MANAGING COMPLEX INFORMATION IN REACTIVE APPLICATIONS USING AN ACTIVE TEMPORAL XML DATABASE APPROACH

Essam Mansour, Kudakwashe Dube, Bing Wu
School of Computing, Dublin Institute of Technology, Kevin Street, Dublin 8, Ireland
<firstname>.<surname>@dit.ie

Keywords: information management, XML, ECA rule paradigm, active database, and temporal database.

Abstract: Tasks, such as those in patient care practice, require constant monitoring of a dynamic context and environment based on best practices in the form of predefined experience-based information or knowledge. These applications could take the form of reactive applications. The problem of incorporating best practices into routines used in such tasks requires advanced approaches and methods for comprehensively managing complex information. This paper presents a generic and unified framework for Complex Information Management (CIM) in domains where best practices need to be incorporated into day-to-day work. The CIM framework is supported by a high level declarative language, called AIM. The approach adopted here uses the combined application of the event-condition-action (ECA) rule paradigm, a temporal mechanism, advanced DBMS features and XML technologies. Furthermore, the paper also presents the conceptual architecture for the complex information management system, AIMS. The main contribution of CIM framework and approach lays in managing the complex information by introducing multiple management planes under a unified framework.

1 INTRODUCTION

Reactive applications have the ability to process events of interest to domain entities. They can respond to changing situations by issuing alerts, reminders, requests, and/or observations to users. The information and knowledge used in guiding the reactive aspect of applications usually exist in generic forms as best practice, experience, policies and operational rules that guide decisions and actions. This information and knowledge require formalisation to be incorporated into reactive applications. Later, they need to be customized and instantiated to suit the condition or situation in the domain. The result of the customization and instantiation process is what we refer to, in this work, as *complex information.* This will be made up of situation-specific, dynamic information and knowledge that represent the best practice in the domain. It further forms the basis of the reactive behaviour that monitors occurrences of relevant situations in the domain. Examples of this type of *complex information* include: the medical patient plan in the intensive care applications; the customer order in securities trading; and the customer banking alerts scenario. The medical patient plan monitors specific clinical data items and responds to the changes of these data items with respect to specific patient according to customized reactive behaviour based on the clinical guidelines. The security order deals with particular stock items, to them a set of constrains and conditions are applied. These constrains are subject to modification according to the customer's experience as response to the stock items changes, until the security order is executed or cancelled.

A major challenge in reactive applications is to provide a unified framework in which the *complex information* can be managed as one body in such a way that the formalisation, specification, execution, manipulation and query dimensions are explicitly provided as first class elements under one roof within the framework.

In the area of active database, managing reactive aspects of applications is addressed at the level of rules and triggers, such as in (Umeshwar et al. 1988; Widom and Ceri 1996; Paton and Diaz 1999; Bonifati et al. 2002), not at the level of a unified body of *complex information.* Consider, as an example, the users of securities trading and intensive care applications demand computerized support for managing dynamic orders and medical patient plans, respectively, instead of individual rules and triggers. Existing support for reactive applications suffer from lack of support for queries and manipulation. Furthermore, problems of unexpected interactions among the individual rules become more likely with the growing of the rule base (Umeshwar et al. 1988).

This paper presents on-going research (Dube and Wu 2001; Wu and Dube 2001; Wu and Dube 2001; Dube et al. 2002; Dube et al. 2002; Dube 2004; Dube et al. 2005; Mansour et al. 2006). The first stage of the research was aimed at providing a generic Framework with a declarative language PLAN (Wu and Dube 2001) for managing clinical reactive applications based on Clinical Practice Guidelines (CPGs). Also a prototype system TOPS (Wu and Dube 2001; Dube 2004) was developed to implement the PLAN framework and language. This second stage focuses on investigating into a formal *complex information* management model and approach that: 1) is based on XML, 2) takes into account temporal issues, 3) continue to exploit the event-driven approach adopted in the previous stage; and 3) allows advanced queries, replays of past history and information mining.

2 FRAMEWORK AND APPRAOCH

This section presents the framework for *complex information* management and the adopted approach to computerising the *complex information.*

2.1 The CIM Framework

The *Complex Information Management* (CIM) framework supports the management of the *complex information* in reactive applications. As illustrated in Figure 1, the CIM framework contains three planes: *specification* plane; *instantiation and customization* plane and; *management* plane with the DBMS, XML technologies and inter-plane communication as the integrating factors. The DBMS provides support for the ECA rule paradigm and the temporal mechanism.

