2001-01-01

APPLYING EVENT-CONDITION-ACTION MECHANISM IN HEALTHCARE: A COMPUTERISED CLINICAL TEST-ORDERING PROTOCOL SYSTEM (TOPS)

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

Kudakwashe Dube
Dublin Institute of Technology

Antenna & High Frequency Research Centre

Conference Papers

Dublin Institute of Technology *Year* 2001

APPLYING EVENT-CONDITION-ACTION MECHANISM IN HEALTHCARE: A COMPUTERISED CLINICAL TEST-ORDERING PROTOCOL SYSTEM (TOPS)

Bing Wu Ph.D* Kudakwashe Dube†

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

†Dublin Institute of Technology

— Use Licence —

Attribution-NonCommercial-ShareAlike 1.0

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

- Attribution.
  You must give the original author credit.
- Non-Commercial.
  You may not use this work for commercial purposes.
- Share Alike.
  If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

For any reuse or distribution, you must make clear to others the license terms of this work. Any of these conditions can be waived if you get permission from the author.

Your fair use and other rights are in no way affected by the above.

Applying Event-Condition-Action Mechanism in Healthcare: a Computerised Clinical Test-Ordering Protocol System (TOPS)

Bing Wu and Kudakwashe Dube
Department of Computer Science, Dublin Institute of Technology, Kevin Street Lower, Dublin 8, Ireland
Emails: {Bing.Wu, Kudakwashe.Dube}@dit.ie

Abstract

This paper addresses issues of the active database application in the challenging healthcare area: the management and execution of computerised clinical practice guidelines/protocols. The problem of how to efficiently and effectively query and manipulate the computerised clinical protocols/guidelines has posed a major challenge but received little research attention until very recently. By proposing a declarative modeling language (PLAN) with an Event-Condition-Action (ECA) mechanism for clinical test-ordering protocols, and an automatic mapping and management system (TOPS), this paper addresses this issue, in an important medical domain, from a unified approach based on an active rule mechanism. The work presented in this paper is part of an on-going research effort that investigates a new application domain for active databases, and proposes some new requirements towards the enhancements of active DBMS functionalities

1. Introduction

The cost of clinical laboratory testing has increased considerably during the past two decades. This has prompted research aimed at controlling clinical laboratory utilization without affecting the continued improvement of the quality of patient care. One of the most effective and proven approaches to clinical laboratory utilization management is the use of clinical test-ordering protocols. The major challenges being faced are those of modeling and specifying clinical guidelines and protocols in a manner that facilitates their integration into information systems, their subsequent management and their linking with the electronic healthcare record systems such as the Synapses Server [1]. Furthermore how to efficiently and effectively manage, query and manipulate the computerised clinical test-ordering protocols has posed a major challenge but received little research attention until very recently.

This paper addresses these challenges by following an active database approach with focus being put on the management and execution of computerised clinical test-ordering protocols.

This paper contains 6 main sections. Section 2 briefs the application domain background and links it with the ECA mechanism. Section 3 presents the general overview of the on-going project. A *declarative modeling language* (PLAN) with an *Event-Condition-Action* (ECA) mechanism for specifying clinical test-ordering protocols and its associated concepts are briefly presented in Section 4. Section 5 discusses the design and development of a computerised clinical *Test-Ordering Protocol System* (TOPS). Finally, Section 6 summaries the whole paper.

2. Clinical Protocols and Event-Condition-Action Mechanism

2.1 Brief background of research on clinical guidelines/protocol

Clinical guidelines/protocols contain medical concepts and knowledge about how to carry out specific activities, such as ordering timely and appropriate tests and planning treatments for clinical patients [2]. The need to improve the quality of healthcare has led to a strong demand for clinical protocols/guidelines supported by computer systems in their creation and execution [3]. Research and practice on the computerised clinical guidelines/protocols have been conducted for over two decades [4]. Recently, they have become one of the major focuses in the healthcare informatics area.

One of the major challenges within this area is not only a matter of disseminating best practice, but also a matter of ensuring that the protocols enshrined in this best practice can be easily and readily *specified* and then *integrated* into the existing computing infrastructure, and are easily *customizable* to suit local practice and needs. However, from a computing technology perspective, most attention have been paid from the Artificial Intelligence discipline towards *specifying* and *executing* computerised protocols with a strong decision-support flavour. Little attention was paid towards the *storage* and *management* of the generated protocols, to which the database applications can play an essential role.

