2006-09-01

An Event-Driven Approach to Computerizing Clinical Guidelines Using XML

Bing Wu
Dublin Institute of Technology, Bing.Wu@dit.ie

Essam Mansour
Dublin Institute of Technology

Kudakwashe Dube
Dublin Institute of Technology

Jianxin Li
Dublin Institute of Technology

Recommended Citation

Wu, B. et al. (2006) *Proceedings of the First International Workshop on Event-driven Architecture, Processing and Systems (EDA-PS'06), in conjunction with ICWS 2006 the IEEE Services Computing Workshops (SCW'06)*, Chicago, USA, doi:10.1109/SCW.2006.3

An Event-Driven Approach to Computerizing Clinical Guidelines Using XML

Essam Mansour, Bing Wu, Kudakwashe Dube and Jian Xing Li
School of Computing, Dublin Institute of Technology, Kevin Street, Dublin 8, Ireland
E-mails: <firstname>.<surname>@dit.ie

Abstract

Clinical events form the basis of patient care practice. Their computerization is an important aid to the work of clinicians. Clinical guidelines or protocols direct clinicians and patients on when and how to handle clinical problems. Thus, clinical guidelines are an encapsulation of clinical events. Hence, an event-driven approach to computerizing the management of clinical guidelines is worthy of investigation. In our framework, called SpEM, the main clinical guideline management dimensions are specification, execution, and manipulation. This paper presents an event-driven approach, within the context of the SpEM framework, to manage clinical guidelines. The event-driven approach is based on the event-condition-action (ECA) rule paradigm in which the ECA rules are specified using an XML-based language over an electronic healthcare record (EHCR) implemented using an XML-enabled DBMS. This approach facilitates the easy querying, operations and execution replay for clinical guidelines. The approach provides a ready solution to the problem of the integration of clinical guideline management systems (CGMS) and the EHCR. This creates an "active EHCR" in which reactivity is defined by the medical logic in the clinical guideline. The paper practices the approach presented here by using a simplified clinical guideline/protocol from the domain of clinical laboratory investigation for microalbuminuria screening.

1. Introduction

The clinical practice guidelines (CPGs) provide guides for clinicians and patients in determining recommended strategies for managing and monitoring the patient's condition [1]. Clinical guideline automation is one of the suggested methods for improving and enhancing the health care services [2]. Clinical guideline automation requires a number of management aspects. *First,* clinical guidelines should be specified in a standard computer interpretable format that is easy to be shared and disseminated among distributed and heterogonous systems. Computer interpretable guidelines should be generic to facilitate their sharability and applicability to a wide variety of clinical cases. *Second,* for these generic clinical guidelines to be usable locally, the health care organizations need to customize them to suit their database and environment. *Third,* health care organizations also need to tailor computer interpretable guidelines to generate a patient plan. The *fourth* aspect is to make the patient plan an executable object that changes over time. The *fifth* aspect deals with query, operations and execution replay support to the computerized guidelines. This fifth aspect provides flexibility in managing huge number of patients. Moreover, it facilitates the evaluation for the clinical guidelines and the observation for the patient plan. The *sixth* and last aspect is to provide this clinical guideline management system (CGMS) infrastructure within the context of a distributed and heterogonous healthcare setup.

This paper presents an event-driven approach to the problem of clinical practice guideline management. The approach is based on the event-condition-action (ECA) rule paradigm [3] within database systems and uses a specification language based on XML [4]. Most healthcare information systems utilize Database Management Systems (DBMS) in managing the Electronic Health Care Record (EHCR). The modern DBMS provides efficient management of data and information. Furthermore, the modern DBMS supports the ECA rule. XML together with the associated query language XQuery assist in providing effective support for managing computer interpretable clinical guidelines and their execution. Thus, the CPG specification and execution history could be queried and operated upon. Therefore, this approach combines an event-driven paradigm based on ECA rules, exploitation of DBMS features and the use of XML technologies in order to facilitate the integration of management support for computer interpretable guidelines and the systems for managing the EHCR. The result of this approach is an *"active electronic healthcare record"* in which reactivity is guideline-based and driven by events reflected within the patient record leading to interventions that are recommended by guidelines. The use of XML facilitates the dissemination of the specified guidelines among heterogeneous systems and makes possible the exploitation of modern XML technologies.

The rest of this paper is organized as follows: Section 2 discusses the background and related work. Section 3 presents our event driven approach and the SpEM framework for computerizing CPGs. Section 4 presents a specification language, called AIM-SL, for the clinical guideline based on the ECA rule paradigm and XML.

Proceedings of the IEEE Services Computing Workshops (SCW'06)
0-7695-2681-0/06 $20.00 © 2006 IEEE

Section 5 discusses an event driven execution model for a CPGs. Section 6 presents the proposed conceptual architecture of a system that utilizes the approach and framework. Section 7 demonstrates the approach using a sample case study for managing microalbuminuria screening guidelines for diabetes patients. Section 8 concludes the paper.

