Towards Requirements Engineering of Active Database Systems

José H. Canós*, Javier Jaén** and Isidro Ramos*

*Departament de Sistemes Informàtics i Computació

Universitat Politècnica de València

Camí de Vera s/n

E-46071 València (Spain)

{jhcanos, iramos}@dsic.upv.es

**Department of Computer Science

Virginia Tech

Blacksburg (VA), USA

fjaenmar@vt.edu

Abstract

Active systems are emerging in many fields, being particularly interesting those like Active Database Management Systems which always provide some kind of reactive capabilities. In this paper, starting from the assumption that activity is an important notion not only for database systems, but also for capturing semantics in the Requirements Analysis and Specification field, we present a model of active behavior that is independent of a concrete database data model, in the sense that the set of events is not predefined (imposed by the database system used), rather it varies with each problem being modelled. We define both a syntax and semantics to formalize different dimensions of active behavior. This approach, first, extends an OO model supporting formal requirements specification and rapid prototyping (OASIS) with active capabilities and second, allows implementing information systems in different architectures by translating the active concepts of our model to those of particular active database systems (relational, object-relational or object-oriented). This is a first step towards a CARE tool for the specification and prototyping of active systems.

Keywords: Active Databases, Requirements Engineering, Dynamic Logic

1. Introduction

The work on active database management systems (ADBMS) is reaching an interesting level of maturity with a number of active systems available and, more important, a wide agreement on what the functionality of such systems should be; the ADBMS Manifesto [Dit95] defines a basic terminology and enumerates a set of required features for a DBMS to be considered as active. Essentially, the (re)active behavior is associated to the DBMS itself, which should be "able to react automatically to situations in the database and beyond" [Dit95]. This characterization is applied to (object-) relational as well as to object-oriented (OO) DBMS. In both cases, a situation is the occurrence of an event; in the relational case events are for instance the insertion or deletion of a tuple in the DB, while in OODBMS different events may be defined (creation or deletion of objects, invocation of a method in an object, etc.). No matter the kind of system, active behavior is usually associated to the DBMS, so that activity is seen as an add-on to the data model underlying the system.

This approach is, in our opinion, too low-level. From a Software Engineering point of view, a system+IBk-s active dimensions are just properties of some of its entities, and must be described at the Requirements Analysis and Specification phase of the software lifecycle. As a consequence, the set of candidate events for representing active behavior must consist of events in the Universe of Discourse (UoD), otherwise one would be taking implementation decisions at a very early phase of the lifecycle.

In this paper we are interested in extending OASIS [Pas92], an OO model supporting formal requirements specification and rapid prototyping, with active capabilities in order to make it suitable for Requirements Analysis and Specification of active systems. In this model, a set of Past Future (PF) rules whose semantics is based on Dynamic Kripke Structures (DKS) is attached to each class in a conceptual schema defining the active behavior of its instances. This behavior is not characterized by a predefined set of events related to a particular DBMS, but by events that belong to the UoD resulting in a notion of activity which is domain-oriented rather than data model-oriented.

An interesting feature of this approach is that its data model independence allows implementations of object-oriented and active schemas either over relational, object-relational or object-oriented ADBMS depending on the implementation requirements of the information system being developed. The definition of the mappings between the active notions in OASIS and the active capabilities of the different systems is currently under research. Apart from this transformational approach, a prototype ADBMS that implements the OASIS model has been developed [Can95].

The paper is organized as follows: Section 2 presents an overview of OASIS. Section 3 extends OASIS at both the syntactic and semantic levels to deal with activity; first, by introducing Dynamic Logic in order to define semantics for active behavior, second by formally defining new concepts like DKS and PF rules and, finally, by analyzing OASIS new active dimensions. Section 4 reports several aspects of our proposal together with some methodological guidelines for Active Requirements Analysis and Specification (ARAS). Finally, we give some concluding remarks and enumerate some future work in the field.