0-7695-1128-7/01 $10.00 © 2001 IEEE

Currently, there is an on-going project in the Dublin Institute of Technology. Its main aim is to develop a generic framework and its associated language to specify automatic clinical test-ordering protocols based on an Event-Condition-Action mechanism. Together with this framework and language, a mechanism and implementation of protocol-based system for the *creation, execution* and *management* of the specified clinical test-ordering protocols employing an active database will be developed.

2.2 ECA rule application and clinical test-ordering protocols

The Event-Condition-Action (ECA) rule (also known as trigger) mechanism is well established in the database community. It originated from the need to free individual applications from behavioural knowledge [5]. An active database is normally referred to as a DBMS with an integrated trigger mechanism. Recently, the real-world application areas of active databases (trigger applications) have been identified by [6] as *Business rules, Scheduling, Supply chain management, Web application* and *Workflow management.* Also in [6], from functional and behavioural points of view, triggers are classified into 9 types: 1) constraint-preserving, 2) constraint-restoring, 3) invalidating, 4) materialized, 5) meta-data, 6) replication, 7) extenders, 8) alerters and 9) ad-hoc triggers. It is pointed out there that type 5, 6 and 7 are clearly derivatives of type 4, and type 4 itself can be considered a specific instance of type 2. In other words, for many trigger applications, the primary purpose is to monitor and maintain some kind of constraint [6].

In the clinical test-ordering domain, the activity of a test-ordering rule in a clinical protocol can be seen as following such a procedure: when any of the specified events occur, check the test-ordering condition; if the condition is true, then a test order is issued. Therefore every test order is a result of an event followed by a decision that is made to order the test. A possible *event* that triggers a test order may be the emergence of a patient with a problem, the passage of time, the occurrence of abnormalities in a patient's condition, or a combination of these events. A possible *condition* can be a specification of the medical condition of a patient. A possible *action* can be the issuing of a test order, the sending of an alarm or the issuing of a reminder to a clinician. Other actions can affect the test-ordering plan itself such as adding new, suspending or even removing a scheduled test order for a patient.

It is important to observe that the working scenario described here has some unique features: *Firstly,* It is *event-driven* and can also be *time-driven.* A clinical test can be ordered based on the patient's condition. It can also be triggered on certain time points for some scheduled regular tests. For example, for a Liver-transplant patient, a U&K test (the clinical meaning is not important here) may be scheduled on days -1, 0, 1, 2, 3, 4, 6, 8, 11(+3). Here –1 means the day before the operation, 0 the day of operation and +3 means every 3 days later on until further notice. *Secondly,* the actions of a test-ordering rule can be *alarm-* or *alert-oriented.* It can also be *dynamic-modification-oriented.* An action of a test-ordering rule may specify that on arrival of a test result, send paging information to a clinician. However, there is a much more complicated scenario. On checking the arrived test result, some more tests may need to be ordered immediately or at some late time points – if the ordering logic is pre-determined. Obviously, it can also be the case that an action may be pending on a medical expert's decision, which involves external actions. *Finally,* the reaction time for a test-ordering rule would generally not be in terms of 'seconds' or 'minutes', but a test order may be repeated at time points within a long time interval as the previous example indicated.

Therefore this may be seen as a new application domain for active databases, which falls under type 9, *ad-hoc* triggers identified in [6] but incorporating special requirements for temporal triggers and comprehensive high-level DBMS facilities for dynamically manipulating triggers automatically and with human concurrence from the application.

3. Overview of the Project

This section serves as an introduction to the following sections. It discusses the context of the paper on a high level. Subsection 3.1 discusses the clinical requirements for the proposed TOPS system. Subsection 3.2 describes the project approach and the overall architecture of the TOPS system.