2. Background and Related Work

The event-driven approach and systems have been used in several application domains, which include information delivery services, network management and distributed systems. The monitoring and detection of clinical events play key roles in the practice of disease management and patient care. In a pioneering study that used a computer to detect and respond to clinical events, MacDonald [5] concluded that computer detection and response to simple clinical events will have a positive effect on the behavior of clinicians and build a foundation for more complex clinical event detection. Studies of clinical events occurring before and during disease progression help to inform treatment and deepen understanding of disease progression [6]. Hence, clinical events could be seen as a core driver to clinical practice guidelines and protocols.

The event-driven approach based on ECA rule paradigm has been adopted in computerizing clinical guidelines. Besides our work, TOPS [7, 8], the Arden Syntax [9] and HyperCare [10] are the two major relevant works that computerized CPGs by following the event-driven approach. Arden Syntax does not distinguish between the generic specifications of clinical guidelines and the generated instance. HyperCare computerizes a clinical guideline without providing a generic mechanism to be applied to other guidelines. Like other computer-based clinical guideline approaches, both the Arden Syntax and HyperCare provide a little or no support for manipulation and querying clinical guideline information.

Several research efforts propose XML-based languages for formalizing clinical guidelines [11, 12, 13] and ECA rules [14] with the aim of providing for improved sharability. As in other approaches [12, 15, 16], these research works focus on specification and execution while providing a little or no support for manipulation or querying of clinical guideline information.

This paper presents the second stage of a research work that has developed TOPS [7, 8], whose aim was to provide a generic approach [17] for managing clinical guidelines. While TOPS addressed, the limitations of existing systems outlined in preceding paragraphs, it did not exploit XML technologies to support guideline management and information dissemination. Furthermore, TOPS functionality needed further enhancements with features such as execution scenario replay for guideline instances. This paper presents work within the context of on-going work that differs from above works in providing a unified management framework for CPGs that not only specify, store, execute the CPGs, but also manipulate, query, share the specified CPGs, and easily integrated into the systems managing EHCR.

3.1. An Event-Driven Approach to Computerizing Clinical Guidelines

Clinical guidelines should ideally be enforced as soon the patient information, which represents some change in the patient circumstances, becomes available. It would benefit the patient, if physicians could be informed of the recommendations from clinical guidelines based on the contents of the medical record. Thus, the event-driven approach appears to be the best way to support effective and efficacious computerization of clinical guidelines.

Figure 1 illustrates the main aspects making up our event-driven approach to CPG computerization as presented in this paper. The approach hypothesizes that: 1) Clinical events could be the basis for the event-driven approach to computerizing clinical guidelines; 2) The ECA rule paradigm as implemented in database systems is a promising technology for supporting an event-driven approach to computerizing CPGs; 3) Using XML to specify ECA rules enables sharability and easy management of CPGs that is computerized by following an event-driven approach; and 4) clinical events, XML, clinical guidelines and ECA rules constitute the core concepts whose mixture, in the context of database technology, provides a strong basis for supporting a new and useful type of electronic healthcare record that has active capability drawn from guidelines.

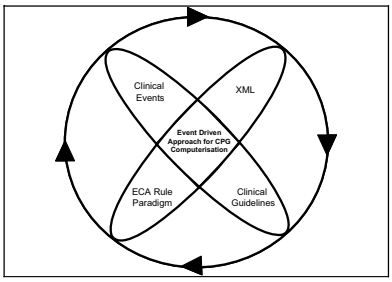

Figure 1. The main aspects to the event-driven approach to the computerisation of CPG

The event-driven approach to clinical guideline computerization is centered around the concept of an *active electronic health care record (EHCR)*. In an active EHCR, the active aspect is made up of computerized clinical guidelines in execution. Since the EHCR itself is patient specific, its active part must necessarily be patient-specific although provision should be made to extend the elements of reactivity to a group of patients. Most of the medical information is managed using the DBMS that provides efficient handling for the data storage and retrieval. The modern active DBMS supports the ECA rule, which is event-driven in nature.

The active database system could be used as a basis for an event-driven approach to the computerization of clinical guidelines based on ECA rule paradigm. This approach naturally gives rise to an active EHCR in which a set of active rules represent clinical guidelines and react to changes within the patient's medical record. In the approach presented here, clinical guideline specification is based on XML in order to support both easy management and exchange of the computerized guidelines.

3.2. The SpEM Framework

SpEM [17] (Specification, Execution and Manipulation) is our framework supporting the computerized CPG management. As illustrated in Figure 2, the SpEM contains three planes: *specification* plane; *execution* plane and; *manipulation* plane with the active database as the integrating factor among the three planes.