2. Introducing OASIS

Information Systems are seen in OASIS as societies of communicating objects that evolve through state changes following certain behavior rules. An object is observed in order to know some of its structural properties, which are represented in the model by introducing the notion of attribute. Each object is referred to by means of its unique identifier and an object+IBk-s state is given at any moment by the set of values of its attributes, i.e., a change in any of the attribute values will lead to a change in the object+IBk-s state. The value of an attribute may only change as a consequence of an event occurrence in the life of the object and according to a set of constraints that restrict the possible states that an object can reach during its lifetime. The cooperation among OASIS objects is established through communication by messages and interaction mechanisms. There are two communication mechanisms in our model: observations and updates, the former to obtain information about an object+IBk-s state and the latter to modify it. The distinction between observations and updates is merely syntactic: both are special cases of event sharing, an interaction mechanism resulting in two or more objects synchronizing their lives [Har92].

Structural and behavioral aspects are collected in the notion of type. In OASIS an object type is defined as a tuple T=(A,X,F,P), where:

* A is a set of attributes, which may be constant, variable and derived from other attributes.

* X = Xs +8Mg- Xa is a set of events; Xs are the services provided by the instances of the type, and Xa are the actions that might be requested to other objects in the system, even to themselves. For every action there must be a service with the same name in some type of the system.

* F= Fv +8Mg- Fd +8Mg- Fp +8Mg- Ft is a set of formulae capturing different aspects of the OASIS behavioral model; Fv are the valuation formulae that state how attribute values are changed by events; Fd is a set of derivation rules associated to derived attributes; Fp are the event preconditions which define the valid states of an object for an event to occur; Ft are the triggering rules, defining a very limited type of active behavior in OASIS objects; and Fc are integrity constraints expressed as temporal formulae that must hold over any sequence of states of an object.

* P is a term built over the alphabet Xs +8Mg- O, being O the operators of the Basic Process Algebra (BPA) defined in [Wie93]; every actual life of an object must be a subprocess of P.

The interested reader may find a formal and detailed presentation of OASIS in [Can96,Pas92].

3. Extending OASIS with Active Dimensions

OASIS, in its initial version, was conceived as a model that was very suitable for Requirements Analysis of systems of great structural complexity. To cope with this, OASIS provided a great richness of structural constructors like aggregation, specialization, and parallel composition, among others. However, limited capabilities supporting active expressiveness were available: triggering rules (a special case of condition-action rules), and processes (describing valid sequences of events or lives). In depth

studies of current research in the field of active systems [Cha93,Dit95] reveal the non existence of models an languages for ARAS and encourages us to extend OASIS with a complete set of active dimensions to make it suitable for this purpose.

An entity, and in particular an OASIS object, may behave in two general ways: first, by reacting to some kind of stimuli: another object+IBk-s service request, an event occurrence, a condition over a state; and second, by acting spontaneously following its own initiative. These two general behavioral categories will be known respectively as reactive and active behavior. Reactive behavior can be classified, according to both the type of stimuli and the originated reaction, into event-state (ES), event-action (EA), and condition-action (CA) reactions. ES reactions describe state changes after event occurrences, EA reactions cope with causal dependencies between events and actions, and CA reactions describe stimuli based on conditions over states. Finally, active behavior is purely spontaneous and is used as a way of characterizing patterns of objects+IBk- lives.

All these possible behavioral aspects have in our model a unified and well-defined syntax and semantics which is based, as presented later, on DKSs. Let us introduce some previous definitions that will help us to define both the syntactic and semantic components of our behavioral model.

3.1 Syntax

Processes, seen as combinations of services, actions and conditions over states in terms of sequential composition, choice and iteration, are the linguistic foundations for the proposed extension to OASIS.

Definition 1

A process is defined inductively as follows:

* +8Gw- is the empty process;

* a service e+8M4- Xs is a process;

* an observation o??a +8M4- Xa is a process;

* an update o::e +8M4- Xa is a process;

* check(+8GrwKQ- ,where +8Go- is a first order formula, is a process;

* if p1, p2 are processes, then p1 & p2 is a process;

* if p3, p4 are processes, then p3 or p4 is a process;

* if p is a process, then +8GE- p +8GI- is a process;

* only are processes those defined by i), ..., viii)

The symbol & represents sequential composition of processes, and or represents choice. We also admit

iteration by means of recursive processes. The process defined in viii) is called a transaction, and includes as additional semantics that of the transactions in the database world [Ram93]: an all-or-nothing execution policy and the non observability of the states an object reaches during the execution of the transaction; the symbols +8GE- and +8GI- have the same meaning as respectively the begin and commit operations in databases.