3.1 The clinical requirements

The main aim of this project is to assist healthcare professionals with the specification, implementation, management and execution of clinical test-ordering protocols. There are two major areas in which this assistance can be given. *First,* assistance can be provided for healthcare professionals to specify and manage a computerised *test-ordering protocol* for a particular *category* of patients, such as Liver-transplant, or Diabetes. This involves creation, storage, query and manipulation of computerised test-ordering protocols on a general level for different categories of patients. *Second,* assistance can also be provided for healthcare professionals to dynamically develop and manage a *patient test-ordering plan* for a particular individual patient. This *patient test-ordering plan* is obtained for the patient from a computerised test-ordering protocol of the particular

category to which the patient belongs. This involves *creation, storage, query, execution* and *dynamic manipulation* of patient test-ordering plans. It is very important to notice the relevance and difference of these levels of assistance. A *test-ordering protocol* is a generic specification of a clinical protocol for a particular *patient category*. An individual patient will only be associated with a *patient test-ordering plan*. Therefore, the main requirements can be listed as follows: 1) A specification language is needed for test-ordering protocols and patient test-ordering plans; 2) Tools are needed for the specification, storage, query and manipulation of computerised test-ordering protocols; and 3) Tools are also needed for the generation (from protocols), execution and dynamic management of patient test-ordering plans.

3.2 The Specification Language and Management System for Clinical Test-Ordering Protocols

To fulfill previously raised requirements, two major technical goals of this project are identified. First, a generic modelling framework and a specification language to assist clinical professionals to specify automated clinical test protocols is to be developed. Second, a management system is needed for creation, storage, query, execution and dynamic manipulation to assist clinical professionals to order correct and timely clinical tests for their patients.

The first goal has been achieved and, a specification framework and language, named *PLAN*, has been developed based on the Event-Condition-Action Mechanism. Section 4 outlines PLAN language and its associated modeling concepts. A more detailed description of PLAN has been given elsewhere [7].

The task for achieving the second goal is well under way. An advanced database application system, named *TOPS*, has been developed based on the active database technology in a heterogeneous healthcare database system. The implementation on the Oracle DBMS platform is on-going. A high level architecture of the Test-Ordering Protocol System (TOPS) is illustrated in Figure 1.

It can be seen that TOPS architecture consists of five main components: Patient Category Manager, Test Ordering Protocol Manager, Test Ordering Plan Engine, an active database and the External Communicator.

The Patient Category Manager deals with categorising patients into proper clinical categories. The Test-Ordering Protocol Manager permits the clinicians to edit, query and manipulate test-ordering protocols. The Patient Test Plan Manager performs the similar tasks as the Test-Ordering Protocol Manager but with the targets as patient test plans. The active DBMS serves the purpose of storing test-ordering protocols, instantiated patient test plans and the execution states of patient test plans. Its trigger mechanism drives the execution of patient test plans. Finally, the External Communicator provides the interfacing services for communication with other Healthcare Information System (HIS), such as Laboratory Information System (LIS) and the EHCR Server (Synapses [1]). More detailed discussions on these components are presented in Section 5.

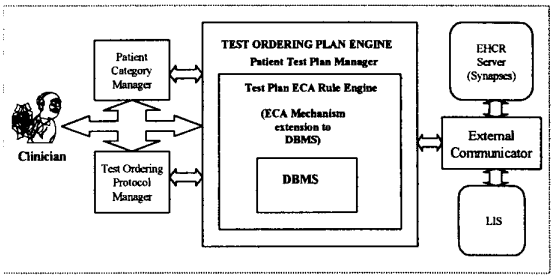

Figure 1. General Architecture of the Test-ordering Protocol System (TOPS)

Before moving into next sections, it may help to sum up the main features of the authors' approach in general: 1) It places emphasis on issues of *storing, executing, querying* and *manipulating* of specified protocols; 2) It draws a clear line between the *static* test request protocols and the *dynamic* patient test plans; 3) It allows *not only* higher level management of generic test request protocols, *but also* lower level management of test plans for individual patients; and 4) It takes a database and ECA rule approach.

4. Modelling and Specifying Clinical Test-Ordering Protocols

This section first introduces some concepts and terms of the PLAN language in Subsection 4.1, then in Subsection 4.2 it presents a simplified version of the generic framework for modeling test-ordering protocols based on the ECA rule mechanism. Two examples of test-ordering rules expressed in a simplified version of PLAN are also presented.

