To the memory of my late brother

Artwell Dube

ACKNOWLEDGEMENTS

This work could not have been undertaken without funding from the Office of Postgraduate Studies and Research of the Dublin Institute of Technology (DIT). Throughout the study period, I got invaluable assistance, support, guidance and encouragement from my research supervisor, Dr Bing Wu, to whom I'm grateful for accepting to be my research mentor. I would like to thank my research advisor Professor Jane Grimson for the support and advice she gave during the entire period of my research project. These two together with the MediLink Project members, especially Mr Bill Grimson and Dr Lucy Hedermann, provided an excellent research framework and conducive environment for my interaction with other researchers in Trinity College Dublin (TCD) and DIT. I am grateful to Ms Maev Maguire and Dr Fred Mtenzi for their encouragements throughout the study period; for taking a close and caring attention and interest in my progress and general welfare; and for their crucial representations in support of my work. I'm grateful to Professor Gordon Foster and his TRINET Project team especially Mr Victor Thorne and Mr Tom Fallon for bringing me to Ireland and to the DIT in August of 1999. I would also like to thank the National University of Science and Technology (NUST), Bulawayo, Zimbabwe, for providing funds for my initial travel to Ireland.

This work could not have been achieved without the constant support and loving attention from my wife, Anotida and my young brother, Tinashe, and sister-in-law, Anesu, helped in many ways towards our welfare in Dublin. My special thoughts go to my two lovely daughters, Makagonashe and Anokudzwa Tinotenda, who patiently and courageously endured our absence from home in Zimbabwe while studying in Ireland. I'm grateful to Mr Ezekiel Chiwowo and Mrs Elita Mazonde for looking after Maka and Ano, during our absence. Last but not least, I'm grateful to my parents, Mr Zakaria Mhaka and Mrs Martha Wanzirai Rukanda, my late brother Artwell, and my uncle, the late Mr Anos Mutangirwa, for teaching me, from an early age, to stay focused, to persevere, to endure and to continuously strive towards the attainment of any ideal that I set myself to attain.

Kudakwashe Dube

Hokowhitu, Palmerston North, New Zealand

06 February 2010

SELECTED PUBLICATIONS ASSOCIATED WITH THIS BOOK

1. **Dube, Kudakwashe** and Wu Bing (2009); *A generic Approach to Computer-Based Clinical Practice Guideline Management Using the ECA Rule Paradigm and Active Databases;* Int. J. Technology Management, Vol. 47, Nos. 1/2/3:75-95.
2. **Dube, Kudakwashe** and Bing Wu: *An Active Database Approach to Computerised Clinical Guideline Management.* Computer Supported Acitivity Coordination (CSAC) 2006: 107-115
3. **Dube, Kudakwashe**, Essam Mansour and Bing Wu: *Supporting Collaboration and Information Sharing in Computer-Based Clinical Guideline Management.* CBMS 2005: 232-237
4. **Dube, Kudakwashe** (2004), *"A generic approach to supporting the management of computerised clinical guidelines and protocols"* (2004). Doctoral. Paper 3. http://arrow.dit.ie/sciendoc/3
5. **Dube, Kudakwashe**, Bing Wu and Jane Grimson: *Framework and Architecture for the Management of Event-Condition-Action (ECA) Rule-Based Clinical Protocols.* CBMS 2002: 288-294
6. **Dube, Kudakwashe**, Bing Wu and Jane Grimson (2001); *Using ECA Rules in Database Systems to Support Clinical Protocols.* DEXA 2002: 226-235
7. **Dube, Kudakwashe** and Wu Bing (2001), *Supporting Clinical Laboratory Test-Ordering Protocol Specification, Execution and Management: an Event-Condition-Action Rule and Database Approach.* Healthcare Informatics Journal, SAGE Publications, Vol. 7, No. 1, 20-28 (2001).
8. **Dube, Kudakwashe** and Wu Bing (2001); *Specification, Execution and Management of Clinical Test-Ordering Protocols: a Database Approach.* In: Heller B, Loffler M, Musen M and Stefanelli M, Computer-Based Support for Clinical Guidelines and Protocols: Proc. 1st European Workshop on Clinical Practice Guidelines and Protocols (EWGLP2000),13-14th November, Leipzig, Germany, IOS Press, p.31-42, 2001.

ABBREVIATIONS

The following is a list of abbreviations that appear in this book.

24CRCL_PL	24 hour cretimine clearance and protein lost
ACE	Angiotsin Converting Enzyme Inhibitors are a group of pharmaceuticals that are used primarily in the treatment of arterial hypertension and congestive cardiac failure.
ACR	Albumin Creatinine Ratio, a clinical test used in the diagnosis and screening for the renal complications: albuminuria and proteinuria.
ADB	Active Database, a DBMS that incorporates the ECA rule paradigm in addition to the usual data and meta-data management functionality
ADBMS	Active Database Management System:- a DBMS that incoparates an active rule or ECA rule support mechanism.
ASCII	American Starndard Code for Information Interchange
ASTM	American Society for Testing and Materials
AUS	Annual Urine Screening which is applied to diabetes patients to monitor renal complications in diabetes patients. The aim of the screening is to detect these complications early and allow for early intervention, which has been established to reduce the resulting effects of these complications.
BNF	Backus-Naur Form: a formal sentax specification language developed by Backus and Naur
BP	Blood Pressure
CfMS	Careflow Management System
CGP	Clinical Guidelines and Protocols, which are statements systematically developed to guide the practicing clinician and the patient on how best to handle specific clinical problems (Field and Lohr 1992).
CMA	Confirmed Microalbuminuria: when microlbuminuria has been diagnosed, it is said to be confirmed. When a patient's ACR test result is found to be greater than 3.0 in two

	out of three tests performed within six months, microalbuminuria may be diagnosed and treatment may be initiated (Mogensen 2003).
CGPM	Conceptual Protocol and Guideline Model
DBMS	Database Management System.
DDO	Dirty-depedency Opearation Problem that occurs when an application or a client is trying to process an uncommitted event signals or messages. The LDO, DDO problems are found in both active database and distributed databases.
DFD	Data Flow Diagram
DUT	Dip-stick Urine Test, which is used to detect the presence of protein in urine. This test is used in the annual urine screening for diabetes patients.
ECA	Event-Condition-Action rule, a paradigm with the semantics that when an event occurs, check the condition and execute the action only if the condition is satisfied (Widom and Ceri 1996). The basic form of the ECA rule paradigm is supported in the form of triggers in modern database management system where events are database operations.
EHCR	Electronic Health Care Record, which is defined as "a structured multimedia collection of health-care data about an individual patient" (Grimson, J, Stephens et al. 2001).
EON	A component-based suite of models and software components for the creation of guideline-based applications
EPR	The Electric Patient Record, which has the same meaning as EHCR
GALEN	General Architecture for Languages and Encylopaeadias and Nomenclatures in medicine
GASTON	A methodology and a framework that facilitates the development and implementation of computer-interpretable guidelines and guideline-based decision support systems.
GAUDI	Guideline Authoring and Dissemination Tool
GLARE	Guideline, Acquisition, Representation and Execution
GLEAM	Guideline Editing and Authoring Model

GLIF	Guideline Interchange Format
GP	General Practioner
GRAIL	GALEN Representation and Intergration Language
GUIDE	A component-based multi-level architecture designed to integrate a formalized model of the medical knowledge contained in clinical guidelines and protocols with both workflow management systems and Electronic Patient Record technologies.
HbA1c	Haemoglobin (Hb) that type A, subtype 1c. This a specific type of haemoglobin A that results from the attachment of blood glucose molecules to its molecules. Diabetes patients have high levels of blood glucose and hence would experience high levels of HbA1c than non diabetics.
HL7	Health Level 7, an standards organisation whose mission is to provide a framework and protocol specifications for the exchange, storage, intergration and retrival of health information that support clinical practices and the management delivary and evaluation of health services.
HTTP	HyperText Transfer Protocol
ICU	Intensive Care Unit
IOM	Institute Of Medicine of the United States of America
LAS	Laboratory Advisor System
LDO problem	Loss-Dependependency Operation- a problem that occurs when signalled events or messages sent by an ECA rule in an active database to external applications may be lost or not acted upon by external applications.
LIS	Laboratory Information Systems
LUMPS	Liver Unit Management Protocol System
MAP	MicroAlbiminuria Protocol, which is a CGP for the management and treatment of microalbuminuria in diabetes patients.

