastly we conclude by the section 7.

2 Information system modeling

Information system are usually modeled through two or often more models to capture both the static and dynamic aspects. These models are often very different since the used formalism is different (entity-relationship for static aspect or object oriented model, functional approaches for dynamic aspect). The notions manipulated in the static and dynamic fields are based on different concepts. Validation techniques need distinct approaches. It seems more interesting to model information system by using a uniform approach for both structural and dynamical parts. Furthermore, the emergence of active database technology [8] enforce us to develop new methodologies of database design. For that we propose an object-oriented approach based on active object concept to design an information system.

At the design level, an active object is a complex system that has to be modelled in terms of sensitivity and reaction (see figure 1). Its good operating supposes a consistency, called local, that can be modelled upon reactions that are developed by the object according to events. The fundamental role of an active object is to process events that it receives. Its behaviour is translated into a set of rules that are executed by the object in order to process such events. These rules, which can be expressed by ECA (event - condition - action) rules, allow to build plans of execution in most complex cases, where procedures are materialised by the action part of the rule. These procedures can also produce events (local events) that are consummated by the same object. For that the active object is able to trigger itself. Furthermore, these rules can have different natures with respect to whether they focus on processing or on data. We will talk about rules of production in the first case and rules of integrity in the second case.

An information system is constituted by different areas of application in which evolve a given number of objects. Dynamic is translated into the communication between these objects and activities developed by active objects. One defines the environment of an active object by the set of objects that can receive messages or events.

Figure 1: Meta-model for information system

3 Objects within an information system

An information system can be decomposed as several application fields for describing organisation activities or processes. To model an application field it is necessary to have concepts and links between them constituting the meta-model. An active object is the centre of this metamodel which is organised with entities described in the following sub-section.

The modelling of an information system around the active object paradigm takes into account data and treatments. This modelling follows three steps that allow us to describe the different views of an information system. These views can be classified with respect to whether one is interested by the static aspect, by the reactive aspect or by the global behaviour of activities of the system. Communications between each aspect must be studied.

3.1 Object classification

One distinguishes three object categories in an information system corresponding to components of the design phase :

1. **passive object** that stocks data and replies to messages when requested; it can be qualified as an elementary object. Traditional object-oriented models manage this kind of object.
2. **persistent active object** that stocks data, replies to messages and reacts to events; This kind of object includes rules and constraints. Active databases are examples of systems that take into account some dynamic aspects. More generally, persistent active object must express :

- a reaction to events;
- a genetic part to implement complex or simple activities in reply to events;
- a self-triggering; to illustrate this notion let us consider an aircraft example. When a fall in pressure is observed in the cabin of a plane, that entails by an automatic manner a process of adjustment to increase the pressure to a nominal value. Demons (or triggers) [14] can contribute to the implementation of the spontaneous activity of an object.

A persistent active object is described following a certain number of clauses. The attribute clause allows to describe static data to be manipulated. The methods clause indicates the genetic of the active object that is to tell all methods that it knows to execute when they are called. The sensitivity clause designates the sensitivity of the active object embedding all events implying a reaction from an active object. Each event belong to a type with respect to whether it designates an external or internal event and with respect to whether it is simple or composite (event built from conjunction, disjunction and sequence operators). The reaction clause allows to describe necessary rules for the intelligent behaviour of the active object. Note that if it concerns a rule of integrity, then events concern operations of creation, deletion and update of attributes of an object.

Integrity constraints represent an important component of the dynamic of an information system [2, 15, 4]. Their integration within a model is going to allow the database to provide a more complete image and with more semantics of the real world and therefore to restore at any given time consistency and updating information about objects. A modification of database can lead an inconsistency state when an integrity constraint is violated: it is necessary to notify the user about the violation of the constraint or to execute a procedure allowing to restore the database consistency. For example, during the processing of orders, a constraint of the type *ordered quantities of products of a given type have to be a multiple of 50* consists in checking the validity of the constraint : in the case of a violation, it invites simply to ignore the order considered.