4.1. Main concepts

A *test-ordering protocol* (or *test protocol* in short) is a generic plan for ordering clinical tests for a particular *patient clinical category*. It contains a set of *base schedules* of test orders. Each *base schedule* covers test orders for each variation in patient condition in a specified patient category. A base schedule is expressed in terms of ECA rules. A test protocol also contains a set of *protocol rules*. A *protocol rule* is an ECA rule that dynamically monitors a base schedule of clinical test orders and,

dynamically intervenes in reaction to appropriate clinical situations that may affect test-ordering. In the test-ordering domain, an *event* is an occurrence of something of clinical interest in the care of a patient. Examples of events include 1) test result arrival, 2) passage of time points, 3) patient checks in or out and 4) changes in patient status or condition. A *condition* is a logical expression involving previous test results or other patient attributes. An *action* is an operation that can be performed by the system or by a human agent. The actions can be 1) ordering a further test, 2) issuing an alarm, 3) monitoring patient condition and 4) adding a new or, manipulating an existing test-ordering schedule.

4.2 The PLAN model and declarative language

Based on the above defined core concepts, Figure 2 presents a high-level entity-relationship (ER) model for modeling test-ordering protocols with active rules as the basis for the modeling.

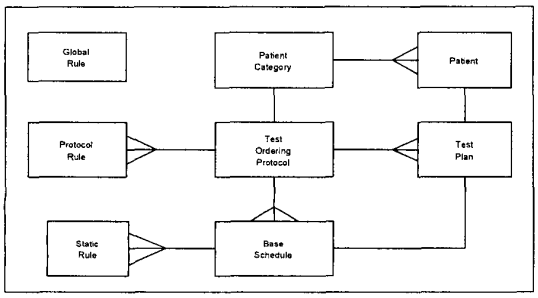

Figure 2. An entity-relationship (ER) model for test ordering protocols

Patients are put into categories based on some clinical indication for purposes of clinical test ordering. Each *patient category* has a *test-ordering protocol*. From this test-ordering protocol, a *test plan* is produced for each individual patient. A *test plan* is generated and consists of a single selected test schedule and all the protocol rules. *Global rules* are applicable to all protocols and serve to monitor and control executing test plans for all patients in all categories; they represent institutional and hospital-wide policies that govern test orders in general.

Based on this model, a declarative specification language PLAN was developed to allow the specification of a test ordering protocol. Figure 3 shows simplified version of one base schedule rule in (a) and one protocol rule in (b). For detailed discussion on PLAN, please refer to [7].

```
RuleName: schedule_rule_1
RuleType: static
RuleStatus: active
StartTime: check_in_day
On day {-1, 0, 1, 2, 3, 4, 6, 8, 11, *3}
Do order_test { U & E}
```

(a) A base schedule rule

```
RuleName: protocol_rule_1
RuleType: protocol
RuleStatus: active
On test result {K}
IF K > 5.5 or K < 3
Do order_test { U & E}
```

(b) A test-ordering protocol rule

Figure 3. Examples of test-ordering rules in PLAN

It can be seen that, in a base schedule rule, the condition part is omitted, as it is always true. In a protocol rule, based on the current available test result (K in the example), a new test order can be added. Each rule has a state. Some typical options for the state of a rule are active, suspended, ready, or executed.

5. TOPS: a Computerised Clinical Test-Ordering Protocol System

This section discusses the development of TOPS. Based on the general descriptions on TOPS in Subsection 3.2, the execution flow of TOPS is described in Subsection 5.1. The architecture design is then discussed in Subsection 5.2. A special case of TOPS' application – the management of patient test plans is discussed in Subsection 5.3. Finally, Subsection 5.4 discusses the implementation issues of TOPS.

5.1 The TOPS Execution Flow

TOPS has four main execution phases: *specification*, *customisation*, *installation* and *execution*, as illustrated in Figure 4. Please note that the numbers in parenthesis for each phase refer to the numbers indicated in the TOPS architecture diagram shown in Figure 5.

During the protocol *specification* phase (1)-(4), the patient category and test ordering protocol are specified. The resulting protocol specification is in the PLAN language and is stored in a database as a set of tables that can be queried and modified.

During the protocol *customisation* phase (5)-(7), the protocol is customised to produce a patient test-ordering plan. Data on the patient's clinical condition is used to select the appropriate test ordering base schedule. A complete test-ordering plan for the patient is composed from the base schedule and the protocol rules.

During the test plan *installation* phase (8)-(14), the patient test plan is interpreted and set up to produce an instantiated patient test-ordering plan into the active DBMS. The base schedule rules and protocol rules are parsed and translated into a set of ECA rules (triggers) with exact event, condition and action specifications that can be monitored, evaluated and executed respectively.