MAS	MicroAlbuminuria Screening
MLM	Medical Logic Module, which is essentially an ECA rule specified by using the Arden Syntax and is a single software unit that is responsible for making a single medical decision (HL7 1999)
MonCooS	An acronym derived from **Mon**itoring, **Coo**rdination and **S**uggestion. The MonCooS approach is presented in this book as way to support the management of CGPs by allowing the specification, execution and manipulation of CGP knowledge and information to be performed in providing clinicians with automated assistance that focuses only on *monitoring* vital indicators, *coordinating* interventions and making *suggestions* as opposed to decisions, which are left to domain experts. The MonCooS approach tries to make effective use of the ECA rule paradigm in modern DBMS's.
MS SQL	The Microsoft SQL server, a relational database management system from Microsoft Corporation.
MUMPS	The Massachusetts (General Hospital) Utility Multi-Programming System, a computer language developed in the late 1960s and used predominantly in medical applications (Bowie and Barnett, 1976)
OODBMS	Object- Oriented Database Management System
OQL	Object Query Language
OS	Operating System
PLAN	Protocol LANguage originally proposed by Wu (1998) for specifying CGPs by following the ECA rule paradigm.
PRESTIGE (DILEMMA)	A project that was focused on the application of telematics technologies to support the dissemination and implementation of clinical practice guidelines and protocols.
PRODIGY	A computer-based decision support system (for prescribing in particular) that integrates with commercial primary care information systems in England. PRODIGY phase 3 incorporated support for chronic disease management.
PSE	Problem Scenario Entity

PSM	Problem Solving Method
PSO	Problem Scenario Object
RIM	Reference Information Model for healthcare applications developed and maintained by HL7
RuleML	Rule Markup Language (Boley et al, 2001)
SAMOS	The Swiss Active Mechanism-based Object-Oriented Database System: An active database system prototype constracted as a wrapper to the passive ObjectStore object-oriented DBMS.
SCR	Serum Creatinine Ratio, a test used in monitoring glycaemia with the purpose of optimising it.
SIEGFRIED	System for Interactive Electronic Guidelines with Feedback and Resource for Instructional and Educational Development
SpEM	An acronym derived from **Sp**ecification, **E**xecution and **M**anipulation. SpEM is a framework introduced in this book for supporting the specification, execution and manipulation of CGPs.
SQL	The Structure Query Language for manipulating data in relational database systems.
TOPS	Test Ordering Protocol System, a prototype system presented in this book.
TOPSQL	The TOPS Query Language, a high level declarative query language for manipulating CGP information and knowledge in TOPS.
UAE	Urine Albumin Excretion
UML	Universal Modelling Language a modelling language defined and maintained by Object Management Group
UTI	Urinary Tract Infection: In diabetes patients, laboratory tests need to be performed in oder to detect urinary tract infections during the annual urine screening performed in diabetes patients

WDL	Work flow Definition Language
XML	Extensible Mark Up Language
XRML	eXtensible Rule Markup Language (Lee and Sohn,2003)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PART I: INTRODUCTION AND BACKGROUND

Chapter 1 Introduction

The cost of clinical laboratory testing has been increasing considerably from year to year during the past four decades (van Walraven and Naylor 1998). Since the 1980's, healthcare organisations have been pressurised to control clinical laboratory utilization without affecting quality of patient care (Grossman 1983; Eisenberg 1985; Peters, M and Broughton 1993; O'Moore, Groth et al. 1996). It has been established that the use of *clinical test ordering protocols* supported by Information Technology can enhance quality, efficacy and proper usage of clinical laboratory resources (Matimer, McCauley et al. 1992; O'Moore, Groth et al. 1996; Bates, Kuperman et al. 1999; van Wijk, M .A. M., Bohnen et al. 1999; van Wijk, M.A.M., Mosseveld et al. 1999) and promote best practise in the clinical laboratory environment (Boran, O'Moore et al. 1996; O'Moore, Groth et al. 1996; Bates, Kuperman et al. 1999; Berry, Wu et al. 1999).

Clinical test ordering protocols are systematically developed statements, usually in natural language, that provide guidance on what clinical laboratory tests clinicians should order, what clinical laboratories should do in response to a test order, and what laboratories, clinicians and patients should do in response to test results in certain clinical circumstances. *Clinical Guidelines and Protocols* (CGPs), are statements that express medical knowledge for guiding patients and clinicians in making decisions about appropriate healthcare for the specific clinical circumstance of the patient (Field and Lohr 1992). *Clinical test ordering protocols* are a type of *clinical guidelines and protocols*. CGPs exist mainly as paper-based natural language statements, but are increasingly being computerised.

The application of modern Information Technology offers the potential to facilitate the incorporation of clinical guidelines, such as clinical test-ordering protocols, into the routine used daily by clinicians with the aim of improving patient care quality and optimising clinical resource utilisation. Supporting computerised CGPs in a healthcare environment so that they are incorporated into the routine used daily by clinicians is complex and presents major computing and information management challenges. This book a study that investigated this challenge. The main argument in this book is that the management of computerised CGPs should incorporate their *manipulation* (operations and queries), in addition to their *specification* and *execution,* as part of a single unified management framework.

A key feature of this book is that modern advanced database technology is applied to the task of managing computerised CGPs. The event-condition-action (ECA) rule paradigm and its implementation in active databases is recognised to have a huge potential in supporting computerised CGPs. The event-condition-action (ECA) rule paradigm represents a proven, simple, useful, practical and easily deployable approach to the computerisation and management of medical knowledge and hence the computerisation of CGPs. (ECA) rules are specified by an *event,* a *condition* and an *action* whose combined behaviour is such that the event must occur in order for the action to be executed subject to the condition evaluating to true (Widom and Ceri 1996). The ECA rule paradigm has been used to specify medical knowledge before and proved to be promising in supporting standardisation and sharability of the resulting knowledge modules (Hripscak, Luderman et al. 1994; HL7 1999). While the ECA rule paradigm has been used in computer-based clinical alerts and reminders for many years, its further development, integration and enhancement with other CGP representational formalisms and computational methods has received only limited attention in the support for computerised clinical practice guidelines. The Arden Syntax for Medical logic Modules (MLMs) (Sailors, Bradshaw et al. 1998) pioneering the application of the ECA rule paradigm in computerised clinical alerts, reminders and monitoring systems. HyperCare (Caironi, Portoni et al. 1997) claims to be the pioneering work in the use of the ECA rule paradigm within active databases to implement a specific CGP. However, HyperCare had the limitation that it did not provide a generic method that can be used to computerise other clinical guidelines, instead, it was specific to a particular guideline for which is did not provide a flexible specification model and language and a comprehensive manipulation framework for querying and performing operations on the computerised CGP.

As noted earlier, the ECA rule paradigm provides the means to specify knowledge required to support functionality such as monitoring and coordination in situations that require a timely response. Active Databases combine the ECA rule paradigm with the data management functionality of a database management system (DBMS) (Dittrich, Gatziu et al. 1995) to present a promising environment for supporting CGPs as well as the electronic medical record and clinical workflow. As noted earlier, up till this work, only one limited effort directed at harnessing active database technology for supporting CGPs is known to the author (Caironi, Portoni et al. 1997). No attention has yet been paid towards developing a unified framework that incorporates a generic way to combining the ECA rule paradigm and

active databases and, possibly, other formalism and computational methods, to provide support for the full-scale management of CGPs.

On one hand, the ECA rule paradigm and active databases have been thoroughly investigated and their theoretical foundations are now well known. On the other hand, the support for management of the ECA rules exists only in very limited form, e.g., database triggers, within modern systems. There is a need to demonstrate the practical requirement for a comprehensive ECA rule paradigm support in modern systems so that important real-life application domains such as healthcare could benefit.

This chapter introduces the study by first presenting, in Section 1.1, the problem under investigation in terms of the statement of the research question, the study hypothesis and, finally, the method of evaluation of the solution to the research problem. The research question is presented from both a general perspective and the perspective of the application domain. In Section 1.2 the aims and objectives of the study are presented. In Section 1.3, an outline of the methodologies to be used are outlined. Section 1.4 presents, the contributions of this work. Finally, the organisation of this book is presented in Section 1.6.

1.1. Managing Computerised Clinical Guidelines

Clinical Guidelines and Protocols (CGPs) are a special type of complex domain knowledge. The problem of how to efficiently and effectively manage computer-based CGPs has continued to pose a major challenge to the computing domain. The ECA rule paradigm has been proven to be effective in supporting the specification of medical knowledge (Hripscak, Luderman et al. 1994; HL7 1999). The ECA rule paradigm and active databases have also been used successfully in applications that require data management as well as monitoring and coordination. Such applications include workflow support (Eder, Groiss et al. 1994; Tagg and Lelatanavit 1998) and computer-aided manufacturing (Berndtsson, M. 1994). Thus, the ECA rule paradigm and active databases offer a potential solution to addressing the challenges posed by the computerisation of CGP management.

At a general level, this study addresses the question of using the ECA rule paradigm within the context of database systems in providing a generic and simple way to manage information in a complex application domain that has several important requirements. First, the domain information and knowledge need to be specified and later customised, using current values of the problem attributes, in order to be applied to a specific instance of the

problem scenario or case. Second, constant monitoring of domain situations is required with a provision for timely reaction to situations of interest. Third, the dynamic or on-the-fly manipulation and querying of domain information is required for complex objects and processes associated with these objects in the domain.

In addressing this general question the study focuses on a important application in healthcare - *supporting the management of computerised clinical guidelines/protocols (CGPs)* - and seeks answers to the two specific question. First, h*ow can the full-scale manageability of information for the complex domain of supporting computerised CGPs be supported?* In answering this question, the study tackles the following specific issues and questions:

1. Identification of the component aspects of the full-scale management of a CGP: What are the component aspects of the management of CGPs?
2. Formal specification of CGPs: How can we formally specify CGPs?
3. Storage of CGP in a way that enables them to be fully managed: How can we store CGP specification in a way that allows them to be subject to manipulation operations and queries?
4. Customisation of CGP specifications to suit specific needs and situations: How can we customise a CGP specification to suit specific clinical situations?
5. Instantiation and execution of a customised CGP: How can we execute a specified CGP by using a computer?
6. Performing on-the-fly manipulation operations on and issuing queries against both CGP specifications and their executing instances: CGPs and their executing instances both need to be managed. How can this be achieved?
7. A case study for supporting a real computerised clinical protocol for a specific clinical problem. Are the methods we develop applicable to a real protocol?