A process, as defined above, will be used to express either activity that occurred in the past (P-process) or activity that must occur in the future (F-process) having, thus, different semantics.

Definition 2

A Past-Future rule (PF-rule) has the form [p]<f>; p and f are called respectively P-process and F-Process.

3.2 Semantics

3.2.1 Formal framework: Dynamic Logic

Dynamic Logic (DL) is a modal logic proposed by David Harel [Har79] in the late 70+IBk-s as a formalism for reasoning about programs that check and modify a given environment. The main goal was to describe in a formal way the effect of programs execution as a first step for further formal reasoning about them. Although Harel+IBk-s work on DL is very extensive, we are describing next, as an introductory summary, those dynamic formulae that are relevant to our purposes.

A Dynamic formula is expressed as +8GY- +8Fs- p+8F0- +8Hk- where +8GY- and +8Hk- are first order formulae, p represents a program (in a wide sense) and +8Fs- _+8F0- is the necessity operator. The intuitive meaning of such a formula can be stated as follows: "if +8GY- holds , then after execution of the program p, +8Hk- must hold".

Dynamic formulae, as opposed to first order formulae, have a special singularity. The latter are interpreted in a single interpretative structure, whereas the former need several ones. This important difference can be understood by taking into account that dynamic formulae are used to describe an evolving process, i.e., a certain sequence of state changes. Thus, a formal dynamic theory is interpreted by using a Kripke structure K=(+8Fc- , +8Hc- 0, +8HI-) where:

* +8Fc- is a set of possible worlds, each one of them defined as a first order interpretative structure,

* +8Hc- 0 is the initial world, and

* +8HI- (p) +8M0- +8Fc- +8LQ- +8Fc- , being p a program, is an accessibility relation among worlds.

The figure shows a simplified graphical representation of such a structure. This type of structure needs to be extended to capture the notion of history or trace to make it suitable for interpreting processes. This extension will result in a new interpretative structure called Dynamic Kripke Structure.

Definition 3

Given a Kripke structure K=(+8FfwLPAg8Hc-+8DA-+8CzwIPBy-), a trace over K is defined inductively as follows:
* +8Gw- is a trace over K (the empty trace)
* if T is a trace over K, e+8M4- X and +8HfwzvAg8Fc-+8CzwIA-then T+8Po- (e,w) is a trace over K
* only are traces those defined by i), ii).

Definition 4

A Dynamic Kripke Structure is a pair Kd = (K,T), where K is a traditional Kripke structure and T is a trace over K.

3.2.2 Semantics of P-processes

P-processes are used, as described earlier, to express activity that occurred in the past of a given object. Therefore, their semantics will be directly defined in terms of a DKS since this interpretative structure keeps track of the actual life (trace) of every object in a given system.

Definition 5

A DKS Kd = (K,Tk), where K=(+8FfwLPAg8Hc-+8DA-+8CzwIPBy-), is model of a P-process p if and only if:

* p = +8Gw- , or

* p = q & pn , and Tk = T+8Nc- (e,+8Hc-) , with +8Hc- +8M4- +8Fc- such that either

* pn = e, or

* pn = (e1 or e2 or ... or em) and +8CQ- i+8M4- {1, ..., m}: ei = e,

and Kd+IBk- = (K,T) is model of q, or

* p = q & check(+8Go-), such that either

* Tk = T+8Nc- (e,+8Hc-) / +8CQ- (+8HfwLPB3-+8DE-) +8M4- +8HI- (e) such that+8CDwdw-1+JV4- +8Go- , or

* Tk = +8Gw- / +8Hc-0+JV4- +8Go-

and Kd is model of q.

3.2.3 Semantics of F-processes

As opposed to P-processes, F-processes describe future behavior rather than a history; DL already provides a language with a well-defined semantics in order to reason about future evolution, thus we make use of it to give a semantics to F-processes that incorporates the notion of obligation. Taking as basis the fact that any operator of a process algebra can be defined in terms of those of the BPA [Wie93] we just give the formalization for the sequence and choice operators.