More generally, a constraint can be considered as a particular case of a rule since the part action (conclusion of the rule), although existing, is not the most important for the object semantic. On the other hand, when the premise part does not concern a constraint but translates a simple condition, then we are in the presence of a business rule. For example, if clerk service is at least 18 months and that its salary is equal to the minimum salary then it is necessary to increase his salary. Finally, production rules generalise constraints and triggers in the senses that the part action becomes more complex and where conditions depend on other rules. We give below an example of a such object :

Description of object : Stock

attributs :

orderedQuantity : real,

currentQuantity : real,

methods :

update(x : real) return boolean;

constraints :

on update

if currentQuantity > threshold

then raise(supplying);

3. To remain in the context of object-oriented approach, one defines **the activity** as an active object which performs complex services, that is a service active object whose purpose is to represent a given application. It is also an object which reacts to events but it does not receive a message. An activity needs passive and other active object to achieve a task. These requirements are called resources. Their enumeration aid the designer in the identification phase of objects. This kind of object is more detailed in the next section.

All methodological steps consist of identifying all objects of the real world by listing them with respect to those three categories of objects.

3.2 Activity description

Activities which use real word knowledge's are a fundamental part of an organisation; order management in companies, bank management, customer following are such examples of main activities that can be decomposed in more details as procedures distributed among objects of information systems.

So an activity is a procedure set executed in order to accomplish a mission. An activity is triggered by event or event synchronisation. It uses object properties as resources and chains defined object procedures in a given order. One distinguish local and global activities. The later use several system objects. Activities which co-operate build new activities. Co-operation modes use synchronisation mechanisms as synchronous and asynchronous modes. In our model, an activity is considered as an active object. One defines a resource as any object which is necessary for activity execution.

Activity properties

1. activity must carried out up to its end and leave used objects in a consistency state;
2. activity is triggered by event synchronisation;
3. activity uses a finite numbers of procedures;
4. activity re-triggered by the same events and using the same object states produces similar results;
5. activity can be decomposed in several sub-activities each one having at least one main activity properties.

Service active object is described with several clauses, each one allowing to take into account the different notions:

```

service object <activityName>
uses <list of resources>
sensitivity
input events <IE=e1, ..., en>;
output events : <OE=e1, ..., ep>
/* constraints to observe for activity execution */
postConditions
<ex : ? customer exist and has odered more than 5 products>
preConditions
<ex : ? customer has been invoiced>
procedures
<proc1(p1, ...);>
<proc2(p1, ...);>
...
end procedures;
reaction
/* script for procedure execution */
on <e1 in IE>
if <condition1>
then call proc1 [else] <exit>;
on <(e1 and e2) in IE>
if <condition2> then call proc2
on activity_call
if <postconditions satisfied>
then enable IE and OE;
end activity;

```

This activity description emphasises many components. Activity sensitivity is indeed translated by the input event list which permits activity reactions to be triggered. Pre and post conditions are respectively necessary before and after the activity execution, allowing such a consistency execution. These conditions can be formulated with a first order logical expression. This description allows us to describe all procedures that compose an activity. Last the reaction part in the activity description allows a response to some events produced in the environment with respect to a chaining of rules.

	data structure	passive behavior	active behavior
passive object	attributes for object description	methods	
persistent active object	attributes	methods	integrity constraints business rules
activity (process)	resources (object set)	procedures post-pre condition	sensitivity reaction scripts for business rules elaboration

Table 1: Structure of object categories

4 The methodology phasis

In general way, our methodology does not differ from OMT [17] or OOA [6] methodologies in terms of phasis to respect. But at level of each phase, one notes some important differences. Indeed in the classical methods, the object constitutes the single entity of modeling when an activity and active object are the concepts used in our approach. The rule modeling is done at the design level and at the same moment of the activity and object designing (see figure 2).

4.1 Phase 1 : analysis.

In this stage, one must elaborate the communication model that shows message exchanges between different actors and activities of an organisation. The analysis of messages allows to identify objects precisely. More precisely, one identifies activities or processes which are more easily to extract from a real world. Each activity allows to extract the necessary objects. The designer taks is then more easier.