The second aspect of the research problem deals with the question: *How can we use the ECA rule paradigm supported within modern database management systems (DBMS's) as a core concept of the domain information modelling and enforcement frameworks for supporting the full manageability of computerised CGPs?* In answering this question, the study addresses the following issues:

1. Using the ECA rule paradigm in the modelling and specification framework for computerised CGPs;

2. development of a generic mechanism that is based on the ECA rule paradigm to execute CGPs;
3. Using the modular nature of the ECA rule paradigm as a basis for the customisation of CGP specifications in order to suit individual patients;
4. Exploit the ECA rule mechanism of a modern DBMS, such as Oracle9i, as an engine to support CGP execution and manipulation, i.e., performing operations and issuing queries;
5. Identify the limitations of the modern DBMS, if they exist, in supporting ECA rule paradigm-based applications.

The specific focus of this study is on solving the problem of providing a comprehensive and flexible environment for the full *management* of clinical protocol definitions and the process of their enforcement for each patient. Emphasis is placed on the efficient and effective *management* of the information and knowledge that is associated with the computerised clinical test ordering protocols. The main component parts of the problem are: the specification; the provision for persistence or storage; the automated enforcement or execution; and the manipulation, i.e., performing operations and querying, of the domain information associated with the clinical test ordering protocols.

The study's hypothesis is that the ECA paradigm supported within database systems could be an effective and practical tool for supporting important aspects of the management of complex domain information when used as a core concept within the domain knowledge model and its implementation. A further hypobook is that the use of the ECA rule paradigm in the active database environment would make it possible to automatically support the dimension of manipulation of information associated with CGPs.

The study will demonstrate its solution to the problem under investigation by focusing on the effectiveness of the developed framework, approach and mechanism in allowing domain information, within the context of clinical test-ordering protocols, to be specified using a declarative ECA rule paradigm-based language; executed using an ECA or trigger mechanism in a modern database system; and manipulated using a declarative query language. The main challenge is to show that the management of domain knowledge and information can be supported and managed easily. A prototype system will be developed to demonstrate the feasibility of the framework and approach developed. The prototype system will be evaluated in a case study that will be undertaken in consultation with clinical domain experts at St. James's Hospital and in the inter- disciplinary research group within the MediLink Project.

1.2. Aim and Objectives

The aim of this study is to investigate how to manage domain information in the provision of assistance to healthcare professionals, in ordering correct, appropriate and timely interventions, according to a set clinical guideline or protocol. An example of a clinical intervention of interest to this study are clinical laboratory investigations, which need to be performed on a patient. Clinical orders need to be made at the appropriate time and place, with prompt notification of results. Furthermore, it is important to provide for patient-specific recommendations, alarms and alerts. In an environment that allows dynamic adaptation and modification of the regime, important aspect of this aim is to provide monitoring and coordination without expropriating the task of reasoning from the domain experts. As has already been pointed out it is proposed that the *event-condition-action (ECA) rule* paradigm in the context of *active databases* is a promising technology that could be harnessed to effectively achieve this aim. Consequently, this also incorporates using the ECA rule paradigm and active databases in developing a generic framework and approach with specification and manipulation languages, and a software mechanism for the specification, storage, execution and manipulation of clinical protocols.

The main objective of this study is to develop a generic way for specifying, storing, executing and manipulating clinical guidelines or protocols knowledge and information from both the static and dynamic standpoints. Of interest to this study is the provision of the functionality that allows clinicians to perform operations and query both the static and dynamic aspects of the guidelines or protocols within the system. The specific objectives are as follows:

a) To develop a generic framework and approach for managing domain information in the form of clinical protocols;
b) To enhance the design of the language, PLAN, for specifying clinical test ordering protocols. PLAN was initially proposed by Wu (1998) as a declarative specification language that follows the ECA rule paradigm;
c) To develop a declarative operator and query language for the manipulation of test ordering protocols;
d) To develop translators for the specification and manipulation languages;
e) To develop a software mechanism to support the management of the domain information associated with clinical protocol definitions and enforcements;

f) To design and implement a prototype system for the full-scale management of domain information using a case study involving the support for clinical test-ordering protocols for the diagnosis and management of micro-albuminuria in patients with diabetes mellitus; and
g) To test and evaluate the prototype system, together with the underlying frameworks, concepts and methods, in the care of patients with assistance from medical experts at a local hospital.

1.3. Methodology

To establish the state-of-the-art, a literature review was conducted. The literature review framework was designed in close attention to the aims and objectives of the research.

In order to comprehensively address the problem under investigation, use is made of a unified framework in which the CGP management problem is broken down into core components. *Modularisation* (Parnas 1972) and the principle of *separation of concerns* (Lopes and Hursch 1995) are used to ensure both the independence and co-operation/collaboration among components within the framework.

The event-condition-action (ECA) rule paradigm (Dittrich, Gatziu et al. 1995; Widom and Ceri 1996) is used as a basis for modelling the domain information and for implementing the enforcement mechanism that applies the domain information to the real world scenarios. The *Object-orientated paradigm* (Rumbaugh, JR, Blaha et al. 1990; Booch 1993) is used as the intermediate model for CGP information between the specification, enforcement and manipulation mechanisms on the one hand and the storage mechanism on the other.

The involvement of the actual decision-making at the operational level will be fostered through external interaction and communication in which the clinician absolutely dominates and dictates while the system only suggests, prompts and alerts. Artificial Intelligence methods that involve complex automatic reasoning or automatic derivation and enforcement of domain knowledge are not employed.

In the task of enhancing and implementing the language, PLAN, use is made of well-established classical techniques and tools for designing formal languages. The Backus-Nuar Form (BNF) is used to specify PLAN as a high-level declarative language for allowing domain knowledge to be: a) declaratively specified, b) easily manipulated and c) declaratively queried. The parser for PLAN is developed from the principles of recursive

descent parsers. Instead of using language translation techniques, to handle the parser outputs, an object-oriented mechanism is used to translate the parser output into the database model.

The Unified Modelling Language (UML) techniques and modelling tools (Rumbaugh, J, Jacobson et al. 1998; OMG 2001) are used to design software modules. To model CGPs, UML state charts are used in such a way as to facilitate the involvement of clinical domain experts. Entity-Relationship modelling (Chen 1976) and relational database design techniques (Elmasri and Navathe 2000; Ullman and Widom 2001) are used to design the database.

Consultations on medical aspects of this Study were conducted with medical domain experts at Tallaght and St James's Hospitals in Dublin.

The evaluation of the solution to the problem is attained by: the development of a prototype system; and the testing and evaluation of the prototype system, which is conducted both theoretically and through a practical demonstration aimed at soliciting feedback from clinical domain experts using patient scenarios from St James's Hospital.

1.4. Contributions

The main contribution of this book is a generic framework and approach for the management of information and knowledge for supporting the management of computerised CGPs. Further contributions of this research are:

a) A characterisation of the problem of managing CGP information as consisting of the three generic planes of specification, enforcement/execution and manipulation, with each plane having its own levels of abstraction and interacting, in a dynamic fashion, with the other two planes.
b) A generic software mechanism for supporting the framework and approach for managing CGP knowledge and information. This software mechanism is based on the ECA rule paradigm within the context of database systems and lays the groundwork for easy integration of the CGP support mechanisms within the electronic healthcare record (EHCR) (Grimson, W, Berry et al. 1998; Grimson, J, Stephens et al. 2001) and clinical workflow.
c) An approach to the use of the ECA rule paradigm for both conceptual modelling and implementation of CGP management within a unified framework by using a ECA rule

mechanism of a modern DBMS. This approach creates a basis for the demonstration of the ECA rule paradigm as a viable technology for supporting real applications (particularly the management of CGPs), thus, pointing to the need for further enhanced support in modern DBMS;

d) A prototype system, TOPS, for supporting the management of CGPs for clinical test-ordering, which supports the framework and approach developed in this study by making use of a declarative specification language to specify protocols, an ECA rule mechanism of a modern DBMS and its extension as the execution engine, and a query and manipulation language to query and manipulate domain information and knowledge, in the form of CGPs and patient data, within the system; and
e) A case study for the management of a computerised protocol for microalbuminuria in diabetes patients.