During the test plan *execution* phase (15)-(23), the test plan is executed. The test plan execution is driven by the ECA rule mode of operation. When an event signal occurs, the reactive mechanism goes on to determine if it is a test plan event and, if it is, then its associated condition is evaluated; if the condition is true, a signal is generated to trigger the appropriate action.

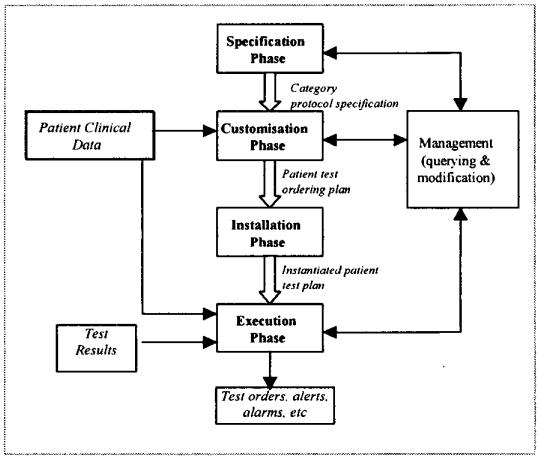

Figure 4 Execution Flow of the Clinical Test-Ordering Protocol System (TOPS)

The installation and execution phases are tightly coupled with the former taking only a short instant, while the later might take several days, weeks or months.

5.2. The TOPS Architecture

The Test Ordering Protocol System (TOPS) detailed architecture is illustrated in Figure 5. The TOPS architecture consists of the five main components: 1) The Patient Category Manager (PCaM); 2) The Test Ordering Protocol Mananger (TOProM); 3) The Patient Test Ordering Plan Manager (TOPlaM); 4) The Test Ordering Plan ECA Rule Engine (TOPEng) – also acts as the reactive wrapper to the underlying database system; and 5) The Test Ordering Protocol Database System – an active DBMS with an event and trigger mechanism.

The Patient Category Manager (PCaM) allows the creation, deletion and modification of patient categories and the assignment of individual patients to these categories for the purposes of test ordering. The categories are symptom, disease or sub-disease based. The PCaM also maintains the list of all patients assigned to existing categories.

The Test Ordering Protocol Manager (TOProM) permits the user to edit new test ordering protocol specifications, query and modify existing test ordering protocol specification components, and delete existing protocol specifications.

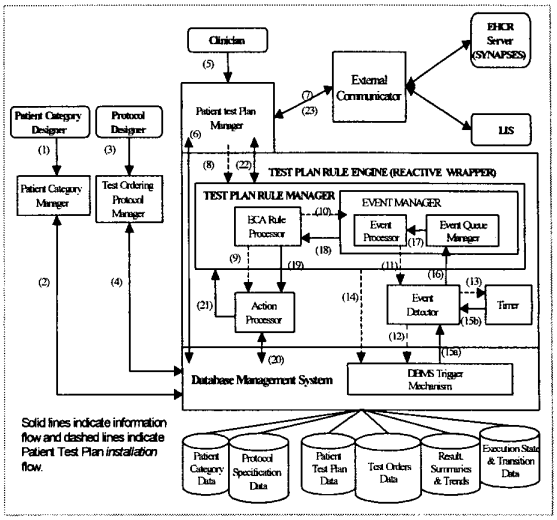

Figure 5. Architecture of the Test Ordering Protocol System (TOPS)

The Patient Test Plan Manager (TOPlaM) allows the user to obtain a categorised patient's test plan specification from the test ordering protocol specification through a protocol customisation process as already described. The TOPlaM permits the user to edit the test ordering plan specification, query and modify existing test ordering plan specification components, and delete existing plan specifications. The major function of the TOPlaM is to allow the user to install, execute and manage the patient test-ordering plan. The non-rule components of the test plan are interpreted and organised into execution state data. The TOPlaM also accesses the patient's medical record through the Electronic Health Care Record (EHCR) Server (Synapses [GBG+98]), sends test orders and receives test results from the Laboratory Information System (LIS).

The database serves the purpose of storing test-ordering protocols, instantiated patient test plans and the execution states of patient test plans. The trigger mechanism drives the execution of patient test plans.

The External Communicator (ExCom) provides the interfacing services for communication with other healthcare information system (HIS) such as LIS and the EHCR server.

