EUROSIM 2007

9-13 September, 2007, Ljubljana, SLOVENIA

Proceedings of the
6th EUROSIM Congress
on Modelling and Simulation

ISBN-13: 978-3-901608-32-2
ISBN-10: 3-901608-32-X
Title Abbrev.: Proc. EUROSIM 2007

Publisher:
ARGESIM - ARGE Simulation News
Div. Mathematical Modelling and Simulation
Vienna University of Technology, Vienna, Austria

COMPACT disc DATA STORAGE

Bibliographic information published by **Die Deutsche Bibliothek**:
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie
Detailed bibliographic data are available in the Internet at http://www.ddb.de

ISBN 978-3-901608-32-2

SLOSIM
Slovenian Society for
Simulation and Modelling

EUROSIM
Federation of European
Simulation Societies

University *of Ljubljana*
Faculty *of Electrical Engineering*

EUROSIM 2007

9-13 September, 2007, Ljubljana, SLOVENIA

Proceedings of the
6th EUROSIM Congress
on Modelling and Simulation

Edited by
Borut Zupančič
Rihard Karba
Sašo Blažič

Vol.2: Full papers

A PETRI NET BASED SIMULATOR FOR ACTIVE DATABASE SYSTEMS

Joselito Medina-Marín[1], Marco A. Montufar-Benítez[1], Aurora Pérez Rojas[1], Oscar Montaño-Arango[1], José Ramón Corona-Armenta[1], Jaime Garnica-González[1]

[1]Autonomous University of Hidalgo State,
Advanced Research in Industrial Engineering Center,
Carr. Pachuca-Tulancingo Km. 4.5, Col. Carboneras,
Pachuca, Hidalgo, Mexico

jmedina@uaeh.edu.mx (Joselito Medina-Marín)

Abstract

Active database systems were introduced to extend the database functionality. As well as a repository of data, active database can detect the occurrence of events in a database system and react automatically to that event occurrence and execute certain actions either inside or outside the database. This behavior is specified by means of ECA (event-condition-action) rules, i.e., when an event has occurred, if the condition is evaluated to true, then an action is executed. The development of a set of ECA rules involve the knowledge of the database structure and the relationships that can exist among the ECA rules, which may produce an inconsistent state in the database. Therefore, it is so important to verify a rule set before its implementation in the active database, and one method to determine if a rule set will produce consistent states of the database is through the simulation of ECA rule firing. In this paper a simulator for active databases, named ECAPNSim, is described. ECAPNSim uses the definition of ECA rules like a structure of an extended Petri net model, the Conditional Colored Petri Net (CCPN). Conditional Colored Petri Net definition involves the knowledge and execution model, which describe the features that an active database system must have. Furthermore, in order to simulate the occurrence of database events, ECAPNSim has been enhanced with the addition of distribution functions for each place that denote events of the ECA rule set.

Keywords: Petri net, Active Database, ECA rules, Simulation.

Presenting Author's biography

Joselito Medina-Marín. He received the M.S. and Ph.D. degrees in electrical engineering from the Research and Advanced Studies Center of the National Polytechnic Institute at Mexico, in 2002 and 2005, respectively.
He is presently a Professor of the Advanced Research in Industrial Engineering Center at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México. His current research interests include Petri net theory and its applications, active databases, simulation, and programming languages.

1 Introduction

Traditional databases (DB) were developed to store a huge amount of information. In this DB type the information only is accessed by insert, delete, update and query algorithms, which were previously programmed in a Data Manipulation Language (DML) by the DB administrator. The set of all this data manipulation programs is the Database Management System (DBMS). However, the execution of those programs is performed only by the request of either a DB user or the DB administrator.

Nevertheless, there are systems that cannot be implemented by using a traditional DB approach. Such systems are those where is well known that if certain events occur in the DB and if the DB state satisfies certain conditions, then an action or procedure is performed in the DB. Therefore, it is necessary to use an approach where a DB could have the ability to react automatically when an event occurs either inside or outside DB environment, after this, it can verify the DB state to evaluate conditions, and if condition is evaluated to true it can execute procedures that modify the DB state. In order to provide of active behavior to traditional DB, Active Databases (ADBs) were introduced. If a human being takes charge to detect the event occurrences, verify conditions, and execute procedures instead an ADB system, then the system may not work well. Thus, it is very important to add enough information to DB about the active behavior and convert a traditional DB into an Active one.