1.5. Book Organisation

Figure 1 illustrates the structure and organisation of this Book. This Book consists of four major parts. **Part I** describes the problem under investigation. The context of the problem is also set. The background to the problem is presented in the form of a review of the state-of-the-art in the support for CGP management and in the applications of active database systems and the ECA rule paradigm. The later is presented with a view to harnessing for supporting CGP management. Part I consists of chapters 1 to 4 as illustrated in Figure 1. In particular, Part I serves to outline current trends in the domains under investigation, presents the motivation for this research work, states the problem, aim, objectives and methodology adopted and, finally, details the contributions of this work. This part also exposes the context and the background to the problem being investigated through a review of the literature. The literature review is two-pronged: first, a review of current practice in supporting the management of clinical guidelines and protocols is undertaken; and, second, a review of the applications of the event-condition-action (ECA) rule

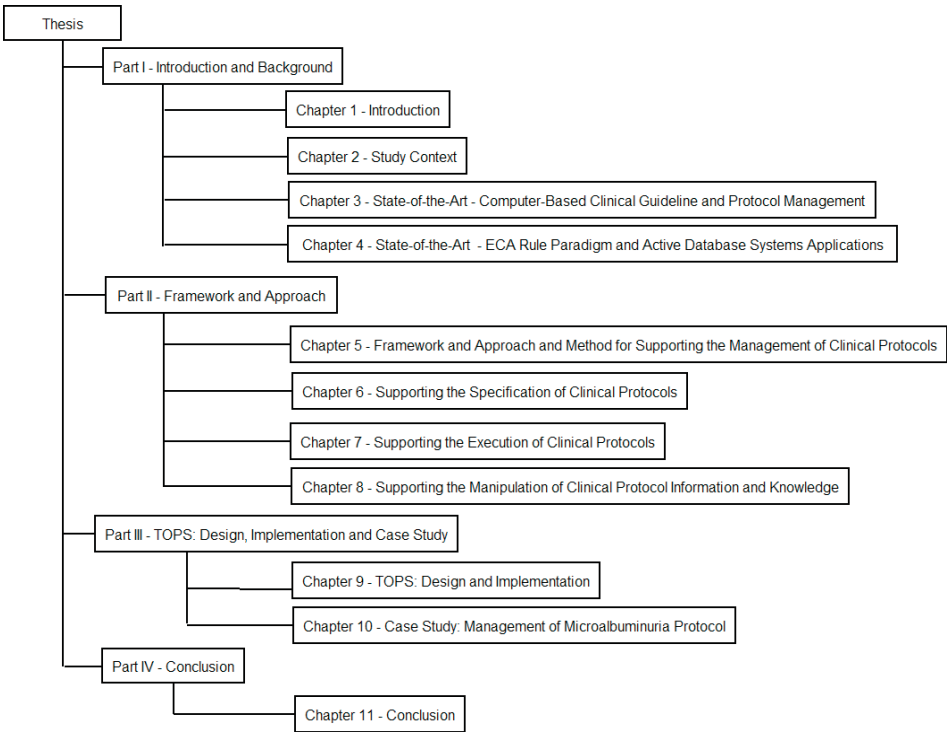


Figure 1 Organisation of this book

paradigm and active database systems is presented with a view towards harnessing the ECA rule paradigm for supporting the management of clinical guidelines and protocols (CGPs). Part I is organised as follows: Chapter 1 presents an introduction to the study; Chapter 2 defines the context of the problem that has been investigated and the review of the state-of-the-art is presented in two parts: Chapter 3 reviews the computer-based management support for clinical guidelines; and Chapters 4 reviews the (ECA) rule paradigm and active systems technology and their applications in general as well as in the supporting the management of clinical guidelines and protocols.

Part II presents the framework and approach, which resulted from this study, for managing clinical protocols. This part also discusses, in depth, the approach and methods developed in this Study for supporting the specification, execution and manipulation of information and knowledge for clinical protocol management. Part II consists of chapters 5 to 8 as illustrated in Figure 1. Part II first reviews the unified and generic framework, SpEM (**Sp**ecification, **E**xecution and **M**anipulation), for the management of Clinical Guidelines and Protocol (CGP) knowledge. The approach, called MonCooS (**Mon**itoring, **Coo**rdination and **S**uggestion), incorporating the method and mechanism for computer-based management of information and knowledge for supporting CGPs in the healthcare domain, is then presented.

A specification language, PLAN (**P**rotocol **LAN**guage is also presented. In the MonCooS approach, the event-condition-action (ECA) rule paradigm and active database technology are used as the basis for the specification and execution within the unified SpEM framework. The rest of this part is organised as follows: Chapter 5 presents the framework and approach for supporting the management of computerised clinical protocols. The next chapters then give a more in-depth treatment of the three main aspects of the problem; Chapter 6 presents PLAN, a declarative protocol specification language that follows the ECA rule paradigm and the PLAN specification model for specifying and storing clinical protocols; Chapter 7 presents a generic approach and mechanism for executing formally specified clinical protocols; and, finally, Chapter 8 presents the approach and method for the manipulation of the information associated with the protocol specifications and protocol executing instances.

Part III presents the design and implementation of the prototype system, TOPS, and the case study in which TOPS is used in the management of the microalbuminuria protocol for diabetes patients. Part III consists of chapters 9 and 10 as illustrated in Figure 1. The aim of Part III is to present the design and evaluation of the approach and framework through the implementation of the prototype system, TOPS, the **T**est-**O**rdering **P**rotocol **S**ystem and a case study that uses TOPS to manage a clinical protocol. TOPS provides assistance in the management of clinical protocols for the domain of clinical laboratory test-ordering. TOPS implements the SpEM framework and the MonCooS approach, which have been presented in Part II. The case study presented deals with the management of the micro-albuminuria protocol for patients with diabetes mellitus. This part is organised as follows: Chapter 9 presents the requirements for TOPS, the design of TOPS that meet the specified requirements; and Chapter 10 presents a demonstration of the applicability of the framework, approach and method presented in Chapters 5-8. This demonstration is undertaken by using the microalbuminuria protocol developed with the help of clinical domain expert at St. James's Hospital in Dublin.

Part IV presents a review of this book and a conclusion. Part IV consists of chapter 11 as illustrated in Figure 1 and concludes this book. Part IV reviews the research challenge or problem that was addressed by this study. A review of each chapter in this book is presented. The contributions made by this study are summarised. The benefits of the outcomes of this study are outlined. The pointers to future directions arising from this study are presented. Finally, a statement on the limitations of the evaluation of the study outcomes is given.

1.6. Summary

This chapter has introduced the problem under investigation. It presented the motivation for this research from the perspectives of both the clinical domain and active systems applications. The aims and objectives were discussed and the methodology outlined. The chapter also identified the contributions to knowledge made by this research work. Finally, the chapter described the organisation of this book.

Chapter 2 Study Context

2.1. Introduction

Attempts to reduce costs and practice variation and optimise resource utilisation in healthcare have led to the formalisation of medical domain information and knowledge, acquired through experience and medical research, to create clinical guidelines and protocols (Field and Lohr 1992). The event-condition-action (ECA) rule paradigm, as found in active databases (Dittrich, Gatziu et al. 1995) and originating from production rules (Newell and Simon 1972) in traditional expert systems, promises to be an effective means of representing, sharing, enforcing and manipulating information and knowledge. The ECA rule paradigm in active database systems (Dittrich, Gatziu et al. 1995) could be used to provide an excellent framework for facilitating the solution to the problem of the integration of clinical guideline, patient record and clinical workflow systems. This book consentrates on the problem of supporting the management of clinical protocols, with focus on clinical laboratory test-ordering protocols. The aim of the investigation is to develop a generic approach that makes use of the ECA rule paradigm in active database systems within a unified modelling and implementation framework for supporting computerised CGPs. This Chapter sets the context by first presenting, in Section 2.2, some definitions of the main concepts and terms as used in this Book. The main aspects of the research are then set into the context of the clinical guideline domain in Section 2.3, clinical test-ordering protocols in Section 2.4 and the ECA rule-based support for clinical guidelines in Section 2.5. Finally, Section 2.6 presents a discussion and summary of this Chapter.

2.2. Definitions of Terms and Concepts

It is important to clarify a number of key concepts in accordance with their use in this book. These concepts are presented in the next paragraphs.

Clinical Guideline: The American Institute of Medicine defines a *clinical guideline* as: "… a set of systematically developed statements to assist the medical practitioner and the patient in making decisions about appropriate healthcare for specific clinical

circumstances."(Institute of Medicine (IOM) 1992) . The following is an analysis of this definition:

- *"systematically developed":* The development of clinical guidelines involves an orderly and lengthy process that takes into consideration recent scientific knowledge, experiential evidence, consensus among healthcare experts and current practice.
- *"assist the medical practitioner and patient"* : Guidelines are not meant to be compulsory but to uphold the domain expert's dominance and discretionary rights, i.e., they are meant to assist not dictate to the clinician and the patient, who have a right to override them when necessary.
- *"making decisions":* medical decision-making is the primary task of clinicians. Patients make decisions about their own health. Patients also have the final say in major decisions on what is done to them by clinicians during the process of care. Clinical guidelines help clinicians and patients to make informed decisions with regard to the appropriate care for the patient.
- *"appropriate healthcare":* All medical decisions made by the clinician and the patient are aimed at achieving the best patient outcomes in an effective and efficient way. Consequently, appropriate healthcare is patient care that leads to the attainment of this aim.
- *"specific clinical circumstances":* each clinical guideline that is developed deals with a specific clinical problem. However, they do not take into consideration the specific circumstances of an individual patient. It is the task of the clinician to put the guideline knowledge and advice into the specific context of the patient.

Clinical guidelines can also be viewed as "knowledge models of preferred processes of care" (OpenClinical 2001). The guidelines need to be locally adapted to be applicable to the local patient and disease scenarios, since while medical knowledge is universal, clinical practice is local (Nykanen 2000). A clinical guideline can be combined with the organisational model in order to harness workflow technologies to create a *care flow* (Quaglini, S., Stefanelli et al. 2000b) environment for dissemination, medical knowledge utilisation and healthcare team communication and coordination.