3.2.3.1 Sequential composition of events

Let us suppose we have an object o whose type has a process term that includes a subterm like the shown in the following figure:

where ei and ej are events of X. Let v1, ..., v3 denote the states that o reaches after the execution of the different events that compose the subprocess +IBQ-as already mentioned, those states will be characterized by a set of 1st-order formulae. In terms of them, the above process has the following semantics:
* v1 [ei] v2
* v2[+8Ng- ei] false
* v2[ei] v3

By +8Ng- ei we denote in ii) the non occurrence of ei; there has been some controversy in the literature about the action negation operator; some proposals [Geh92] have a complement operator, and other [Cha93] explicitly omit a not operator from the event definition language. We use the notation +8Ng-a as a syntactic facility to express the occurrence of any action different from a.

3.2.3.2 Choice among events

In this case the situation is slightly different, though the treatment is similar to the previous one; for the subprocess

we distinguish between v2 and v3 because the execution of each action in the choice may lead the object to a different state; let us suppose that v2, v3 represent the states reached by the object after the execution of the events ei, ej respectively; thus the semantics in terms of dynamic formulae is given by:

* v1[+8Ng- ei]v3
* v1[ei] v2
* v1[+8Ng- ej] v2
* v1[ej]v3

3.2.3.3 Semantics of the check process

The check process as a component of a F-process forces the validation of a certain first order formula +8GY- in the current state of an object. The semantics for this process can be also expressed in terms of dynamic formulae:

+8Ng- +8GY- [check(+8GY-)] false

3.2.4 Dimensions of behavior in terms of PF-rules

So far, we have defined a general formal framework to be used in the characterization of OASIS behavioral components; next we show them together with their formalization through PF-rules.

3.2.4.1Reactive behavior

3.2.4.1.1 ES reactions

When an event e occurs in the life of an object, the object behaves following a predefined pattern:

* Evaluation of the event precondition +8GY- in the object+IBk-s state

* Change of the object+IBk-s state according to the valuation formula +8Hk- [e]+8GQ- .

* Integrity constraints checking: integrity constraints +8Go- are checked in the new state reached by the object.

If either the precondition is not true or the final state of the object does not satisfy the constraints, the event is rejected and there is no change of state.

The following PF-rule defines this type of behavior:

[check(+8GbwKfAg8CbwIA-check+8CjwefAg-)& e] <check(+8GQ-) & check(+8Go-)>

3.2.4.1.2 EA reactions

EA reactions are introduced in OASIS to cope with what other authors call causal dependency between events, that is, the occurrence of an action a in the life of an object after the previous occurrence of another event e, regardless the state of the object. This behavior can be expressed in terms of PF-rules as

[e]<a>

3.2.4.1.3 CA reactions

CA reactions are CA rules where the condition is a P-process, and the action that must be fired if the condition holds (in other words, if the trigger is activated) is a F-process. Each time a change of state occurs on the life of an object, the state-driven triggers are checked and the associated actions are executed for those whose conditions hold. A CA trigger with condition +8GY- and action a can be written in terms of PF-rules as:

[check(+8GY-)]<a>

In all previous cases we talk about reactivity rather than activity because this behavior is not truly spontaneous, yet it is dependent of the occurrence of an event.

Triggers extend ES reactions in the following way: after integrity checking, a trigger monitoring is done in order to check which rules are activated in the new state. A policy to resolve conflicts when more than one rule is activated must be defined.

3.2.4.2 Active behavior

Active behavior is introduced in our model as a way of characterizing patterns of objects+IBk- lives. Thus, it is not a behavior that is triggered by a particular change of state at any point of an object+IBk-s life but by the fact that it has been created. An object in this case performs a certain active behavior by its own and not as a reaction to external stimuli. A pattern of life p for a given object can be expressed as a PF-rule as follows:

[birth]<p>

where birth denotes the creation of the object and p is the F-process that corresponds to the component P of the OASIS type as described in section 2.

4. Conclusions and future work

In this paper we have presented a model of activity to be used in the requirements engineering phase of the software lifecycle supporting the claim that (re)active behavior is just another property of real entities and should not be modeled taking as primitives events in a particular DBMS. This independence allows implementations of information systems over different target ADBMS.

The behavioral OASIS model distinguishes between reactive and active behavior of objects. We have shown that all of them, though different in essence, can be syntactically expressed by using Past Future rules in a unified way. Complex processes consisting of events and conditions over states can be described in order to represent activity in, both, the future and the past life of objects. We have also defined a common declarative semantics in terms of Dynamic Kripke structures and Dynamic Logic formulae, setting the basis for further research on operational semantics to reach implementations of ADBMS consisting of sets of PF rules.