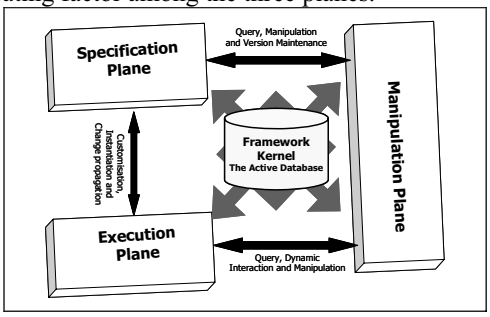

Figure 2. SpEM framework

The CPG management process is fitted into the three planes of the framework. In the *specification plane,* CPGs are translated into formal specifications and held permanently in the database in a manageable form. In the *execution plane,* the stored CPG specifications are used to create patient-specific CPG instances that are executed by an engine based on the active database. The guideline execution is modelled as a workflow or care flow process that is driven by the guideline instance, which is persisted in the active database. In the *manipulation plane,* the specifications and the executing CPG instances are manipulated using supported operations and queries based on the SQL features and extensions of the underlying database system. Thus, the active database is the kernel to an execution engine for CPGs while also permanently holding the EHCR in addition to CPG information and running care flow process information. It also provides facilities for querying and manipulating information. Thus, we have the concept of an *active EHCR*. Furthermore, the active database can guarantee future sharing of information through the generic nature of databases and the standard language, the SQL as well as the standard data format, XML, which is increasingly becoming a core part of the modern DBMS.

4. The Language for Specifying Clinical Guidelines

This section presents an event-driven specification language, AIM-SL, for clinical guidelines based on ECA rule paradigm and XML. AIM-SL stands for *Active Information Management Specification Language*. AIM-SL is one of two components of a generic and high-level declarative language, called *Active Information Management language* (AIM), for specifying, querying and manipulating clinical guidelines. The second component language is called AIM Query Language, AIM-QL, which defines the query, operator and replay language for supporting the manipulation plane of the SpEM framework. AIM-QL will not be presented in this paper.

4.1. The Model for the AIM- SL

Figure 3 illustrates the protocol specification model of AIM-SL. The figure expresses the XML Schema of the AIM-SL grammar in the form of a tree structure.

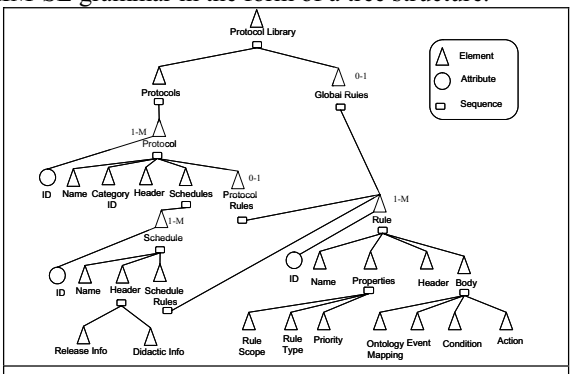

Figure 3. A tree model for the XML Schema of AIM-SL

The tree structure in Figure 3 uses a notation that is an extension of the notation used by Yoshikawa and Amagasa [18]. The AIM-SL model is made up of five main concepts, which are: protocol library, protocol, schedule, rule and header. The protocol library is a collection of protocols and global rules. Global rules are rules whose scope is global to all protocols and patient population. The protocol is a collection of schedules and rules whose scope is all schedules within the protocol. The schedule is a collection of rules. The header is a collection of pieces of release and didactic information. The didactic information provides literature related to the CPGs and cites the reference to the source of the knowledge that is encapsulated in the computerized CPGs. In the CPG execution, the didactic information could be attached with the clinical recommendation as an explanation.

4.2. The Specification of the Rule in AIM-SL

As shown in Figure 4, the rule consists of the elements: `name`, `properties`, `header`, and `body`. The rule also has an `id` attribute. The `properties` element consists of the elements: `ruleScope`, `ruleType`, and `priority`. The `ruleScope` element specifies the scope of the rule. The global scope spans an entire organization. The protocol spans its own scope, which includes all schedules. The schedule also spans its own scope. The `ruleType` element specifies the type of the rule, which could be either static or dynamic. A static rule performs an action subject to occurrence of a time event. A dynamic rule is an ECA rule. The `priority` element specifies the order in which the rule should be invoked.

```
<xsd:element name="rule">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="properties">
        <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="ruleScope" type="ruleScopeDT"/>
          <xsd:element name="ruleType" type="ruleTypeDT"/>
          <xsd:element name="priority" type=" xsd:integer"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element ref="header"/>
    <xsd:element name="body">
      <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="ontologyMapping"/>
        <xsd:element ref="event"/>
        <xsd:element ref="condition" minOccurs="0"/>
        <xsd:element ref="action"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID"/>
 </xsd:complexType>
</xsd:element>
```

Figure 4. The XML Schema definition for the rule.