Clinical guidelines encode domain knowledge and need to be managed in order to be useful. Therefore, the incorporation of clinical guidelines into the routine used by the clinicians can be seen as a domain knowledge management task. This work investigates the

support for the management of computerised clinical guidelines. Supporting computerised clinical guideline management involves formally representing medical knowledge and assisting clinicians by using information technology to make this knowledge available for use during decision-making and by performing routine tasks that are amenable to computerisation.

Clinical protocol: The main difference between a clinical guideline and a clinical protocol is that a clinical guideline is clinical or medical knowledge that is context-insensitive while a clinical protocol is context-sensitive because it is clinical or medical knowledge incorporated into daily routine and is derived from customising and enhancing the guideline with localised and patient-specific detail. This is why Miksch (1999) views a clinical protocol as a highly detailed clinical guideline, which, she states, is usually mandatory. In essence, a clinical protocol, just like a clinical guidelines, encapsulate knowledge about medical concepts and knowledge about how to carry out specific activities (Gordon, Herbert et al. 1997). Consequently, the terms "clinical guideline" and "clinical protocol" refer to the same basic concept and, in this book, may be used interchangeably.

Computerised Clinical Practice Guidelines (CGPs) and Protocols: Clinical guidelines or protocols generally exist as human expertise, organisational custom and paper or text-based publications. They are meant to be read by clinicians who are expected to apply the knowledge contained in the guidelines to clinical problems that they encounter during their daily practice. When clinical guidelines or protocols are formally specified and enforced by using appropriate computational techniques implemented in a computerised mechanism, they are then referred to as *computerised clinical guidelines or protocols.* This book is concerned mainly with computerised clinical guidelines or protocols.

Clinical Test-Ordering Protocol: Clinical laboratories and clinicians use *clinical test-ordering protocols* to define: what tests clinicians should order; what laboratories should do in response to an order; and what both laboratories and clinicians should do in response to test results in certain clinical circumstances. These protocols may be incomplete, informal, unwritten and tend to represent the experiences and wishes of senior medical and administrative staff (Peters, M, Broughton et al. 1991). The differences between protocols and the difficulty in enforcing them result in variations in clinicians' utilisation of clinical laboratory services and in the clinical laboratories' responses to test orders. This problem can be resolved by defining consensus protocols, which Peters et al (Peters, M., Clarke et al.

1991) refer to as *locally agreed protocols*, which can be enforced with the support of a computerised system.

Clinical Guideline or Protocol Management: In this book, the term *management* of clinical guidelines or protocols refers to the following aspects:

Specification: This involves the formal representation of the clinical guideline knowledge by using a model and a language in order to allow the guideline knowledge to be stored and manipulated by computer-based methods.

Execution or enforcement: This is the computer-based application of the formal guideline or protocol specification to the solution of a clinical problem. This book will take guideline execution and guideline enforcement to refer to the same concept – the computer execution of a computerised clinical guideline or protocol with respect to a patient. The issues of a clinician's compliance to clinical guidelines or protocols and the methods by which this can be achieved are outside the scope of this book. Guideline or protocol execution will be achieved through a computer-based mechanism. The guideline execution or enforcement mechanism involves collaboration between human agents, the clinician and the patient, on the one hand, and a software mechanism, on the other.

Manipulation: the manipulation of the clinical guidelines knowledge and information through use of *operators* and issuing of *queries* as well as *sharing* the guidelines knowledge among healthcare professionals and organisations. Operators and queries are performed on both the static and dynamic aspects of the clinical guidelines knowledge as well as their specifications and instances. Sharing of clinical guidelines consist of two aspects: the *customisation* of the generic clinical guidelines to suit local situations and the *dissemination* of the guidelines to the healthcare professionals and/or organisations.

The Event-Condition-Action Rule Paradigm: An ECA rule consists of *events*, *conditions* and *actions* whose combined semantics mean that when the event occurs, the condition is evaluated and, if it evaluates to true, then the action is executed (Gatziu, Geppert et al. 1991). Thus, each ECA rule consists of three components:

- An event part, containing a so-called *transition predicate* that lists all possible events which are of concern to the rule;
- A condition part, which can be an arbitrary predicate, and
- An action part, which is an arbitrary list of executable functions.

The event and the condition together constitute a *situation* that the rule has to monitor. Situation monitoring involves detecting an event of interest and evaluating a condition associated with the event. The situation is said to have occurred only if the event has been detected and the condition evaluates to true. The action is performed only if the situation has occurred (Dittrich, Gatziu et al. 1995). Characteristics of ECA rules and their collective behaviour in both relational and object-oriented database systems have been analysed by various researchers in the area of *active databases* and are now well known (Paton and Diaz 1999). In clinical guidelines, events are detectable happenings that occur to a patient and range from disease progression to what clinicians do to a patient; conditions are checks on patient clinical attributes that are made based on clinical laboratory measurements and observations; and actions are clinical interventions that are triggered by occurrences of events or conditions or both and can generate events and/or give rise to satisfaction of conditions. Consequently, the ECA rule paradigm contains the compositional primitives for clinical guidelines.

2.3. Clinical Guideline and Protocols

Tu et al. (1999) have characterised the clinical guideline domain as consisting of health-care providers, patients, and the decision support systems. These multiple agents interact at different time points, called *encounters* (Tu, S. W. and Musen 1999), which may simply be times when a monitoring system detects the arrival of new data. At each encounter the following three things may happen: observations about the patient are recorded; decisions are made; and actions are carried out (Tu, S. W. and Musen 1999). It is also possible for healthcare providers and patients to take actions outside encounters (Tu, S. W. and Musen 1999) but this may still be within the context of the guideline or may mean that both the patient and the clinician are exercising their discretion. The rationale for introducing clinical guidelines and protocols is to reduce unjustified variations in clinical practice, improve healthcare quality and contain costs (Grimshaw and Russell 1993). Clinicians need to be made aware of the guidelines. They also need to be encouraged to comply with the guidelines during routine practice.

Studies have established that clinician compliance to guidelines is improved when the guidelines are presented to them at the point of care when they are treating the patient and also accessing the patient's record (Grimshaw and Russell 1993; Tu, S.W. and Musen 2001).

The presentation of the guidelines and the point of care must not be intrusive. An examination of a variety of clinical guidelines by Tu et al. (Tu, S. W. and Musen 2000) led to the abstraction of a set of the following generic guideline tasks: decision-making; setting goals; work specification; and interpretation of data. Decision-making is the main tasks for guidelines as highlighted in the definition by the Institute of Medicine (1992). The following are two classes of clinical guidelines that are based on the distinction between the notions of time points and timeline: *consultation guidelines,* which specify guideline tasks whose consequences are not being tracked over time; and *management guidelines*, which model guideline tasks that lead to dependent changes in patient states over time (Tu, S. W. and Musen 1999).

Studies have also established that when clinical decision support systems are developed to provide, at the point-of-care, patient-specific assistance in decision-making and integrated with clinical workflow, they can improve clinicians' compliance with clinical guidelines and hence patient outcomes (Grimshaw and Russell 1993; Lobach and Hammond 1994). The development of computer-based management strategies to implement clinical guideline-based decision-support systems has become a critical issue in promoting the use of clinical guidelines in daily practice (Nykanen 2000).

During the 1980's, the healthcare community has paid more attention to guideline development than to guideline implementation for routine use in clinical settings (Audet, Greenfield et al. 1990). In the 1990's and 2000's, this situation improved significantly as a number of guideline systems have emerged during this period (Wang, Peleg et al. 2002), for example EON (Musen, M.A. , Tu et al. 1996), Asbru (Shahar, Miksch et al. 1998), Pro*forma* (Fox, Johns et al. 1998) and PRESTIGE (Gordon and Veloso 1996).

2.4. Guidelines and Protocols for Ordering Clinical Laboratory Tests

As noted earlier in Chapter 1, the unit cost of performing a single clinical laboratory test had decreased relative to inflation during the 1990's (van Walraven and Naylor 1998). In the same period, the number of tests ordered had increased dramatically(van Walraven and Naylor 1998). As a result, the cost of clinical laboratory testing has increased considerably (van Walraven and Naylor 1998). The scenario has continued unabated into the first decade of the 2000's. This has prompted the introduction of research and initiatives aimed at

controlling clinical laboratory utilisation without adversely affecting the continued improvement of the quality of patient care. The initiatives that have been introduced include feedback, participation, education, cost awareness, financial incentives, penalties or risk-sharing, administrative change and rationing (Grossman 1983; Eisenberg 1985; Peters, M, Broughton et al. 1991). One of the most effective and proven approach to clinical laboratory utilisation management is the use of clinical test ordering protocols, which are mandatory clinical practice guidelines (Grimshaw and Russell 1993).

Table 2.1 A test-ordering protocol for Viral Hepatitis (in natural language) (Protocol Steering Committee 1998)

Suspected Condition (Please, write on requisition)	**Laboratory Test(s) Performed**
Acute Hepatitis	Inti-HAV IgM → if *positive*, no further testing required → if *negative*, test for: *HBsAg** → if *positive,* further testing only on request → if *negative*, test for *anti-HCV**
Hepatitis B Carrier	*HBsAg*
Previous/Chronic Hepatitis	*Anti-HBc* (total) → if *positive,* test for *anti-HBs**, and *HBsAg** **and** *Anti-HCV*
**Tests can be added automatically*	

Test-ordering protocols are generally available to clinicians in natural language form in the medium of paper or electronic text on the Internet. Table 2.1 presents an example of a protocol for Viral Hepatitis testing, whose aim is to assist physicians in selecting the most appropriate laboratory tests for conditions of suspected Viral Hepatitis (Protocol Steering Committee 1998). Some protocols are not presented as test-ordering protocols *per se* although they heavily involve guidelines on ordering laboratory tests. Table 2.2 presents an example of such a protocol for the management of renal disease in type 2 diabetes (Lanarkshire Diabetes Group 1999).