Active behavior of a DB can be defined through a base of active rules, which has the specification of events that will be detected, conditions that will be evaluated, and actions or procedures that will be performed in the DB. The model most widely used is the event-condition-action rule (ECA rule) model, whose general form is as follows [1]:

on event e1

if condition c1

then action a1

ECA rule model works in the following way: when an event e1 that modifies the current DB state occurs, if condition c1 is evaluated to true against DB state, then either an action a1 is executed inside DB or a message is sent outside DB.

An event e1, which can trigger to an ECA rule, can be of two types: primitive event or composite event [2]. A primitive event is generated by the execution of an operation over the DB information (insert, delete, update, or select), a DB transaction, a clock event (which can be absolute, relative, or periodic), or the occurrence of a DB external event. On the other hand, composite events (disjunction, conjunction, sequence, closure, times, negation, last, simultaneous, and any) are formed by the occurrence of a combination of primitive and/or composite events.

Composite events increase the complexity of a base of active rules because composite events are represented by complex structures, which need to be evaluated when a composite event is raised. In the same way that a composite event increases the complexity of a base of active rules, relationships between ECA rules increase the complexity of a base of active rules.

Furthermore, active rules must be validated before its implementation into a real active database system, in order to known its behavior and to verify the presence of situations that may produce an inconsistent state in the database system.

This verification can be performed trough the simulation of the active rules. In this paper an ECA rule simulator is presented, which uses a Petri net model, named Conditional Colored Petri Net (CCPN), to depict ECA rules as a Petri net structure, and with the token game animation the event occurrence and rule triggering are analyzed in order to detect active database problems such as No termination and confluence [2].

2 Related work

There are several research studies about active databases and the development of ECA rules. Relational systems, such as starburst [3], Postgres [4], Ariel [5], SYBASE [6], INFORMIX [7], ORACLE [8], among others, provide an active functionality based on triggers, but they cannot handle composite events at all.

Triggers only supports the composite event disjunction, and structure primitive events that are defined over a table, moreover, in the action part of triggers cannot be executed another trigger.

On the other hand, Object Oriented DB systems (such as HiPAC [9], EXACT [2], NAOS [10], Chimera [11], Ode [12], Samos [13]) provide more elements of active systems, like the composite event handling. Nevertheless, because of the different structures and classes used to develop Object Oriented DB systems, there is not a standard model to define ECA rules in these systems.

Few researches have adopted Petri nets as ECA rule specification language [13], [14] [17]. In [17], the authors proposed an Action Rule Flow Petri Net (ARFPN) model, and a workflow management system was illustrated to verify their ARFPN model. However, their model has much redundant structure because of using many BEGIN OFs, END OFs to describe events, conditions and actions. SAMOS is a successful ADB system, Petri nets is partially used for composite event detection and termination analysis. But, the framework is not Petri-net-based.

Colored Petri Nets (CPN) is a high-level Petri nets which integrates the strength of Petri nets with the strength of programming languages. Petri nets provide the primitives for the description of the synchronization of concurrent processes, while programming languages provide the primitives for the definition of data types and the manipulation of their data values [18]. So it is more suitable for active database than ordinary Petri nets since it can manipulate data values. By using CPN one can not only revealing the interrelation between ECA rules but also capture the operational semantics. For these reasons, CPN is very suitable for modeling and simulation of active rules. References [17] adopted CPN as rule specification language. However, there exists much redundant PN structure for using "begin of", "end of" events, conditions and actions repeatedly. So, Their CPN model is very large even for a small rule set. Therefore, the complexity of CPN management increases. In SAMOS a SAMOS Petri Nets (S-PN) was proposed for modeling and detection of composite events. S-PN is also CPN-like where a different perspective for colors was taken. Colors in S-PN are token types, and one token type is needed for each kind of primitive event.

3 Conditional Colored Petri Net definitions

There are several proposals to support reactive behaviors and mechanisms inside a DBS, which is best known as an ADBS. Nevertheless, these proposals are designed for particular systems, and they cannot be migrated to any other system, moreover, there is not a formal ADBS proposal.

In this paper, a general model to develop ECA rules in an ADBS is proposed, based in PN theory, which can be used as an independent engine in any DBS. An ADBS must offer both a knowledge model and an execution model. Knowledge model specifies the elements of the ECA rule, i.e., the event, condition, and action part. On the other hand, execution model describes the way in that the ECA rule set will be executed.