4.2 Phase 2 : design

This phase concerns the description of objects and their behavior. The objects are identified in the previous stage with respect to the three categories described in the section 3. This stage is composed of three steps.

Step 1 : activity and object description. In this step the designer must describe, for each category, the objects in terms of data, procedures and rules. In addition for object concerning the activities, one must precise :

- The description of resources used by this activity; that is the list of objects that are necessary for the activity execution. For example, the order activity needs customer, order and stock objects.

Figure 2: Mapping of the design according to three axis.

- The description of communication aspects (sensitivity) by giving the list of input and output events which constitutes the user interface of the activity with the other objects. For example an order management is sensitive to customer phone calls and produces invoices.
- The description of pre and post conditions that guarantee the consistency of the activity execution. These conditions allow to check the good environment of the activity context. For example an order management needs a positive balance of a customer before any order acceptance.
- The description of procedures in terms of signatures; that is the name and the parameters of each procedure.
- The description of the activity reaction by establishing scripts for each procedure execution;

The resource list determined in the previous step aids to build the database schema; that is, the structuring of objects. Thus, for each resource, the designer must give :

- The description of attribute and method clauses.
- The event identification for sensitivity description.,
- The identification of rules.
- The establishing of event-rule relationships for reaction description.

After these operations, the designer can specify the specialisation and relationships between all the objects.

Step 2 : behavior modeling. One attaches graphical models to represent each part of the methodology. The behavior representation is done following different approaches. One can cite the Petri net approach where the object

states correspond to places of Petri nets and activities to transitions. Instances of object types, which reside in states, correspond to tokens in Petri nets. However, this proposed diagram is too big for real world problems, and usability of these diagrams is not clear yet. Decision tables are also used to relate sets of conditions to prescribe actions. These tables are only appropriate for those parts of an application that are decision-intensive. Another mean to model the behavior consists of using formal specification languages such as VDM which are probably the more formal language used for the specification of sequential system. The object automaton allows to represent each part of the methodology. The graphical model (which is not explained in this paper) which support our approach is based on an automata. It includes different means for the representation of input and output events, procedures and used resources. This diagram is a mean for checking the *local behavior*.

The proposed for the behavior specification by means of a uniform model that specifies both object and process behaviors. Each behavior specification is represented by an automaton. Our model is based on active rule-oriented approach for the specification of the application behavior according to Event-Condition-Action mechanism that constitutes the key concept used in active database systems. Our approach also permit a uniform representation of business rules which are an important part of information systems. Business rules may correspond to both consistency rules that define integrity constraints on valid database states or activity rules that prescribe action or operation sequences which have to be performed. In the following, we describe the different constructs used in our model. Let us note that every constructs used in the automaton have only one semantic. In our approach, an object designates an object in the database as well as a process of the information system.

- *Event*. The automaton includes two kind of events: produced (or input) and consummated (or output) events. The consummated event concerning an object o_k is raised by any object o_i in the system. It allows a rule to be evaluated. The produced event is generated by the object after or during the execution of an action (operation). The two kinds of events must be differentiated. The event has three part: the type of event, its name and its parameters. The type of event can be internal (database event), external (user-defined event) or temporal. The name of event corresponds to the designation of the event and the parameters correspond to attributes or values. At this level, all events are primitive. For the composite events, we used the synchronization mechanism. Formally, an event can be noted as: $TE_i(p_1, \dots, p_n)$ where $T = E$ for external event, $T = I$ for internal event and $T = T_p$ for temporal event. T is considered as the event type.
- *Synchronization*. The synchronization is used for describing composite events. The composite events are a combination of primitive events with additional properties. The synchronization includes the type of operator. We distinguish between different types of operators: boolean operators (disjunction, conjunction, negation), sequence operators such as periodic and aperiodic operators. The input of synchronization is constituted by different primitive events and the output is the composite event. A synchronization is characterized by a life cycle corresponding to a waiting state (when any event is present), an activable state (corresponding to the presence of some events but not all events) and active state (when all events participating to the synchronization are raised).
- *Object state*. The model also allows the representation of object states. An object state is described by an identifier which represents an order number during the life cycle of the object. The state notion proposed presents a different semantic in comparison with that used in the literature. In addition, the object state has a status and a label. Status of objects are predefined, we distinguish two object status: idle status, corresponding to a stable object state and waiting status that corresponds to an event waiting. One considers this status as incoherent because the object has not finished its treatment depending on an input event. Finally, the state labels are comments on object state and allow to specify the reason of concerned state.
- *Condition*. The condition part is tested when an event occurs. This is done through a condition evaluation. The condition can be elementary or complex. An elementary condition is either a predicate or a query over the database (e.g. a comparison of an attribute value of an object and a constant, a comparison between attribute values of two objects or a set-operation on data). Complex conditions consist of several (elementary or complex) conditions which are combined by boolean operators. The condition evaluation is achieved in term of yes or no giving two alternatives.