The `body` element has a complex type composed of three elements: `ontologyMapping`, `event`, `condition`, and `action`. These elements are described in the next paragraphs.

a) Ontology Mapping. In order to make clinical guidelines generic, the guideline specifications should be independent of the health care organization's database. Therefore, general terms, in the guideline specification, are used. As shown in Figure 5, the `ontologyMapping` element allows an individual organization to customize the generic specification by creating a mapping between the generic terminology and local terminology. The local terminology is defined in the local schema used by the local medical record system

```
<xsd:element name="ontologyMapping">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="term" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:choice>
           <xsd:element name="event">
            <xsd:complexType>
              <xsd:sequence>
             <xsd:element name="name" type="xsd:string"/>
             <xsd:element name="DBOperation" type="DBOperationDT"/>
                <xsd:element name="documentName" type="xsd:token"/>
                <xsd:element name="nodePath" type="xsd:string"/>
                <xsd:element name="nodeName" type="xsd:token"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
           <xsd:element name="element">
              <xsd:complexType>
               <xsd:sequence>
                <xsd:element name="name" type="xsd:string"/>
                <xsd:element name="documentName" type="xsd:token"/>
                <xsd:element name="nodePath" type="xsd:string"/>
                <xsd:element name="nodeName" type="xsd:token"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          </xsd:choice>
          <xsd:attribute name="id" type="xsd:ID"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure 5. The XML Schema definition for the ontology mapping.

b) Event. Here, the domain events have been classified into an episode, relative time event, and absolute time event. The episode is one in a series of connected incidents associated with a domain entity. The admission of a patient is an episode that may happen to a patient within the context of a series incident in disease management. The relative time event is a time point measured relative to the time of occurrence of an episode. The relative time event happens once-off or repeatedly. The absolute time event is a time point specified by an absolute timestamp, such as "1 June 2006". As shown in Figure 6, the `event` element has a complex type composed of one of the elements: `absoluteTime`, `relativeTime`, and `episode`. The event also has an id attribute. The `absoluteTime` element has the dateTime simple type.

```
<xsd:element name="event">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="on">
        <xsd:complexType>
        <xsd:choice>
          <xsd:element name="absoluteTime" type="xsd:dateTime"/>
          <xsd:element ref="episode"/>
          <xsd:element ref="relativeTime" />
        </xsd: choice >
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="episode">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="id" type="xsd:IDREF"/>
      </xsd:extension>
    </xsd:simpleContent>
<xsd:element name="relativeTime">
  <xsd:complexType >
    <xsd:choice>
      <xsd:element name="onceOff" type="base_relativeTime_DT"/>
       <xsd:element name="every">
         <xsd:complexType>
           <xsd:complexContent>
                <xsd:extension base="base_relativeTime_DT">
                  <xsd:sequence>
                <xsd:element name="for" type="xsd:duration"/>
                  </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

Figure 6. The XML Schema definition for the event.

```
<xsd:element name="condition">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="description" type="xsd:string"
minOccurs="0"/>
       <xsd:element name="operand1" type="xsd:token"/>
       <xsd:element name="operator" type="logicalOperatorDT"/>
       <xsd:element name="Operand2">
        <xsd:simpleType>
          <xsd:union memberTypes="xsd:double xsd:string"/>
        </xsd:simpleType>
       </xsd:element>
     </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"/>
  </xsd:complexType>
</xsd:element>
```

Figure 7. The XML Schema definition for the condition.

c) Condition. A condition is a logical expression meaningful to the domain. Here, the condition is two operands connected by comparison operators (=, <>, >, >=, <, and <=).

As shown in Figure 7, the condition element consists of a sequence of elements: `description`, `operand1`, `operator`, and `operand2`. The condition also has an id attribute. The `description` element is an optional element that explains the meaning of

```
<xsd:element name="action">
  <xsd:complexType>
   <xsd:sequence>
     <xsd:element name="description" type="xsd:string" minOccurs="0"/>
       <xsd:element name="do">
        <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="AIM-QLAction" type="AIMQLActionDT" minOccurs="0"/>
        <xsd:element name="proceduralAction" type="proceduralActionDT"
minOccurs="0"/>
      </xsd:sequence>
     </xsd:complexType>
   </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="id" type="xsd:ID"/>
 </xsd:complexType>
 </xsd:element>
```

Figure 8. The XML Schema definition for the action.

the condition.

d) Action. An action is an operation meaningful to the domain. As shown in Figure 8, an action could be a command such as add rule, remove rule and change status. The command is specified using AIM Query Language (AIM-QL). An action could also be a procedural action, such as sending an email, SMS, or invoking a method.

5. The Event-Driven Execution Model for the Patient Plan

In this section, an event driven execution model for a patient plan is explained. The patient plan is an instance of a generic protocol customized for a specific patient. The patient plan exists as an active collection of patient specific guideline information that changes over time. Rules in a patient plan may be dynamically added or removed. The state of the rule changes from time to time. The patient plan grows with the time, because it keeps its execution history within its body.