To have a marked effect on costs and to be functional, clinical test ordering protocols must: address high-volume ordering areas; be amenable to a few simple rules that can easily be remembered by clinicians; be conveniently expressed in the test order; be easily carried out by the clinical laboratory staff, and require general agreement among clinicians, laboratories, and payment agencies (Smith and McNeely 1999).

Table 2.2 Protocol for the management of renal disease in Type 2 diabetes (in natural language) (Lanarkshire Diabetes Group 1999)

Guideline Title	Management of renal disease in Type 2 diabetics – prevention and detection		
Objective	To reduce patients entering end stage renal failure by one third.		
MUST DO	1. Annual early morning first void urine: • If blood and leucocytes present - look for appropriate pathology e.g. Urinary Tract Infection (UTI). • If free of blood and leucocytes - send to local hospital laboratory for MICROALBUMIN measurement. • If 20 - 200mg/l - repeat TWO separate mornings • if 2 of 3 readings are 20 - 200mg/l - then MICROALBUMINURIA. • If < 20mg/l - then normal and re-test in one year. 2. If MICROALBUMINURIA, prescribe an ACE inhibitor for type 1, but avoid in potentially pregnant woman. Control BP (<140/80 mmHg) in type 2. 3. If DIPSTICK POSITIVE PROTEINURIA (stages 3, 4 *See Appendix 2*). • control BP < 140/80 mmHg. • in type 1 refer to STATE REGISTERED DIETICIAN for dietary protein assessment and modification if appropriate. *4.* Keep record of results. (*See page 7).*		
SHOULD DO	1. Refer stage 3, 4 to hospital diabetic clinic. *(See Appendix 2)* 2. Refer stage 5 to hospital nephrologist *(See Appendix 2).* (Dr. Bill Smith or Dr. Malcolm Hand at Monklands Hospital).		
Appendix 2: Stages of Diabetic Nephropathy	*Stage*	*Abnormality*	*Condition*
	1	Urinary albumin < 20 mg/l	Normoalbuminuria
	2	Urinary albumin 20-200 mg/l	Microalbuminuria
	3 & 4	Urinary albumin >20 mg/l (= dipstick albuminuria): ACTION REQUIRED	Macroalbuminuria
	5	Plasma creatinite > 200 umol/l: ACTION REQUIRED	End stage renal failure

These constraints have severely limited the number of areas that clinical test ordering protocols can be implemented. Furthermore, a drawback to the use of test ordering protocols for laboratory utilisation control is that clinicians do not show a sustained test-ordering-behaviour change in response to the deployment or dissemination of clinical guidelines even when they are in agreement (Kanouse and Jacoby 1988; Elson and Connelly 1995a; Elson and Connelly 1995b). There are many explanations to this one of which is the fear of litigation. Despite these constraints, it is beneficial to provide computerised support for clinical test ordering protocols as this would give rise to a number of desirable results, which include reduction of the following: unnecessary test orders, which will lower laboratory costs; the number of sample collections through sample and result re-use; and turn-around time required to reach a diagnosis (Smith and McNeely 1999). Further to this, a system implemented as an interface between the clinician and the laboratory offers the possibility of solving some of the difficult problems of developing, disseminating and adhering to test ordering protocols. The major benefits of such a system include the ability to:

- represent more sophisticated and widely applicable protocols than can currently be implemented with traditional approaches;
- make those protocols available to clinicians at the time of ordering and viewing the results;
- make test ordering protocols specific to the clinical circumstances of the patient; and
- provide a complete record for retrospective review of the clinical problem, test orders and test results.

Two major approaches have emerged in the support of clinical laboratory test ordering protocols. The first approach is the *proactive* approach in which support for test ordering protocols is based on proposing appropriate investigations, and the second approach is the *reactive* approach in which support involves denying inappropriate investigation (Peters, M., Clarke et al. 1991; Boran, O'Moore et al. 1996; van Wijk, M .A. M., Bohnen et al. 1999; van Wijk, M.A.M., Mosseveld et al. 1999). The net effect is that only those tests that the clinicians and the laboratory staff agree to be necessary for the management of the patient are ordered routinely.

This work addresses the support for the management of clinical laboratory test-ordering protocols through a unified framework that covers specification, execution and manipulation, and applies the ECA rule paradigm and database systems in the modelling and implementation framework. The aim is to provide assistance to clinicians in which test-ordering protocols that have been agreed with the laboratory are declaratively and generically specified and stored, customised for specific patients, enforced or executed by a computerised mechanism, and manipulated through operations, queries and sharing mechanisms such as healthcare middleware like the Synapses electronic healthcare record (EHCR) server (Grimson, W, Berry et al. 1998). The test-ordering protocol enforcement takes the proactive approach with the exception that the system proposes tests that have been subject to agreement or consensus.

2.5. ECA Rule-Based Support for Clinical Protocols

In terms of the ECA rule paradigm, a clinical guideline can be seen as "a method, that identifies *actions*, that are to be performed and that specifies *conditions* that govern *when it is appropriate* to perform them" (Pattison-Gordon, Cimino et al. 1996). From this definition, it

can be noted that a clinical guideline also includes situation monitoring, i.e., event monitoring with condition or appropriateness criteria determination. Thus, it can be seen that, by definition, a clinical guideline embodies the ECA rule paradigm. The recognition of the usefulness of the ECA rule paradigm in supporting the management of information and knowledge in the medical and clinical guideline domains has led to the development of the Arden Syntax for Medical Logic Modules (Hripscak, Luderman et al. 1994), which is the first, and currently the only, established standard for representing medical knowledge (HL7 1999).

In the clinical test-ordering domain, from the ECA paradigm point of view, a test order activity in a clinical test-ordering protocol can be expressed generically as: *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 could be a result of a recognisable event followed by a decision-making process that includes appropriateness criteria determination that is made before the test is ordered. 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 a 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 interesting and unique features: *First,* the scenario 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. *Second,* the actions of a test-ordering rule can be *alarm-oriented* 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 new test result, some more tests may need to be ordered immediately or at some later time – if the ordering logic is pre-determined. Obviously, it can also be the case that an action may be

pending, awaiting a medical expert's decision, and this 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 an interesting application domain for the ECA rule paradigm, which falls under *ad-hoc* triggers identified by Ceri et al. (Ceri, S., Cochrane et al. 2000) but incorporating special requirements for temporal ECA rules and comprehensive high-level facilities for dynamically manipulating the rule automatically with human concurrence from the application.

2.6. Summary

This chapter has set the context of this work by defining the major concepts that are involved in the research topic under investigation and outlining the context of the issues being dealt with in this investigation. Clinical guidelines and protocols are a form of domain information and knowledge whose management is critical to attainment of desirable healthcare outcomes. Previous research has already established that computerised test-ordering protocol systems can be helpful to both clinicians and clinical laboratories if they are integrated with other healthcare information systems such as the electronic patient record (EPR) and the laboratory information system (LIS) (O'Moore, Groth et al. 1996). The main aim of such a system would be to provide the automatic enforcement and dynamic management of the locally agreed protocols and "*prompt* rather than *dictate"* (Peters, M, Broughton et al. 1991). This study contends that the management of clinical guidelines is achieved through the three dimensions: specification, execution and manipulation. These three dimensions constitute the essential functionality that should be aimed at by a clinical guideline management approach. Most existing approaches have focused mainly on the specification and execution only and provides minimal support for manipulation management. This study is unique in that it incorporates the three aspects within a unified framework. This study proposes the use of the ECA rule paradigm for supporting the management of domain knowledge for clinical guidelines in clinical laboratory test-ordering domain. The *event-condition-action (ECA) rule paradigm* within the context of *active databases* (Dittrich, Gatziu et al. 1995) can be used to enable the electronic patient record to issue prompts and reminders to clinicians so that they can perform tasks that need to be carried out, and to suggest patient-specific decisions or procedures. An additional advantage is that active databases have also been shown to be a

viable technology for supporting workflow processes (Eder, Groiss et al. 1994). Active databases with temporal features are a promising technology for supporting clinical guidelines within an organisational setting requiring timely communication and coordination among healthcare team members. Since clinical processes are often highly unpredictable and safety critical (OpenClinical 2001), active database can be used to monitor clinical process while providing sufficient flexibility for clinicians and patients to override ECA rules when necessary and ensuring that clinicians retain the final decisions. Further to this, databases systems are efficient in managing data generated by clinical processes.

Chapter 3 Computer-Based Clinical Guideline and Protocol Management

3.1. Introduction

Clinical guidelines are usually paper-based and difficult for a busy practitioner to access at the point of care. Undoubtedly, there is a need to develop strategies that facilitate the dissemination, sharing and improvement of the method of presenting clinical guidelines and protocols for ease of accessibility and promotion of clinicians' compliance with them. Moreover, clinicians have been observed not to show a sustained behaviour-change in compliance with clinical guidelines, even if they agree with them (Kanouse and Jacoby 1988). However, studies have established that if the guidelines are presented to the clinicians at the time when they are making a decision to order a test or accessing the test results or treating the patient, clinicians tend to comply with the guidelines more than at any other time (Grimshaw and Russell 1993). This study seeks to help in the promotion of compliance to clinical guidelines and protocols.