- *Action*. The action part is a set of operations that contains (at this stage) a brief description of the action. The action part can be viewed as an action or sequence of different actions. Actions may be also classified according the location where they are performed.
- *Resources*. The resources are properties which are used in the event processing. In the case of object behavior these properties correspond to object attributes which serves as event parameters. In the case of process behavior, these properties correspond to object which plays the role of any actor in the system.
- *Business rule*. Business rules can be viewed as state transitions which allow to pass from an object state o_i to another object state o_j . They aggregate an input event (or synchronization of input events), a condition (optional), an action and an output (or produced) event.

Step 3 : interaction between active objects. In this last step, the designer build the interaction diagram between objects in terms of message exchange and event production. The obtained model allows to verify the *global behavior* of the application. The interaction is done with both message and event mechanisms. An interaction is characterized by a synchronization and coupling mode. The synchronisation mode can be either synchronous or asynchronous and the coupling mode denotes the imediate, deffered or decoupled transaction for the evaluation of the two couples (i.e. event-condition and condition-action).

4.3 Phase 3 : logical model design.

The logical model design correspond to the translation of the result of the previous stage to a logical schema of the database. This translation must take into account the activities and rules. In more general way, activities are translated to programs or packages and active object description to classes or tables (in the case of relational DBMS). Integrity rules are converted to triggers and derivation or business rules to methods or procedures. Latter, events are mapped through message calls or database operations.

To illustrate an exemple of derivation of a part of our model we choose relationnal model for practical reasons. For the implementation we propose to perform a mapping of object categories towards relational implementation. For that, we choose the relational database management system ORACLE V.7 for two main reasons. First, ORACLE is among the active DBMSs which are widely present on the market place (and obviously for its availability within our laboratory!). Secondly, it provides some active capabilities which allow us to have natural mapping of the active object specification.

4.3.1 A brief overview

Oracle V.7 provides the database trigger concept that is a stored PL/SQL program unit associated with a specific database table. A database trigger has three parts: a triggering event, an optional trigger constraint, and a trigger action. When the event occurs, the database trigger fires and a PL/SQL block performs the action. So this concept allows us to enforce complex business rules.

Another oracles capability is the procedure notion. A procedure is a subprogram that performs a specific action. A procedure has two parts: the specification and the body. The procedure specification begins with the keyword PROCEDURE and ends with the procedure name or a parameter list. The procedure body begins with the keyword IS and ends with the keyword END. The procedure body has three parts: a declarative part, an executable part, and an optional exception-handling part. The first part contains local declarations, the second one contains statements, and the third one contains exception handlers which treat a particular cases. So with procedures, we can also enforce passive behavior or methods.