The Test-Ordering Plan Engine (TOPEng) is the kernel of the TOPS. The TOPEng consists of the three main

components and a timer: 1) The Test Ordering Plan Rule Manager (TOPRuM) containing the ECA Rule processor (EcaRProc) and the Event Manager (EvM); 2) The Action Processor (AP); 3) The Event Detector (ED); and 4)The Timer.

The TOPRuM handles mainly the event-condition part of an ECA rule, while the AP handles the action part. The ED assists the TOPRuM by generating signals that may indicate the occurrence of a test plan event. The Timer assists the ED by generating time event signals.

The TOPRuM accepts and manages the test plan's ECA rules. The TOPRuM may transform high-level rules into low-level rules by the replacement of logical terms, such as *positive* and *negative* test results, with test result value ranges. The TOPRuM installs the triggers and actions into the ED and the AP respectively. The EcaRProc is responsible for the storage, scheduling, termination management and installation of the test plan rules. The EvM consists of the Event Processor (EP) and the Event Queue Mananger (EQM). The EQM is responsible for buffering event signals from the ED. When event signals occur, the EP processes them according to the installed or registered test plan events and signals the EcaRProc of their occurrences. The AP holds the action implementations, accepts rule action signals from the TOPRuM and executes the appropriate actions on receipt of the event triggering information. The ED monitors the database (orders, results, test result trends and summaries) for changes as required by the TOPRuM. The ED signals the TOPRuM when the changes happen.

Table 1. Protocol Management Operations

	PROTOCOL MANAGEMENT OPERATIONS C-Create, M-Modify, D-Delete, ADT-Activate/Deactivate/Terminate, ✓ - Defined, × - Undefined								
	Static				Dynamic				
	Query	Manipulation			Query	Manipulation			
		C	M	D		C	M	D	ADT
PROTOCOL	✓	✓	✓	✓	✓	×	✓	×	×
CATEGORY	✓	✓	✓	✓	✓	×	✓	×	×
PATIENT	×	×	×	×	✓	✓	×	✓	×
TEST-PLAN	×	×	×	×	✓	✓	✓	✓	✓
SCHEDULE	✓	✓	✓	✓	✓	✓	✓	✓	✓
RULE	✓	✓	✓	✓	✓	✓	✓	✓	✓
EVENT	✓	✓	✓	✓	✓	✓	✓	✓	×
CONDITION	✓	✓	✓	✓	✓	✓	✓	✓	×
ACTION	✓	✓	✓	✓	✓	✓	✓	✓	×

5.3. Management of the patient test plan

There are many use cases that can be discussed for TOPS. Table 1 indicates the main functionality of TOPS. However, because of space limitation, this paper only discusses issues of the management of the patient test plans, which involves dynamic interactions among system components and/or clinicians. The management of the running patient test plan constitutes the following: 1) the dynamic querying of the test plan definition; 2) the dynamic querying of the test plan execution progress; 3) the dynamic addition, deletion, modification and replacement of the ECA rules making up the test plan's logic allowing dynamic adjustments and changes to be made to the plan; and 4) the maintenance of dynamic versions of the patient test plan.

5.3.1 Handling dynamic modification of patient test plans in TOPS

When change is being made to a test plan, only the test plan specification expressed in PLAN language is affected. The change will have to be *propagated* to the running instance and to the protocol. The modified plan specification is saved and the old version is archived for historical or version maintenance. Another option is to tightly couple rule changes with execution. Modification is made to the running version of the plan. Changes propagated to the test protocol are of the following types: 1) addition or deletion of base schedules, and 2) modification, deletion, addition of rules. Modification process has three phrases: a) the retrieval of the component that need to be modified; b) the editing of desired changes; and c) the installation of the changed component

To edit changes to an executing test plan, one of the following could be done: 1) Continue executing the older plan while editing, then freeze it when propagating the changes; 2) Freeze the execution and resume when changes have been completely propagated. (Freezing a plan or schedule means suspending its execution, which is to be resumed after a dynamic management operation is complete and effected. When a plan is frozen everything else including the base schedule and protocol rules are also frozen); and 3) Terminate the execution of the test plan. Only the plan can be frozen. The plan's individual components are never frozen but are only terminated. The plan is frozen only when the plan schedule is being added, modified and deleted. Deleting a plan or its component

can be done only when the corresponding currently executing plan is in a terminated (disabled) state.