5.1. The Patient Plan Structure

The structure of the patient plan is illustrated in Figure 9. Figure 9 expresses the temporal XML Schema of the patient plan as a tree structure that uses the notation of Figure 3. The patient plan is composed of a sequence of the elements: `state`, `schedules`, `protocolRules`, and `globalRules` and a set of the attributes: `validity`, `ProNo`, and `PatNo`. The `validity` attribute group specifies the period in which the content of the element is valid. The attributes called `ProNo` and `PatNo` are for the protocol number and the patient number respectively. The `state` element is composed of one `value` element or more. Each `value` element has one of the following values: *ready, executed, removed*, and *expired*. If the value of the `state` element is changed, a new `value` element is added accordingly. The `schedules` element is composed of a sequence of the elements: `state` and `scheduleRules`; and a set of the attributes: `validity` and `SNo`. The `protocolRules`, `globalRules`, and `scheduleRules` elements are each composed of one `rule` element or more. The `rule` element consists of a sequence of the elements: `state` and `trigger`, and a set of the attributes: `validity` and `RNo`. The `trigger` element represents a portion of the clinical guideline knowledge that is captured as an ECA rule. Here, the rule is mapped to one or more SQL/XML trigger or stored procedure.

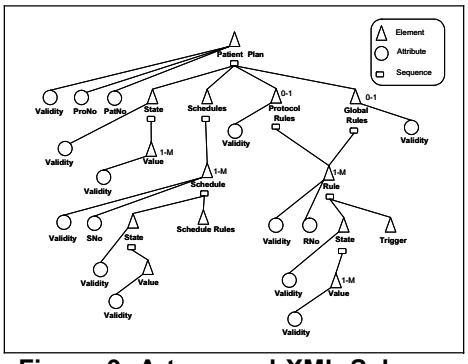

Figure 9. A temporal XML Schema for the patient plan.

5.2. The Patient Plan Execution Model

The patient plan execution model is based on the event-driven model. Figure 10 illustrates the patient plan execution model. The patient plan registers its events in the *event manager.* The event manager deals with the three types of events: the absolute time event, the episode, and the relative time event. Once the event occurs, the event manager notifies the patient plan. If the associated condition is evaluated to true, the associated action is executed.

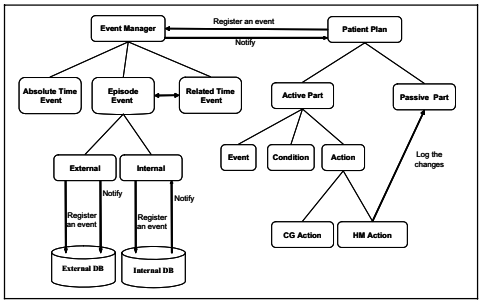

Figure 10. Outline of the patient plan data and an event manager integration architecture.

The *absolute time module* is a timer associated with jobs that notify a specific patient plan. The *episode module* notifies the relative time module, whenever the related episode occurs. The *external database* is integrated into the event manager using the peer-to-peer architecture.

5.3 Generating the Patient Plan

The generic guideline specification represented in AIM-SL is customized for a specific patient. The aim of the customization process is to produce a patient-specific instance, the patient plan. As shown in Figure 10, the patient plan consists of an *active part* and a *passive part*. The active part represents the execution process for a patient plan. The passive part holds the execution history of the patient plan. The *action part* is composed of the clinical guideline's recommendations, *CG action*, and the actions that are associated with updating the execution history of a patient plan, History Maintenance (*HM) action*. The CG action is extended by adding patient

specific information, such as the patient ID. The HM action is a set of update operations applied for the passive part to log the changes that occurred in the patient plan. The condition part also is extended to deal with a specific patient.

6. The Proof-of-Concept Prototype System for Managing CPGs Using the Event-Driven Approach

This section discusses the design and architecture of the proof-of-concepts system, AIMS.

6.1. The Conceptual Architecture

The architecture of AIMS has three modules, as shown in Figure 11. External to AIMS, there are the external systems and users. The top module is the `guideline management module` that allows users to specify, store, customize, execute, manipulate and query CPGs. The `guideline management module` allows also the external systems to supply and receive information from the system. The middle module extends the ECA rule execution mechanism of the underlining DBMS and handles connections to the database. The separation between the `guideline management module` and the `ECA rule extension module` facilitates the utilization of the `ECA rule extension module` in different application. The bottom module is a modern active DBMS that provides ECA rule mechanism and XML storage and retrieval.

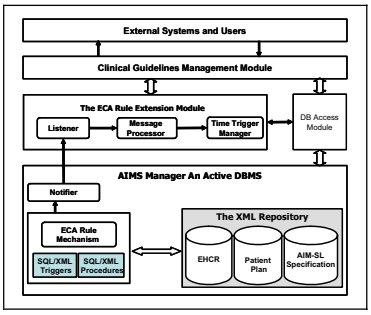

Figure 11. The conceptual architecture of AIMS.