Finally Oracle provides another key concept which is the package notion. A package is a database object that groups logically related PL/SQL types, objects and subprograms. Packages usually have two parts, a specification and a body. A specification is an interface to applications; it declares types, variables, constants and exceptions. The body holds implementation details for the application. So, with packages, we can enforce activities. Now, let us see how to enforce persistent active object and activities:

4.3.2 Mapping of persistent active objects

During logical design, structure (or attributes) of persistent active object is mapped into a relational table, passive behavior (or methods) into procedures, and active behavior (or integrity constraints) into database trigger. For example, the mapping of the persistent active object stock is the following:

```
/* mapping of structure */
CREATE OR REPLACE TABLE Item(
    item_number integer primary key,
    item_name char(30) not null,
    inStock decimal(8,2) not null,
    reordered number,
    threshold decimal(8,2))

CREATE OR REPLACE TABLE Order(
    order_number integer primary key,
    order_date char(30) not null,
    quantity_ordered number not null)

/* mapping of passive behavior */
CREATE or REPLACE PROCEDURE stock_updating as
    (update_date IN DATE,
     quantity IN REAL);

/* mapping of active behavior */
CREATE or REPLACE
    trigger t_stock after update on item
    for each row
when (new.inStock < old.threshold)
begin
raise out_of_stock;
end;
```

4.3.3 Mapping of activities

The activities are derived to packages. The sensitivity of an activity specified by input events at the conceptual level, is implemented through the package specification. Events correspond to procedure calls. For example, the mapping of the activity order is the following:

```
/* package specification */
/* the specification corresponds
   to the sensitivity of an active object */

CREATE OR REPLACE PACKAGE order_activity AS

PROCEDURE order_recording
    (order_date IN DATE,
     quantity_ordered IN DECIMAL);

PROCEDURE reduction
    (ordered_quantity IN REAL,
     balcance IN REAL);
```

```

/* package implementation */
/* In this part the procedures of the
   active object are described */
PACKAGE BODY order_activity IS
DECLARE Status NUMBER := 0;
PROCEDURE order_recording (order_date IN DATE,
quantity_ordered IN DECIMAL)
DECLARE n_order NUMBER;
BEGIN
    SELECT MAX(order_number)
    INTO : n_order FROM Order;
    INSERT INTO ORDER
        (n_order, order_date, quantity_ordered);
    Status := 1;
END;

PROCEDURE reduction(ordered_quantity IN REAL,
                    balannce IN REAL)
BEGIN
    IF ordered_quantity > x
    THEN
balance = (balance - (0,1 * balance))
Status := 2;
    END IF
END proc;

POCEDURE Termination
BEGIN
Status := 0;
END;
/* The following procedure corresponds
   to the script of an active object.
   When it is called it choose the good
   procedure to call */
POCEDURE Script(event INTEGER)
BEGIN
    IF event = e1 THEN order_recording(...) END IF
    IF event = e2 THEN reduction(...) END IF
END;
END Order_activity;

```

5 Active database engineering

We propose briefly in this section a general framework for the development of active database. The new architecture of applications using active DBMS is composed by three functional blocks. The first one corresponds to user-interface that are located in client stations and access to distant database. The second block concerns code executed within operating system environment. The last block concerns the database development. The ECA rule paradigm leads to new architecture of the database since a great part of treatments are designed during the database design.

5.1 Application domains

It invite to consider active DBMS in endorsement of systems of information, eg as implementation target. In this framework, it is important to propose models and aid tools for the conception exploiting the active systems. The utilisation of the active object as modeling mechanism is very relevant in some applications for which the reaction aspect is preponderant. It is the case notably of hyperdocuments in which links are active entities that react to events for the purpose to reach some nodes that satisfy conditions. The active object is thus an alternative to others navigation methods of an hyperdocument. On an other plan, the active object allows an integrated management of the documentation of an enterprise by taking some counts the management of different actors of an organization. An other very important area of application is that simulators for which the active object is alone to be able to take into account the whole dynamic aspect of this kind of application. Later active DBMS have an important role to play in the real-time application since integrity and temporal rules are various and numerous.