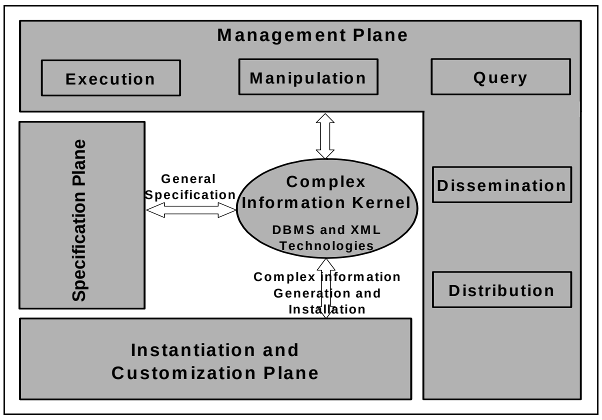
Management Plane
Execution
Manipulation
Query
Specification Plane
General Specification
Complex Information Kernel
DBMS and XML Technologies
Dissemination
Complex Information Generation and Installation
Distribution
Instantiation and Customization Plane

Figure 1: the Complex Information Management (CIM) framework

The *complex information* management process is fitted into the three planes of the framework, as follows:

1) *Specification plane:* Best practice in domains for reactive applications is formalised and translated into formal specifications and held permanently in the database where they can be managed easily.
2) *Instantiation and customization plane***:** The stored generic specifications are used to create the *Complex Information* **(***CI***).**
3) *Management plane*: The installed *CI* is managed. The plane provides support to execute, manipulate, query, and disseminate the *CI*. The distributed management for the *CI* supports the mobility of the domain entities as well as remote access to the information. An engine based on the active database is utilized to execute the *CI*. The *CI* is also subject to the same manipulation operations, as the domain information. The manipulation operations are included in the behaviour of *CI*, or issued by the domain users. The manipulation operations facilitate the propagation of the changes from the generic specification to the *CI*. The *CI* also is subject to the same queries, as the domain information, plus special query support, such as replay *CI*. The replay functionality provides ability to review the evolution of the *CI*.

An XML-based language, called AIM, is developed to support the functionalities of the three planes. AIM language consists of a model for the *complex information* and two sub-languages, AIM-

SL and AIM-QL. AIM-SL and AIM-QL support the functionalities of the *specification plane* and the *management plane*, respectively. The model of *CI* provides support to the *instantiation and customization plane*. The approach adopted in developing AIM language is presented in the next subsection.

2.2 Active Temporal XML Approach

The Active Temporal XML approach utilizes the combined application of the Event-Condition-Action (ECA) rule paradigm, a temporal mechanism, advanced DBMS features and XML technologies. A map of the domains involved in this approach is shown in Figure 2. Based on this approach, AIM language is developed such that:

1) It is an XML based language and enjoys the general benefits of XML.
2) It has an active XML-based specification component language, called AIM-SL, which utilizes the ECA rule paradigm to formalise and specify the application best practice. The design of AIM-SL is based on modularised XML Schemas. This modularization provides flexibility in modifying or enriching the AIM-SL language to suit several application domains.
3) The *complex information* model is developed as an active temporal XML document, which includes the history of the *CI* and the rules that represent the behaviour part of *CI*.
4) It has a high level XQuery-based component language, AIM-QL, that provides support to:
    a) Manipulate both the AIM-SL speciation and the *CI* documents;
    b) Query and replay the *CI*;
    c) Provides temporal extensions to the XQuery language to support the replay functionality.

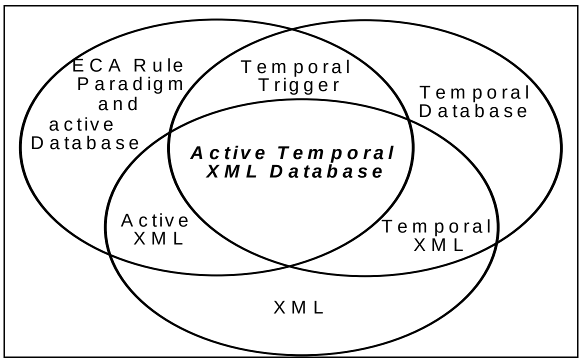
ECA Rule Paradigm and active Database
Temporal Trigger
Temporal Database
Active Temporal XML Database
Active XML
Temporal XML
XML

Figure 2: the active temporal XML database approach

3 CONCEPTUAL ARCHITECTURE

This section presents the conceptual architecture of the AIM System (AIMS), whose development is currently on-going. AIMS utilises the AIM language to manage the complex information according to the CIM framework.