The concepts an ideas presented deal with a very specific area of interest in Software Engineering: a syntactically and semantically unified model that extends the capabilities of an existing model (OASIS) to enhance its active capabilities. However, there is an essential question that must not be avoided: is this an exhaustive solution to the problem of Requirements Analysis and Specification of Active Systems? The answer to this question is, in our opinion, no. A complete solution must deal, on the one hand, with the model-oriented problems and challenges presented in this paper and, in the other hand, with the methodological aspects that support cognitive processes of identification, analysis, and specification of active dimensions in Information Systems. These clearly needed methodological guidelines are currently under research and an exhaustive overview of the existing philosophic and logic fertile literature about the problem of actions, which is out of the scope of this paper, has been performed [Jae98]. As a result of this preliminary study we find particularly interesting the work of Rescher [Res66] in which a thorough catalogue of key generic descriptive elements of actions is provided. His categorization has several strong features that make it suitable for developing a methodology for Requirements Analysis: it is simple, in-depth, and abstract (i.e., no references to database oriented concepts are present). We are currently developing a methodology based on Rescher+IBk-s categories which will complement our model-oriented concepts in order to obtain a complete environment for Requirements Analysis and Specification of Active Systems. Some additional work must be done as well in order to check the suitability of a transformational approach to implementations over different data models. Finally, though a prototype implementing the OASIS model exists, a deeper study of the appropriate architecture for an OASIS-based ADBMS is in project.

Acknowledgements

The work of Javier Jaén is supported by the Fulbright Commission under program "Spain-USA 2000."

References

[Can95] Canós, J. H., Penadés, M.C., Ramos, I. and Pastor, O., A knowledge-base architecture for object societies, Proceedings of the DEXA-95 Workshop, OMNIPRESS, 1995.

[Can96] Canós, J.H., OASIS: a Unique Language for Object-Oriented Databases, Ph.D. Thesis, Universitat Politècnica de València, 1996 (in Spanish).

[Cha93] Chakravarthy, S., and Mishra, D., Snoop: An Expressive Event Specification Language for Active Databases, Tech. Report UF-CIS-TR-93-007, Deptartment of Computer and Information Sciences, University of Florida, 1993.

[Dit95] Dittrich, K., Gatziu, S. and Geppert, A (ed.), The Active Database Management System Manifesto: A Rulebase of ADBMS Features, in T. Sellis (ed.), Proc. of the 2nd workshop on Rules in Databases (RIDS), Athens, Greece, LNCS, Springer-Verlag, 1995.

[Geh92] Gehani, N.H., Jagadish, H.V. and Shmueli, O., Event Specification in an Object-Oriented Database, Proceedings of the International Conference on Management of Data, pp. 81-90, san Diego, California, 1992.

[Har79] Harel, D., First-Order Dynamic Logic, Springer-Verlag, 1979.

[Har92] Hartmann, T. , et al., Aggregation in a Behavior Oriented Object Model, in O. Lehrmann Madsen (ed.), Proc. European Conference on Object-Oriented Programming (ECOOP+IBk-92), pp. 57-77, LNCS 615, Springer-Verlag, 1992.

[Jae98] Jaén, F.J., Canós, J.H. and Ramos, I., On the Nature of Active Systems: Limitations and Formalization, Technical Report, DSIC, Polytechnic University of València, 1998.

[Pas92] Pastor,O., et al., OASIS: An Object Oriented Specification Language, Proceedings of the CAiSE-92 Conference, LNCS 593, Springer-Verlag, 1992, pp.348-363.

[Ram93] Ramos, I. et. al., Objects as observable processes, Proceedings of the 4th International Workshop on the Deductive Approach to Information Systems and Databases, Tech. Report, DLSI, Universitat Politècnica de Catalunya, 1993.

[Res66] Rescher N. The Logic of Decision and Action. University of Pittsburgh Press, 1966.

[Wie93] Wieringa, R.J. et al., Actors, Actions and Initiative in Normative System Specification, Annals of Mathematics and Artificial Intelligence, 7:289-346, 1993.