The development of computer-based guideline or protocol systems have been proposed in order to present clinical guidelines to clinicians at the time when the clinicians need them (Peters, M and Broughton 1993). Attempts have been made to build such systems for the domain of clinical laboratory test-ordering, for example LUMPS (Peters, M., Clarke et al. 1991; Matimer, McCauley et al. 1992) and BloodLink (van Wijk, M.A.M., Mosseveld et al. 1999), but these guideline systems have not adequately addressed the problem of the full-scale management of domain knowledge contained in clinical guidelines that they support.

Most approaches in the literature have used the *production rule* paradigm (Newell and Simon 1972) to model and implement clinical test-ordering protocols. The need to:

- manage the guideline knowledge and its enforcement;
- consider the clinical situations, which includes events and appropriate actions; and
- consider other attributes of the patient possibly contained in the electronic patient record,

makes the ECA rule paradigm in active databases (Dittrich, Gatziu et al. 1995) a promising technology for supporting the management of clinical laboratory test-ordering protocols. A literature review of the ECA rule paradigm and its applications is presented in Chapter 4.

This chapter presents the state-of-the-art in the form of a review of the literature on the domains under investigation. Before the literature review is presented, the chapter presents a brief review of the core issues in the domains understudy. More importantly, the chapter presents our framework, *SpEM* (short for **Sp**ecification, **E**xecution and **M**anipulation of CGPs), which we developed for supporting the management of computerised clinical guidelines and protocols. The SpEM framework is then used as a guide to analysis in the literature review.

The rest of this chapter is organised as follows: *Section 3.2* presents the SpEM framework for guideline or protocol management support. *Section 3.3* presents a literature review of the approaches and systems to the support for the management of clinical guidelines and protocols. The literature review closely follows the SpEM framework. *Section 3.4*. outlines the implications of the literature review findings. *Section 3.5* summarises this Chapter.

3.2. The SpEM Framework for Supporting the Management of Computerised Clinical Guidelines and Protocols

The SpEM framework allows the management of clinical guidelines to be achieved through the three dimensions: *specification, enforcement/execution and manipulation.* Specification is the definition of a clinical guideline or protocol by using a formal language. Enforcement is to the computerised enactment or execution of the formal guideline or protocol specification with respect to a specific clinical case. Manipulation includes: performing operations on, and querying guideline information as well as the information on the objects, subjects and effects of applying the information to specific clinical cases. These three dimensions constitute the three components of the CGP management framework, which will be called, **SpEM**, (**Sp**ecification, **E**nforcement/**E**xecution and **M**anipulation).

3.2.1. SpEM Conceptual Architecture

Figure 2 illustrates the SpEM architecture in terms of the three planes each concerned with one of the three aspects: specification, enforcement and manipulation. In the *specification plane*, the guideline information is captured, formally specified and stored for easy access,

use and maintenance. As illustrated in Figure 2, *specifications* of the captured guideline information are customized to suit the problem scenario and then prepared for *enforcement*.

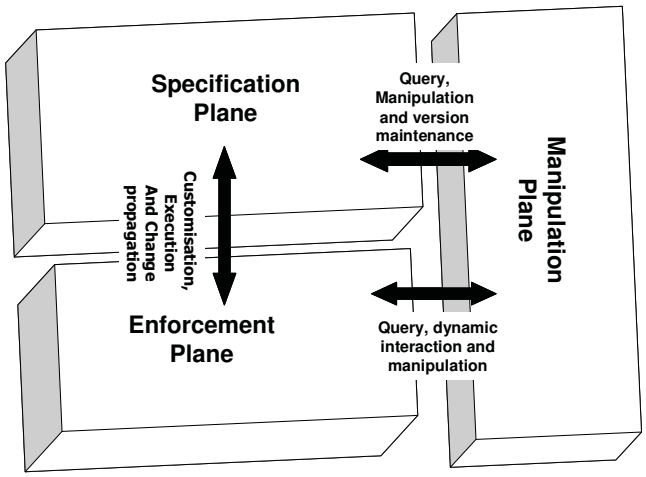


Figure 2 The SpEM framework for clinical guidelines or protocol information management

In the *enforcement plane*, the specified guideline information is put to use in the solution to problems within the domain. This application of information can be manual or computerised or their combination. In the *manipulation plane*, both the guideline information and its application process are manipulated through the performance of defined operations and queried by using a declarative language. It should be noted that both the specifications and execution process are subject to manipulation within the manipulation plane.

3.2.2. CGP Support in the Specification Plane

The specification plane provides a means to specify the global properties or meta-data for clinical guidelines. These global properties define their purpose and when they may be or should be used. The global properties are necessary to allow a computerised system to provide assistance to a clinician in deciding what guideline or protocol could be applicable to particular patient circumstances. Gordon et al. (1996) summarised these most commonly specified properties in guideline systems as including: *guideline task:* e.g., diagnosis and management of chronic asthma; *entry criteria:* what a patient must satisfy in order for the guideline to be applicable to them; *exclusion criteria:* conditions that define when the

protocol must not be applied; *indications and contra-indications:* patient-specific factors that need consideration in order to decide whether or not the protocol can be used.

In order to effectively support CGPs, the specification plane must provide a guideline representation model for expressive guideline knowledge representation, Such a model must incorporate *representation primitives* that make up the basic components of a guideline representation model; *structural arrangement* of the representation primitives that makes up the application process of clinical guidelines; and *modelling of patient data* (Wang, Peleg et al. 2002). The following are the typical generic representation primitives that are required in a guideline or protocol representation model:

- *action:* clinical or administrative task that is recommended to perform, maintain, or avoid during the process of guideline application;
- *decision:* a selection from a set of alternatives based on predefined criteria in a guideline;
- *patient state:* a materialisation of a treated individual's clinical status based upon the actions that have been performed and the decisions that have been made;
- *execution state:* a description of a guideline implementation system based on the stage of the task such as the action and decision during the process of guideline execution (Wang, Peleg et al. 2002).

Patient and execution states are two sides of the guideline application process. The two concepts are closely related to each other. However, patient state can be affected by changes outside the control of a guideline system. Consequently, patient state and execution state may diverge from one another. Most guideline models support either patient state or execution state but not both without loosing expressiveness (Wang, Peleg et al. 2002). In this study, the approach taken views patient state as a domain-dependent property while execution state is viewed as a generic property of the execution mechanism for the guideline application process.

A formal guidelines representation model within the specification plane has the following benefits:

- Provides in-depth understanding of the clinical care processes addressed by clinical guidelines (Greenes, Peleg et al. 2001);
- Can be used to identify different requirements by clinicians for assistance during the process of decision-making ;
- Supports automatic verification and validation of clinical guidelines;

- Can be used to facilitate standard approaches to guideline dissemination;
- Can be used as a generic template in the integration of clinical guidelines with the healthcare information system at a local institution (Wang, Peleg et al. 2002).

3.2.3. CGP Support in the Execution Plane

The execution plane depends on the guideline representation model and language provided by the specification plane in order to support the computer-based execution of the guideline-based care process. The computational method used in the execution of guidelines is dependent on the guideline/protocol representation formalisms used. In this work, the execution plane uses event-condition-action (ECA) rules to execute clinical guidelines. ECA rules have the general form: ON *event* IF *condition* DO *action*. The ECA rule paradigm encapsulates the core elements for capturing and enforcing guideline knowledge. Table 3.1 summarises the guideline/protocol representation formalisms and computational methods from the literature (Tu, S. W. , Johnson et al. 2001; Tu, S.W. and Musen 2001).

In the **rule-based paradigm**, productions rules of the form: IF *condition* DO *action,* have been used to support clinical event monitoring as well as clinical protocols (Shortliffe, Axline et al. 1973; Starren and Xie 1994). **Logic-based methods** represent guideline knowledge in a declarative knowledge base with logical criteria forming the basis for selecting a guideline for application to a patient. Examples of logic-based guidelines representation method are PROforma (Fox, Johns et al. 1996) and PRESTIGE (Gordon, Herbert et al. 1996).