5.2 Case tools

The integration of rule paradigm within the modeling of an information system leads to readapt the existing CASE tools. Indeed the activity modeling is required during the modeling process. The taking into account of different kind of rules identified by the designer has to be expressed and parsed by the case tool. Whereas in the most used case, integrity constraints for example, are often represented as comments within the model. Case tools must offer the possibility to achieve an inference upon rules in order to avoid cycles in the call graph. According to the proposed methodology and the modeling of process (activities) the conception of applications does not make a separation between data and treatments since an object (persistent) or a process (an activity) are modeled following the same paradigm. The ability of active DBMS to manage rules allows to obtain new architecture of applications (see figure 3). The integration of the most important part of the application code exist with the database. The interaction between active DBMS and operating system decrease since the impedance mismatch is more lower than with traditional DBMS. However researches must focus on the communication of client applications and active DBMS.

Future case tools must consider the modeling of database and teh modeling of processes following a uniform way. Since the ECA rule paradigm becomes so important whether at the data model level (through integrity constraints) or at the behavior model level (through business rules). In general way, rules are integrated at different levels of the modeling process. We can summarize this integration as follows:

1. data model level: integration of integrity constraints. The event part of the ECA rule indicates a database operation such as Insert, Delete, Update.
2. state diagram level on which appear dynamic rules. In such a diagram transitions must be constrained to guarantee a local consistency during the object life cycle.
3. activity model level (or modeling of processes). At this level the dynamic is mainly represented through business rules in order to model a process in terms of scripts chaining a set of rules.
4. interaction level in which the communication between objects is represented. At this level temporal constraints are included. In this case the event part of the ECA rule concerns the time following different forms. This level is an integration level in order to check the relationships between three other levels.

5.3 Data repository

The development od these kinds of tools requires the elaboration of a meta-model. The goal of such a meta-model is to implement all concepts that are necessary to manage persistent active object and processes. It is important to offer meta-classes to allow the description of automata (such as the following meta-classes : event, condition, action, synchronization, ...). Each model defined by a designer consists of an instance of that meta-model. The meta-model must be more general in order to implement such dynamic model over any king of DBMS (relational or object-oriented).

Figure 3: Architecture of active applications.

6 Related works

Up to now little works have been dealing with the problem concerning designing of active database applications. Nevertheless one can cite some works. For example, in the relational approach, works of Tanaka [18] propose to extend the Entity-Relationship (ER) termed (ER)2 model by incorporating active database behavior in the form of events and rules as entities of the model. From the dynamic representation of active behavior in an (ER)2 diagram, a high level petri net is derived in which the places are events and the transitions are rules. Concerning the object-oriented approaches, one can cite two works. Firstly, the Object-Oriented System Modeling (OSM) method [12] allowing to describe standard data base structure as well as active behavior. OSM propose a semantic model to express database structure and a rule system to model behavior. A behavior is seen at two level: the object level, in which each object behavior is described by state nets; and an inter-object level allowing to describe how concurrently executing objects interact. The contribution of this methodology is in the explicit representation of each components of ECA rules in the behavioral model. Secondly, the method of Bichler and Schrefl [3] which has introduced active object/behavior diagrams at the conceptual design of active databases. The structural part is represented by objects diagrams which are based on the semantic data model. The passive behavioral part is described by behavior diagrams and the active behavioral part is depicted by means of activation scripts. The passive and active behavior rely on petri nets.

One notes that all of theses approaches make a strict separation between the structural aspect and the dynamical one. Theses three approaches use a semantic model for representing the database structure. The dynamic aspects

rely to Petri nets which are used for representing the database behavior. In the relational approach, Tanaka propose an explicit integration of rules in the model. The (ER)2 model presents a great disadvantage when we have a great number of rules. In the object-oriented approach, the OSM method presents a rule model for describing the behavior. This model is closely related to HiPAC which is an implementation model. Lastly, the great disadvantage of Bichlers method is that petri nets are constructs which may represent different semantic. For example, places may symbolize object states, events or conditions.