The functionality of the `Event Manager`, which is illustrated in Figure 10, is supported through the `AIMS Manager` and the `ECA Rule Extension Module`. The `episode event` and `the Internal DB` are supported by using the ECA rule mechanism provided by the active DBMS. Although, the modern active DBMS supports the ECA rule paradigm, there are some features that are not supported but are required by this application. Some of these features are: time trigger, external communication mechanism for triggers, and allowing one trigger's action to create another trigger. It is not allowed to execute a SQL Data Definition Language (DDL) statement, such as creating a trigger, within the trigger's action, because this forces a commit within the triggering transaction before its completion. A premature commit violates one of the transaction characteristics.

The absolute event, relative time event, the episode and External DB are supported by the `ECA Rule Extension Module`, shown in Figure 11. The time trigger is supported by the `time trigger manager`, which extends the DBMS trigger mechanism. As shown in Figure 11, the *notifier* and the *listener* together provide support for the external communication mechanism for triggers. The notifier sends a message, which could be a statement for creating or dropping a trigger or invocation for a method inside or outside the system, to the listener. The listener sends the message to the *message processor*. The message processor interprets the message and then acts on the message.

The architecture utilizes the modern active DBMS that has support for the ECA rule paradigm and XML repository. There is no native XML management system that provides ECA rule and XQuery support. Therefore, the implementation of this architecture currently uses Oracle 10g DBMS as the active DBMS. Oracle 10g DBMS [19] provides support for SQL/XML [20, 21] and XML repository. SQL/XML provides the ability to incorporate XML queries with SQL.

6.2. Support for the SpEM Guideline Management Framework in AIMS

AIMS provides support for SpEM framework for clinical guidelines as follows:

a) *Specification Plane:* Using AIM-SL, the clinical guidelines are formally represented and specified in a generic form. This phase requires the involvement of the domain expert. For a specific healthcare institution, the generic specification is customized. In the customization phase, the `ontologyMapping` element associated with AIM-SL rules should be defined according to the institute's database. This phase requires the involvement of a technical expert. An ontology for the database schema could be used to automate the customization phase.

b) *The Execution Plane:* A patient plan is created for each patient. The patient plan rules, which are represented using SQL/XML is registered through the `AIMS Manager` and `Time Trigger Manager` according to the rule type.

c) *The Manipulation Plane:* The ordinary XML tools provide the basic support for the query, manipulation, dissemination and sharing for both AIMS-SL specification and the patient plan. The AIM-QL provides a high level query language for both specification and patient plan.

7. A Simplified Case Study in Computerizing a Clinical Protocol

In this section, the management aspects, specification, customization, execution, query and manipulation, are illustrated using sample of microalbuminuria screening (MAS) protocol.

7.1. The Specification

The microalbuminuria screening (MAS) protocol has a schedule. The schedule contains two main rules that are presented in Figure 12. Figure 12 shows two rules for handling MAS. In The protocol is specified and customized using AIM-SL. Figure 13 illustrates a browsing view for the content of the AIM-SL specification of the protocol.

Rule MAS1: **ON** day 2 of the patient admission,
DO order the test albumin creatine ratio (ACR).

Rule MAS2: **ON** receiving the result of test ACR
IF the ACR result is greater than 25
DO order ACR test twice on days
number 6 and 38 of the patient admission

Figure 12. The main rules in the MAS protocol.

7.2. Execution of a Protocol

An executable patient plan is generated based on the specified protocol, which is shown in Figure 14. In the generation process the rule body (`ontologyMapping`, `event`, `condition` and `action`) is used to generate a SQL/XML trigger or stored procedure, which should be registered in the system to become ready for the execution. The stored procedure is used for the static rule, which is invoked by the time trigger manager. The SQL/XML trigger is used for dynamic rules. As shown in Figure 13, MAS2 define a set of clinical recommendation that should happen once the result of the required test in MAS1 is received. Assuming the result of ACR test is received on day 3 and its value is greater than 25. The action of MAS2 of the patient plan adds two new rules, MAS3 and MAS4, and then these changes are logged in the patient plan. Figure 15 illustrates a portion of the patient plan on day 4. This portion has the history of the patient plan and its execution.

```
-<protocol id="ProID-MAS">
  <name> microalbuminuria screening (MAS) protocol </name>
  <categoryID>CID316</categoryID>
 +<header>
 -<Schedules>
   -<schedule id="SIDMAS">
    <name>Basic MAS</name>
   +<header>
   -<scheduleRules>
    +<rule id="MAS1">
    -<rule id="MAS2">
     <name>Rule 2 of basic MAS</name>
     -<properties>
      <ruleScope>Schedule</ruleScope>
      <ruleType>Dynamic</ruleType>
      <priority>0</priority>
      </properties>
     + <header>
     - <body>
      -<OntologyMapping >
       <term id="E2.1">
        <event>
        <name>TestResultRecieved</name>
        <DBOperation>insert</DBOperation>
        <documentName>AIMS_TESTORDERRESULT_TAB</documentName>
        <nodePath>/TestResult</nodePath>
        <nodeName>TestResult</nodeName>
        </event>
       </term>
      </OntologyMapping>
  -<event id="E2.1">
   <on>
    <episode id="EpID2.1">TestResultRecieved</episode>
   </on>
  </event>
 +<condition id="ID36">
 - <action id="AID36">
 - <do>
  -<AIM-QLAction>
   - <add>
    +<rule id="RID3">
    +<rule id="RID4">
   </add>
  </AIM-QLAction>
  </do> </action></body></rule></scheduleRules></schedule>
 </Schedules></protocol>
```