In knowledge model, each ECA rule element is converted into a CCPN element. The event, which activates the ECA rule, is converted in a CCPN structure that is able to perform the event detection. A Primitive event is depicted by a CCPN place, but if the event rule is composite, then the corresponding CCPN structure is generated. Both types of events finish in a place, which will be used as an input place for a transition.

A CCPN transition holds the next element of an ECA rule, the conditional part. It verifies if there are tokens in its input place and evaluates the conditional part of the ECA rule that is holding. Unlike traditional PN transitions, CCPN transitions have the ability to evaluate boolean expressions.

Finally, the ECA rule element action. When action part is executed in a DBS, it modifies the DB state. This can be viewed as an event that modifies the DB state. Events are represented as CCPN places, thus action part is represented by a place too. The difference between places for events and places for actions is that places for events are input places to transitions, and places for actions are output places from transitions.

CCPN execution model is based in the transition firing rule of PN theory. It provides mechanisms to create tokens with information, or color, about events that are occurring inside the DB. New tokens are placed in the corresponding places for those events. This is the way in that an ECA rule set is processed and both composite and primitive events are detected.

By using Colored Petri Nets (CPN) is possible to depict ECA rules, but only those that have primitive events. ECA rules with composite events cannot be represented efficiently with CPN.

Definition Conditional Colored Petri Net (CCPN) [19] is a Petri net extension, which inherits attributes, and transition firing rule from classical PN [14] [15] [16]. Furthermore, CCPN takes concepts from the CPN, such as data type definition, color (values) assignation to tokens, and data type assignation to places.

In the CPN case, data type assignation is performed for all the places of CPN, on the other hand, in the CCPN case, data type assignation for places is not general, because the CCPN handles a kind of place (virtual place) with the ability to hold different types of tokens.

In order to evaluate conditional part of ECA rule stored inside a CCPN transition, a function is defined to do this task. Evaluation function analyzes the boolean expression and match it with the DB state to determine its boolean value.

Some composite events needs to verify a time interval, hence CCPN provides a function that assigns time intervals to a CCPN transition, which will be the responsible to verify if events are occurring inside time interval defined, likewise the evaluation of ECA rule condition is done. These types of transition are named composite transition.

Each event occurs in a point of time, thus, CCPN provides a functions that assigns a time stamp to every token created. Time stamp value is the time instant in which the event has occurred. It is useful to verify if an event occurred inside a time interval or to detect composite events such as sequence and simultaneous.

Finally, every time that an event occurs, a token must be created. CCPN has a function to initialize tokens, in other words, when an event occurs in DB, a new token is created by CCPN and its attributes are initialized to the corresponding event values. The new token is put in the place that represents to detected event.

CCPN is an extension of PN that uses CPN concepts [18]. In order to save event information in tokens and to create new tokens with data about the action part of the ECA rule, CCPN uses the concept of "color" taken from CPN. The values stored in tokens are used to evaluate the conditional part of the rule stored in the transition of CCPN. CCPN uses the multi-set concept from CPN, because a CCPN place may have several events at the same time. Unlike CPN, CCPN evaluates conditions inside transitions; meanwhile CPN evaluates conditions in its arcs.

4 ECAPNSim

ECAPNSim is a graphical interface developed as a part of this research, in order to convert automatically ECA rule sets into CCPN structures. Furthermore, ECAPNSim can provide of active functionality to relational databases by establishing communication via ODBC-JDBC drivers. ECAPNSim detects events in the DB, it performs the evaluation of condition, stored in transitions $t \in T_{rule}$, and it executes actions inside the DB, according to the ECA rule set represented as a CCPN.

ECAPNSim has two modalities: in the first one, ECAPNSim works as a PN simulator, where the simulation of the ECA rule set behavior is performed; and in the second one, ECAPNSim works as the engine of an active database, in other words, ECAPNSim is placed as an upper layer over a DB system, ECAPNSim "listen" the events that modify the DB state and if there is any event that is in the CCPN as a place, then ECAPNSim takes information about the event and create the token about the event, after that, ECAPNSim places the new token in its corresponding place and starts the token game animation (ECA rule firing).

4.1 Incorporation of distribution functions

ECAPNSim was enhanced with the addition of distribution functions, which are useful to simulate and to analyze the event occurrence in an active rule base.

Distribution functions which are able in ECAPNSim are beta, binomial, Cauchy, chi square, exponential, gamma, geometric, uniform, and weibull, among others. The use of this set of functions depends on the active rule base that will be simulated.

Each place in the CCPN has the property for the definition of a distribution function, according to the frequency of the event occurrence. The values for the functions can be determined by a statistical analysis of the data about the real occurrences of the events that fire ECA rules.