Whatever approach is taken there is a point where a running test plan needs to be stopped. In TOPS, these three approaches are adopted depending on the size and potential impact of the modification as well as the modification operation being performed. Modifications that affect the whole plan or base schedule are sizeable enough to warrant either terminating the plan or freezing the plan or schedule. The scenarios for dynamic management of an executing test plan are summarised in Table 2.

Table 2. Scenarios for the dynamic management of test-ordering plans

Dynamic Operation	*Dynamic Manipulation*			
	Plan	**Schedule**	**Static Rule**	**Dynamic Rule**
Create New	Terminate current plan, Move patient to new category, Get new plan, Set up new plan for execution	Terminate current; If new schedule exist, then select new schedule; Else create new schedule; Set up schedule for execution; Update plan specification; If new schedule update protocol specification	Create/edit new rule Update schedule specification Set up rule for monitoring and execution	Create/edit new rule Update plan specification Set up rule for monitoring and execution
Delete	Terminate plan; Drop schedules and all rules	Freeze plan; Drop all static rules Create or select another schedule	Terminate rule Drop rule	Terminate rule Drop rule
Modify	(See modification of plan components: schedule, static rule and dynamic rule)	Freeze plan; Retrieve and modify static rules; Update plan specification; Set up schedule for execution; Unfreeze plan	Terminate current; Retrieve and modify rule; Update schedule specification; Set up rule for monitoring and execution	Terminate current; Retrieve and modify; Updated plan specification; Set up rule for monitoring and execution

5.4. Implementation of TOPS

TOPS is being implemented using Oracle on Windows 2000. The support for ECA rules (or triggers) in Oracle reduces the complexity of the patient test plan engine. An important requirement for the implementation is the support for dynamic adaptation and management of ECA rules, which enables the dynamic management of an executing patient test plan. Currently, it seems that within DBMSs there is a lack of comprehensive query and dynamic management facilities to higher level objects such as triggers. A TOPS chooses to implement the management and execution at two different levels. The management plane is outside the DBMS kernel to provide the comprehensive query and manipulation facilities to *test-ordering protocols* and *patient test-ordering plans*. The execution plane co-operates with the internal trigger mechanism, so as to dynamically perform the tasks specified in *patient test plans*.

The *Test Plan Manager (TOPlaM)* and the *Rule Engine(TOPEcaREng):* The plan rule engine is the reactive wrapper that extends the ECA rule functionality of the database system. Particularly, a component dealing with the complicated temporal aspects of a test-ordering protocol needs to be developed on top of the selected DBMS as it seems that no strong support is provided by active databases [6]. It would be really desirable if such temporal services could also be provided as an integrated part of DBMS triggers. This requirement has been identified in other ECA technology application domains such as network management and control systems, leading to the suggestion for the integration of active and temporal database concepts and technologies [8] [9] [10].

The Category Manager (PCaM) and Protocol manager (TOProM): The PCaM and the TOProM together make up an implementation of the static protocol specifications expressed in PLAN language. The most important aspect of these system modules is the functionality for mapping protocol specifications in the PLAN language into a database. Not too many high level difficulties are expected here as the general mapping mechanisms for automatically generating triggers from declarative language have been well studied and practiced. However, the organisation and manipulation of the generated test protocol still pose technical challenges.

6. Summary, Discussion and Conclusion

This paper has described an on-going project, which aims at developing an advanced active database application system for a computerised clinical test-ordering system (TOPS). A modelling framework and specification language, PLAN, serving the specification purpose, was briefed. The architecture of TOPS, its execution flow, and a management scenario for patient test-ordering plan were also discussed.

The main feature of the TOPS architecture is the extension of reactive capability supported by an underlying active database system in order to execute a patient test-ordering plan following the ECA rule paradigm. Special attention and emphasis is placed on the requirement for performing dynamic management of the ECA rules in order to adapt to the patient's dynamic clinical condition. Since the protocol specifications and the test plan rules are stored in the database in the form of relations, dynamic management, that is, querying,

insertion, deletion and update operations can be performed at a high level. This will, in turn, greatly enhance the clinical professionals' ability to efficiently and effectively *manage*, *query* and *manipulate* the computerised clinical protocols. Therefore a better healthcare service can be achieved. In the TOPS architecture, there are some technical issues that may need special attention.