Figure 13. The MAS protocol specified using AIM-SL.

```
<patientPlan ProNo="ProID-MAS" PatNo="PIDA" startTime="1" endTime="Now" >
   <state startTime="1" endTime="Now">
    <value startTime="1" endTime="Now">ready</value>
   </state>
  <schedules startTime="1" endTime="Now">
     <schedule schIDREF="SIDMAS" startTime="1" endTime="Now">
        <state startTime="1" endTime="Now">
           <value startTime="1" endTime="Now">ready</value>
        </state>
        <scheduleRules>
           <rule RNO="MAS1" startTime="1" endTime="Now">
              <state startTime="1" endTime="Now">
                 <value startTime="1" endTime="Now">ready</value>
              </state>
              +<Trigger></Trigger>
           </rule>
           <rule RNO="MAS2" startTime="1" endTime="Now">
              <state startTime="1" endTime="Now">
                <value startTime="1" endTime=" Now">ready</value>
              </state>
             +<Trigger></Trigger>
           </rule>
        </scheduleRules>
     </schedule>
  </schedules></patientPlan>
```

Figure 14. The initial patient plan generated for patient PIDA

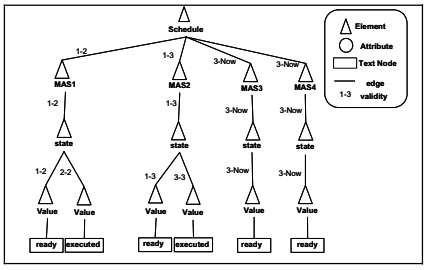

Figure 15. A part of the patient plan on day number 4 of patient admission.

7.3 Query and Manipulation

The AIM-SL specification and the patient plan are stored in an XML repository managed by a modern active DBMS, such as Oracle 10g. The SQL/XML can be used to query and update the specification and the patient plan. Let us assume the clinician would like to modify the action of rule MAS2 by adding rule 6 that should fired on day number 40 of the patient admission to order serum creatinine ratio (SCR). Figure 16 illustrates the Oracle SQL/XML update statement that adds the XML element of rule 6, which assigned to a varchar variable called rule6. The update statement inserts rule 6 as a child of the `add` element, which is in this path, //scheduleRules/rule[2]//AIM-QLAction/add. The SQL/XML is also used to query the specification and the patient plan.

Let us suppose that on day number 4 of the patient admission, the clinician was interested in 1) retrieving the name of all schedule rules from the MAS protocol specification and 2) retrieving the id of all executed rules in the patient plan of patient number PIDA. Query 1, in Figure 16, retrieves the names. Query 2, in Figure 16, retrieves the ids of those rules, whose state value is

Update Statement

```
SQL>UPDATE aims_protocol_tab
 SET OBJECT_VALUE =
 insertChildXML(OBJECT_VALUE,
 '//scheduleRules/rule[2]//AIM-QLAction/add',
 'rule',  XMLType(Rule6))
 WHERE existsNode(OBJECT_VALUE,'//scheduleRules/rule[2]//AIM-
QLAction/add')= 1;

1 row updated.
```

Query1: A Query Statement for AIM-SL Specification

```
SQL>select p.OBJECT_VALUE.extract('//scheduleRules/rule/name')  as
RuleName  from aims_protocol_tab  p;
RULENAME
--------------------------------------------------------------------------------
<name>Rule 1 of basic MAS</name>
<name>Rule 2 of basic MAS </name>
```

Query2: A Query Statement for patient plan

```
SQL>select
p.OBJECT_VALUE.extract('/patientPlan[@PatNo="PIDA"]//rule[state="execu
ted"]/@RNO')  as RuleID  from aims_pplan_tab p;
RuleID
--------------------------------------------------------------------------------
MAS1MAS2
```

Figure 16. Oracle SQL/XML query and update statements

"executed". If the clinician is interested in reviewing the patient plan form 3rd day to 9th day of the patient admission, a temporal support in the XPath and XQuery is required.