7 Conclusion

New representation models of information system although based on approaches that seem to bring a best dynamic aspect perception have a many difficulties to be required in organisations. Indeed, these last prefer to use methods such that MERISE [16] or SADT [19], to quote only the former, for two reasons that seem us important. The first reason is haveed in the fact that a certain experience and a know-how have to be preserved so as to continue the existent exploitation. The second reason concerns the diversity of new models that, whether by lack of stability or by omission of some steps in the design cycle of information system, lead organisation designers to be reticent to their utilisation. In addition these new models do not propose solutions to consider the existent.

There are not design methods of applications taking into account new data base technologies particularly active data bases which are natural support for application behaviour implementation. The work we have presented in this paper can be considered as a step in the design of such methods. For these reasons we have proposed a methodology which is no far from systemic methods at design level. Furthermore this methodology can be supported by case tools implementing defined steps and generating models for active DBMS.

References

- [1] Y. Amghar, M. OuHalima L'objet actif: base d'une nouvelle generation de systemes d'information, revue INTER-FACES de l'afcet, novembre 1991.
- [2] Y. Amghar, A. Flory Integrity Constraints in Object-Oriented Databases in Revue d'ingénierie des systèmes d'information, Hermès Ed., 1994.
- [3] P. Bichler, M. Schrelf Active Object-Oriented Database Design using Active Object/Behavior Diagrams Proc. of RIDE'94, 4th Workshop on Research Issues in Data Engineering, J. Widom and S. Chakravarthy eds, Houston, 1994
- [4] M. Bouzeghoub, E. Simon Integrity Specification and Enforcement in Databases, in Ingénierie des systèmes d'informations, Vol. 2, No 2, 1994.
- [5] A. Buchman, J. Ziermann Building an Integrated Active OODBMS. Requirements, Architecture and Design Decisions In Proc. of 11 th Int. Conf. on Data Engineering, Taipeh 1995
- [6] P. Coad, E. Yourdon Object-Oriented Analysis Prentice Hall, 1990.
- [7] L. Dami and al., Temporal Scripts for Objects in "Active Object Environments" Report, Universit de Genve, ed. Tsichritzis, 1989
- [8] U. Dayal, D.R Mc Carthy The Architecture of an Active Database Management System in Proc. SIGMOD Conference, Portland, May 1989, pp 215-224.
- [9] C. Ellis, T. Gibbs Active objects: realities and possibilities in object oriented concepts databases and applications, Won Kim, lochousty edition ACM press 1989.
- [10] E. Fiume, D. Tsichritzis Dynamic Multimedia Objects in "Active Object Environments" Report, Universit de Genve, ed. Tsichritzis, 1989

- [11] N.H. Gehani , H.V. Jagadish Active Databases Facilities in ODE in Active Database Systems-Triggers and Rules for Advanced Database Processing, ed J. Widon and S. Ceri, Morgan-Kaufman publi., California, 1996.
- [12] S.W. Liddle and al Analysis and Design for Active Object Bases technical report of the Brigham Young University, 1992
- [13] D. Maier and al Development of an Object-Oriented DBMS Report CS/E-86-005, Oregon, Graduate center, april 1986.
- [14] M. Minsky A Framework for Representation Knowledge in The Psychology of Computer Vision, P.H. Winston ed., Mc-Graw hill, 1975.
- [15] D. Plexousakis Integrity Constraint and Rule Maintenance in Temporal Deductive Knowledge Bases Proceedings of the 18th VLDB Conference, Vancouver, Canada, 1993.
- [16] A. Rochfeld L'univers des mthodes de conception INFORSID, Nancy 1989, pages 477-514.
- [17] T. Rumbaugh and al. Object Oriented Modeling and Desing Prentice-Hall, Englewood Cliffs, New Jersey, 1991.
- [18] A.K. Tanaka On Conceptual Design of Active Database Phd-Thesis, Georgia Institute of Technology, Dec. 1992.
- [19] E. Yourdon, L.Constantine Structured Design Prentice-Hall, 1979.
- [20] D. Woelk and al. An Object-Oriented Approach to Multimedia Databases Proceeding ACM SIGMOD, Washington D.C., May 1986.