5 Conclusion

Currently there are database management systems that support ECA rule definition by the use of "triggers", however "triggers" has several restrictions that limits the power that an active database must offer.

On the other side, there are research prototypes that support ECA rule definition, too; and they are more powerful because composite events such as conjunction, disjunction, etc., can de defined. Nevertheless, like database management systems, ECA rule definition is performed in the syntax of every active database.

ECAPNSim is an interface that generates a CCPN from an ECA rule definition typed in the on-if-then form. It carries out the simulation of the CCPN behavior according to the event occurrence in a random way, which depends on the distribution function assigned.

ECAPNSim has been improved with the addition of distribution functions in each place that denote an event occurrence.

6 References

[1] A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Third Edition, McGraw-Hill, 1999.

[2] N. W. Paton, O. Diaz, Active Database Systems, *ACM Computing Surveys*, Vol. 31, No. 1, pp. 64-103, 1999.

[3] J. Widom, The Starburst Active Database Rule System, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 4, August 1996.

[4] M. Stonebraker, G. Kemmintz, The POSTGRES Next-Generation Database Management System, *Communications of the ACM*, Vol. 34, No. 10, October 1991.

[5] E.N. Hanson, The Design and Implementación of the Ariel Active Database Rule System, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 1, 1996.

[6] D. McGoveran, C.J. Date, A guide to SYBASE and SQL Server : a user's guide to the SYBASE product, Sybase, Inc, 1992.

[7] T. Lacy-Thompson, INFORMIX-SQL, A tutorial and reference, ISBN-0-13-465121-9, Ed. Prentice Hall, 1990.

[8] C.J. Hursh, J.L. Hursch, Oracle SQL Developer's Guide, ISBN-0-8306-2529-1, Ed. McGraw-Hill, 1991.

[9] U. Dayal, B. Blaustein, A. Buchmann, U. Chakravarthy, M. Hsu, R. Ledin, D. Mc-Carthy, A. Rosenthal, S. Sarin, M.J. Cary, M. Livny and

R. Jauhari, The HiPAC Project: combining active database and timing constraints, SIGMOD

[10]C. Collet, T. Coupaye, Composite Events in NAOS. *In 7th International Conference and Workshop on Database and Expert Systems Applications*. (DEXA'96). LNCS 1134, pages 244—253, Zurich, Switzerland. 1996.

[11]S. Ceri, P. Fraternali, S. Paraboschi, L. Tanca, Active Rule MAnagement in Chimera, *Active Database Systems: Triggers and Rules for Advanced Database Processing*, Ed. Jennifer Widom and Stefano Ceri, pages 151-176. 1996.

[12]N. Gehani, H.V. Jagadish, Active Database Facilities in Ode, *Active Database Systems: Triggers and Rules for Advanced Database Processing*, Ed. Jennifer Widom and Stefano Ceri. 1996, pages 207-232.

[13]E. Gatziu, K.R. Ditrich, SAMOS, *Active Rules in Database Systems*, Norman W. Paton, Editor. 1999, pp. 233-248.

[14]X. Li, J. Medina-Marín, and S.V. Chapa, A Structural Model of ECA Rules in Active Database, *Mexican International Conference on Artificial Intelligence (MICAI'02)*, Mérida, Yucatan, México, April 22-26, 2002

[15]X. Li, J. Medina Marín, Composite Event Specification in Active Database Systems: A Petri Net Approach, *IEEE International Conference on System, Man, and Cybernetics*, The Hague, The Netherlands, Oct, 2004.

[16]J. Medina Marín, X. Li, An Active rule base Simulator based on Petri Nets, *The Third International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems MSVVEIS-2005*, Miami, USA., May 24, 2005.

[17]M. Schlesinger, G. Lörincze, Rule modeling and simulation in ALFRED, *the 3rd International workshop on Rules in Database (RIDS'97)* (or LNCS 1312), Skövde, Sweden, June, pp. 83-99, 1997

[18]K. Jensen, An Introduction to the Theoretical Aspects of Colored Petri Nets. *Lecture Notes in Computer Science: A Decade of Concurrency*, vol. 803, edited by J. W. de Bakker, W.-P. de Roever, G. Rozenberg , Springer-Verlag, pp. 230-272. 1994.

[19]J. Medina Marín, Desarrollo de reglas ECA, un enfoque de red de Petri, Ph. D. Dissertation, CINVESTAV-IPN, México, 2005.