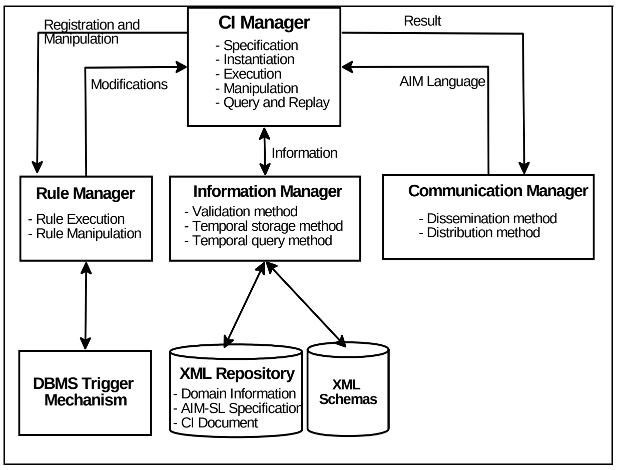

Figure 3: conceptual architecture of AIMS

A conceptual architecture of AIMS is presented in Figure 3. The *Complex Information (CI) manager* provides the high level management for the *complex information*. *CI manager* provides support to specify the application best practice, generate, execute, manipulate, and query *CI* documents. The *rule manager* provides for the execution of *CI* rules, their manipulation and extension of the active mechanism to support temporal rules. The *information manager* extends an XML database system to provide temporal support and utilizes an XML DBMS to validate and store the AIM-SL specification and the *CI* documents. The *communication manager* supports the remote access and distributed management to the AIM-SL specification and *CI* documents. The AIM language is implemented in the *CI* manager. AIMS is being implemented by using DB2, java, and XML technologies, such as XQuery and Web services.

AIMS avoids the unexpected interactions that most likely appear with the growing of the rule base, because:

1) The rules are modularised and joined to a specific domain entity instead of specific relation, such as table in the relational database, and
2) The rule manager is able to remove a set of rules according to its objective or scope.

4 CONCLUSION

This paper has presented the Complex Information Management (CIM) framework and approach, which consists of three planes for specifying, customizing and managing the *complex information*. The AIM language that is developed to support the functionalities of the CIM framework planes has been briefly outlined. The framework also includes a *complex information* model that enables easy performance of information management functions. The paper has also presented a conceptual architecture for the AIMS system, which utilizes the AIM language in managing the *complex information* by following the CIM framework. The development of the prototype system, AIMS, is currently in progress. The CIM framework and approach are unique in:

1) Computerizing the *complex information* as one distinct entity that is easy to be specified, executed, manipulated, queried and disseminated using one language, AIM.
2) Allowing the *complex information* to be subject to the same manipulation operations and queries, as the domain information, plus special operations and query functionality.
3) Utilizing the generally available highly optimized and easily maintained technologies, such as XML technologies and ECA rule paradigm as incorporated in the modern DBMS, to provide a tool that assists domain experts and users in managing the *complex information*.

REFERENCES

Bonifati, Angela, Daniele Braga, Alessandro Campi, et al. (2002). Active XQuery. Proceedings of the 19th International Conference on Data Engineering ICDE, San Jose (California).

Dube, Kudakwashe (2004). A Generic Approach to Supporting the Management of Computerised Clinical Guidelines and Protocols, Dublin Institute of Technology (DIT).

Dube, Kudakwashe, Essam Mansour and Bing Wu (2005). Supporting Collaboration and Information Sharing in Computer-Based Clinical Guideline Management. 18th IEEE Symposium on Computer-Based Medical Systems (CBMS 2005), Dublin, Ireland, IEEE Computer Society.

Dube, Kudakwashe, Bing Wu and Jane Grimson (2002). Using ECA Rules in Database Systems to Support Clinical Protocols. 13th International Conference on Database and Expert Systems (DEXA 2002).

Dube, Kudakwashe, Bing Wu and Jane Grimson (2002). Framework and Architecture for the Management of ECA Rule-Based Clinical Protocols. 15th IEEE Symposium on Computer-Based Medical Systems (CBMS 2002), Maribor, Slovenia, IEEE Computer Society.

Dube, Kudakwashe and Bing Wu (2001). "Supporting Clinical Laboratory Test-Ordering Protocol Specification, Execution and Management: an Event-Condition-Action Rule and Database Approach." Healthcare Informatics Journal 7(1): 20-28.

Mansour, Essam, Bing Wu, Kudakwashe Dube, et al. (2006). An Event-Driven Approach to Computerizing Clinical Guidelines Using XML. In the First International Workshop on Event-driven Architecture, Processing and Systems (EDA-PS'06), In conjunction with ICWS 2006 / SCC 2006., Chicago, USA, IEEE Computer Society.

Paton, Norman W. and Oscar Diaz (1999). "Active database systems." ACM Comput. Surv. 31(1): 63--103.

Umeshwar, Dayal, Barbara T. Blaustein, Alejandro P. Buchmann, et al. (1988). "The HiPAC Project: Combining Active Databases and Timing Constraints." SIGMOD Record 17: 51-70.

Widom, Jennifer and Stefano Ceri (1996). Active Database Systems: Triggers and Rules For Advanced Database Processing, Morgan Kaufmann.

Wu, Bing and Kudakwashe Dube (2001). PLAN: A Framework and Specification Language with an Event-Condition-Action (ECA) Mechanism for Clinical Test Request Protocols. Hawaii International Conference on System Sciences (HICSS-34), Los Alamitos, California, IEEE Computer Society.

Wu, Bing and Kudakwashe Dube (2001). Applying Event-Condition-Action Mechanism in Healthcare: A Computerized Clinical Test-Ordering protocol System (TOPS). CODAS.