8. Summary, Conclusion and Future Work

This paper has presented an event-driven approach for supporting the computerization of clinical guidelines. Clinical events are considered here to be drivers within the logic of CPGs and, hence, form the basis of the rationale for the event-driven approach to the computerization of CPGs. In this paper, the event-condition-action (ECA) rule paradigm is utilized as the basis for the event-driven approach to computerizing CPGs. Guidelines are specified by using the language, AIM-SL, which is based on XML. An event-driven model of execution for guideline instance was also presented. The approach was demonstrated by using a proof-of-concept prototype system and an extract of clinical guideline for the microalbuminuria protocol.

This approach facilitates the query and manipulation of the specified clinical guidelines. It also supports the integration of the clinical guideline system into the health care system by providing an active electronic health care record. This approach leads to useful concept of an active electronic health care record where reactivity is derived from clinical guidelines.

The evaluation and the implementation of the approach and proof-of-concepts system are still ongoing. As future work, a high level query and manipulation language that is easy to be used by the clinicians will be developed. Moreover, the XPath and XQuery will be extended to provide temporal query support.

References

[1] M. J. Field and K. N. Lohr, *Guidelines for Clinical Practice: From Development to Use*. 1992, Washington, DC: National Academy Press.

[2] J. M. Grimshaw and I.T. Russell, *Effect of clinical guidelines on medical practice: a systematic review of rigorous evaluations.* Lancet, 1993. 342: p. 1317-22.

[3] Jennifer Widom and Stefano Ceri, *Active Database Systems: Triggers and Rules For Advanced Database Processing*. 1996: Morgan Kaufmann.

[4] Tim Bray, Jean Paoli, McQueen Sperberg, Eve Maler, and Fran¦§ois Yergeau, *Extensible Markup Language (XML) 1.0 (Third Edition)*. 2004, W3C Recommendation.

[5] C.J. McDonald, *Use of a computer to detect and respond to clinical events: its effect on clinician behavior.* Ann Intern Med., 1976. 84(2): p. 162-167.

[6] A.U Khanda, I Gemmellb, A.C Rankina, and J.G.F Cleland, *Clinical events leading to the progression of heart failure: insights from a national database of hospital discharges.* European Heart Journal, 2000. 22(2): p. 153-164.

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

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

[9] P. D. Clayton, Pryor T. A., Wgertz O. B., and Hripcsak G. *Issues and Structures for Sharing Medical Knowledge Among Decision-making Systems: The 1989 Arden Homestead Retreat*. in *Annu Symp Comput Appl Med Care*. 1989.

[10] P Caironi, L Portoni, C Combi, F Pinciroli, and S Ceri. *HyperCare: a Prototype of an Active Database for Compliance with Essential Hypertension Therapy Guidelines*. in *AMIA Ann Fall Symposium*. 1997. Philadelphia, PA: Hanley and Belfus.

[11] Anil K. Dubey and Henry C. Chueh, *An XML-based Format for Guideline Interchange and Execution.* AMIA Annual Symposium, 2000: p. 205-209.

[12] John Fox, Jon Bury, Michael Humber, Ali Rahmanzadeh, and Richard Thomson. *Publets: Clinical Judgement On The Web*. in *AMIA Annual Symposium*. 2001.

[13] Robert A. Greenes, Mor Peleg, Aziz Boxwala, Samson Tu, Vimla Patel, and Edward Shortliffe. *Sharable Computer-Based Clinical Practice Guidelines: Rationale, Obstacles, Approaches, and Prospects*. in *Medinfo 2001*. 2001. London, UK.

[14] Angela Bonifati, Daniele Braga, Alessandro Campi, and Stefano Ceri. *Active XQuery*. in *Proceedings of the 19th International Conference on Data Engineering ICDE*. 2002. San Jose (California).

[15] Lucila Ohno-Machado, John H. Gennari, Shawn Murphy, Nilesh L. Jain, Samson W. Tu, Diane E. Oliver, Edward Pattison-Gordon, A. Greenes Robert, Edward H. Shortliffe, and G. Octo Barnett, *The GuideLine Interchange Format:A Model for Representing Guidelines.* American Medical Informatics Association, 1998. 5: p. 357-372.

[16] Y. Shahar, S. Miksch, and P. Johnson, *The Asgaard Project: A task-specific framework for the application and critiquing of time-oriented clinical guidelines.* Artificial Intelligence in Medicine, 1998. 14: p. 29-51.

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

[18] Masatoshi Yoshikawa and Toshiyuki Amagasa, *XRel: a path-based approach to storage and retrieval of XML documents using relational databases.* ACM Trans. Inter. Tech., 2001. 1(1): p. 110--141.

[19] Muralidhar Krishnaprasad Vikas Arora Zhen Hua Liu. *Native Xquery processing in oracle XMLDB*. in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 2005. New York, NY, USA: ACM Press.

[20] Eisenberg Andrew and Jim Melton, *SQL/XML is making good progress.* SIGMOD Rec., 2002. 31(2): p. 101--108.

[21] Sql/Xml, *the first edition of the SQL/XML standard*. 2003, published by the ISO as part 14 of the SQL standard: ISO/IEC 9075--14.