Firstly, the management functionality of DBMSs on high-level database objects needs enhancement. For instance, although operations for creating, dropping or replacing of a named trigger are provided, to select a trigger based on its features seems not to be possible yet. This is an important desired feature of an active DBMS, at least to the project presented in this paper. The problem of a lack administration tools for triggers has also been raised from a different perspective in [11].

Secondly, although a trigger can be enabled or disabled, it may not be enough to meet the requirements that a test-ordering rule should 'sleep' for a period of time and then 'awaken' itself later.

Thirdly, it seems that currently a user can only add a trigger into the active DBMS manually. There is a need in our application for a trigger to be dynamically manipulated automatically within an application. For example, a test-ordering rule may want to add in a new, or even delete, a currently active test-ordering rule.

Finally, clinical protocols are not only event-driven, but they are also time-driven in some cases. It would be really desirable that an active database could also provide a trigger with comprehensive temporal features. The current TOPS approach is to have a temporal component outside the DBMS, which checks the temporal events and produces a temporal trigger. This is obviously not an ideal situation.

Although the application of ECA rules in dealing with the clinical test-ordering protocol is promising, this area is only a special sector of a huge healthcare area. The problem of how to deal with the general clinical protocols and guidelines, rather than the special laboratory test-ordering area (though still big enough to stand alone), still needs much more attention. The AI people have been working in this area for over two decades with strong decision-support orientation. It may now be the time for database people to contribute in dealing with the highly challenging issues of management, query and manipulation of the computerised healthcare information.

Acknowledgement

The authors would like to give acknowledgment to the Office of Postgraduate Studies and Research of the Dublin Institute of Technology who are sponsoring this work under the Strategic Research and Development Project Code 7985 9678.

7. References

[1] Grimson W, Berry D, Grimson J, Stephens G, Felton E, Given P and O'Moore R (1998). *Federated healthcare record server - the Synapses paradigm.* International Journal Medical Informatics, Elsevier Science, Vol.52, pp.3-27.

[2] Gordan C, Herbert I, Johnson P, Nicklin P, Pitty D and Reeves P (1997) *Telematics for Clinical Guidelines: A Conceptual Modelling Approach.* Medical Informatics Europe '97, IOS Press.

[3] Miksch S (1999). *Plan Management in the Medical Domain.* AI Communications, Vol.4.

[4] Musen M A (2000) *Two decades of Models and Components: Why has Automation of Guideline-Directed Care been so Difficult?* First European Workshop on Clinical Practice Guidelines, EWGLP 2000, Leipzig, Germany.

[5] Widom J and Ceri S, (eds.) (1996). *Active database systems: Triggers and Rules for Advanced Database Processing.* San Francisco, California: Morgan-Kaufmann.

[6] Ceri S, Cochrane R J and Widom J (2000) *Practical Applications of Triggers and Constraints: Successes and Lingering Issues* In proceedings of the 26 international conference on very large data bases. Cairo, Egypt. pp254-262.

[7] Wu B and Dube K. (2001) PLAN*: a Framework and Specification Language with an Event-Condition-Action (ECA) Mechanism for Clinical Test Request Protocols.* In Proceedings of the 34th Hawaii International Conference on System Sciences: the Mini-Track in Information Technology in Healthcare, Abstracts and CD-ROM of Full Papers, IEEE Computer Society, Los Alamitos, California, p.140

[8] Dittrich KR and Gatziu (1993). *Time issues in active database systems.* In: Snodgrass RT (ed) (1993), Proceedings of the International Workshop on an Infrastructure for Temporal Databases, Arlington, Texas, June 1993.

[9] Hasan MZ (1995*). An active temporal model for network management databases.* In: Proceedings of the IFIP/IEEE 4th International Symposium on Integrated network Management, May 1995, Santa Barbara, California. Chapman and Hall, London, 1995: pp.524-535.

[10] Pamamritham K, Sivasankaran R, Stankovic JA, Towsley DT and Xiong M (1996). *Integrating temporal, real-time and active databases.* SIGMOD Record ACM Special Interest Group on Management of Data, Vol.25 No.1: p.8-12, March 1996

[11] Simon E and Kotz-Dittrich A (1995). *Promises and realities of active database systems.* In: Proceedings of the 21st VLDB Conference, Zurich, Switzerland, 1995