Table 3.1 Guideline representation formalisms and computational techniques

Model of Representation	Example	Method of Representation	Computational Method	Tasks
Rule based	MLMs using Arden Syntax (Clayton, Pryor et al. 1989), Decision Table (Shiffman 1997)	Event-condition-action rule paradigm	Mix of production system and procedural program	Primarily Decision Making, Data Interpretation, Goal Setting, and Action Sequencing possible but not supported explicitly
Logic-based	PROforma (Fox, Johns et al. 1996)	Declarative formal logic	Activation of PROforma tasks through evaluation of constraints/ preconditions and assertion of post-conditions	Decision Making, Action Sequencing, Data Interpretation, Goal Setting and Action refinement through decision/actions
Network-based	ONCOCIN (Shortliffe, Scott et al. 1981)	Augmented Tranistion Networks(ATNs), Rules	Episodic Skeletal Plan Refinement	Action Sequencing through ATN, Rule-based Decision Making and Action refinement, Data Interpretation through temporal Queries
	PROGIGY III (Shortliffe, Scott et al. 1981)	ATNs of patient states and decisions, Hierarchy of actions	ATN Traversal, Action Refinement as Decisions	Decision Making, Sequencing of Decisions, Action Refinement
	GUIDE/Pavia Models (Quaglini, S., Stefanelli et al. 2000b)	GL/Petri Nets/WPDL	Petri Net, Workflow Management System (WfMS)	Action and Decision Sequencing, Decision Making
Decision Theory	GUIDE/Pavia Model (Quaglini, S., Stefanelli et al. 2000b)	Decision tree, Influence diagram	Decision Theory Techniques	Decision making

In **network-based models**, guideline knowledge is represented as graphical flowcharts or networks that have arcs specifying sequencing of actions and hierarchical decomposition for controlling complexity. Logical criteria using patient-specific data are used to further control the execution of actions. The semantics of the flowchart languages are those for formal network modelling tools such as *augmented transition networks* and *Petri Nets*. Examples of network-based models include: ONCOCIN (Musen, M.A. , Tu et al. 1992), PRODIGY (Johnson, P.D., Tu et al. 2000) and GUIDE (Quaglini, S. , Stefanelli et al. 2001). In another approach to the classification of guideline representation formalisms, de Clercq et al. views the formalisms developed to-date as falling into one of the following two classes (de Clercq, Blom et al. 2000):

Primitive-based approaches model guidelines in terms of explicit primitives that characterise the stereotypical tasks that a guideline is to perform, e.g., actions and decisions. Examples of primitive-based guideline modelling approaches include Arden Syntax (Hripscak, Luderman et al. 1994), PROforma (Fox, Johns et al. 1996), and GLIF (Ohno-Machado, Gennari et al. 1998).

In **generic problem-solving method (PSM)-based approaches**, the modelling methods do not focus specifically on guideline-based care, but focus more on abstract behaviour of decision-support systems in general. The works of Schreiber et al. (Schreiber, Akkermans et al. 1999), Motta (1999), and Musen et al.(1995) would fall into this category. System behaviour is modelled in terms of independent classes of re-usable components presented as: *domain ontologies* that describe concepts and their relationship in a domain;

and *domain-independent algorithms*, known as *problem-solving methods* (PSMs), for performing generic tasks such as classification, planning, critiquing and constraint satisfaction. Examples of PSM-based guideline approaches are those that are based on Protégé 2000 (Musen, M. A. , Gennari et al. 1995; Grosso, Eriksson et al. 1999) such as EON (Musen, M.A. , Tu et al. 1996).

3.2.4. CGP Support in the Manipulation Plane

The manipulation plane provides the *operations* on and *queries* against guideline knowledge and information. The operations *add, delete* and *modify* may be performed at high-level on the collection of guideline specifications, the specification database, e.g., adding a new protocol to or deleting an existing protocol from the database. The operations may also be performed at a low-level on the individual guideline specification when components are added, deleted or modified from the specification. Manipulation of the individual guideline instance may involve execution-oriented operations like start, stop or truncate. *Queries* may be issued in order to obtain information about guideline composition and/or execution. An example of a high-level query could be: *Which protocols (specifications) in the system would involve blood pressure measurements?* An example of a low-level query could be: *Within a given protocol (specification) in the system, which part or component requires waiting for a period of 3 months?* Another important aspect of the manipulation plane is the re-play of what happened during some period in the past history of executing a guideline or protocol.

In guideline systems that support the creation of guideline specifications, it is usually the case that these specifications are used to create protocol instances that execute with respect to each individual patient. It is also possible that the generic specification and its instances are clearly separated. Changes could be made to either the specification or to any of the instances. If such changes are made, the manipulation plane needs to support any form of change propagation or consistency maintenance that may be required between components within the system. For instance, in the Asgaard/Asbru system (Shahar, Miksch et al. 1998), during execution, the clinician may decide to deviate from the guideline and the system captures these deviations together with the associated intentions and allows execution to proceed (Miksch 1999). The captured deviations may represent new knowledge which may be used to change either the specification to create a new version or the other instances that are already executing.

3.2.5. Requirements for Realising the SpEM Framework

In the *specification plane*, a declarative language is required to specify guideline or protocol information. In the *execution plane*, a suitable mechanism is required to enforce the guideline information. In the *manipulation plane*, manipulation operators and a query language are required to manipulate and query the both the specification and the execution planes. When these requirements are fully met, the SpEM Framework ensures the full-scale manageability of information. Supporting information management involves providing facilities for specifying, storing, enforcing, maintaining and disseminating the information (Borghoff and Pareschi 1997; Benjamins, Fensel et al. 1998; Buckingham Shum 1998). The main aspects of the problem of supporting guideline information management that are of interest to this work are the three components of the SpEM Framework. Guideline information is required to be formally specified to create generic computerised specifications, which should be subject to persistence, execution, and manipulation in a specific problem context. This requires at the very least:

1. A specification model and language;
2. A persistence mechanism such as a database system;
3. An execution mechanism; and
4. A manipulation and query language.

3.3. Clinical Guideline Management Support Approaches and Systems

The literature review follows the SpEM Framework presented in Section 3.3. The aim of the review framework, illustrated in Figure 3, is to establish the state-of-the-art in the support for the full management of computerised CGPs in terms of the SpEM framework.

Of interest to the review is the support for the three planes of specification, enforcement and manipulation. For each work or guideline system reviewed, the several aspects will be of interest. The first aspect of interest is the support provided by the guideline system for the specification of guidelines/protocols, which is provided for through a specification model and its language as well as some form of persistence for the specifications. The specification model and language for computerised guidelines/protocols may follow one or a hybrid of paradigms which may be rule-based (e.g., using production or ECA rules), logic-based (e.g.,

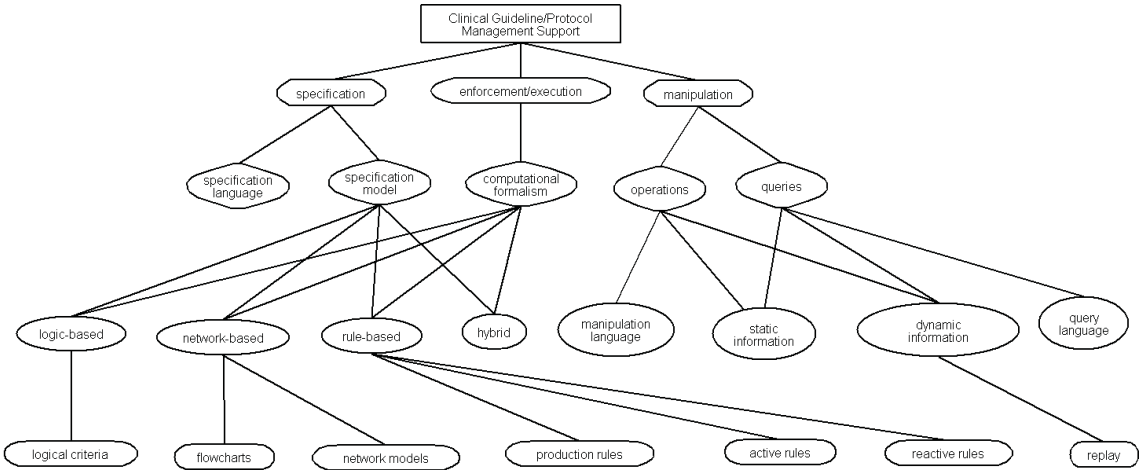


Figure 3 A classification of issues in the support for the management of computerised clinical guidelines/protocols

using some logical criteria or constraints), network-based (e.g., using a graphical flowchart or a network model such as augmented transition networks or Petri nets);

The second aspect of interest is the support provided by the guideline systems for the computer-based execution of a guideline specified by using the system's specification language. The software mechanism to execute a guideline uses a computational formalism that may be rule-based, network-based or a hybrid of computational formalisms.

The third aspect of interest is the support provided by the guideline systems for the manipulation of guidelines/protocol knowledge and information, which may be provided from both the static and dynamic perspectives. The static perspective includes guideline/protocol specifications and patient demographics. The dynamic perspective includes knowledge and information about the execution process and its output as well as modification and version information associated with specifications. Replaying what has happened during a specified time interval is a useful feature to include as part of the dynamic perspective for supporting guideline/protocol manipulation. Manipulation would be made possible by providing manipulation and query languages to handle operations and queries on the guideline/protocol information.

Among the pioneering works related to some aspects of the specification and execution of clinical protocols, are that of MacDonalds et al. (1980) and East et al. (1990). MacDonalds et al. (1980) developed a computerised medical record system that detected and reminded the responsible clinician about clinical events that might need corrective action. East et al. (1990) developed a computerised protocol system to direct the management of

arterial hypoxemia in critically ill patients with adult respiratory distress syndrome. Since these early works of MacDonalds et al. and East et al., a number of clinical guideline systems have emerged in various areas of healthcare especially in the domains of *diagnosis and therapy planning* and clinical laboratory *test-ordering*. The next sections present a review of some of the major guidelines systems and works that are of interest to this study.

3.3.1. Computerised Clinical Laboratory Test-Ordering Protocol Systems

The class of clinical guidelines or protocols that are of interest to this study is that for guiding clinicians in ordering clinical laboratory tests. Hence, before reviewing works on computer-based support for clinical guidelines in general, this section starts by reviewing major works that address computer-based support for clinical test ordering guidelines or protocols.

Peters *et al* (1991) implemented a computerised management protocol system (mainly for liver transplant patients), called the Liver Unit Management Protocol System (LUMPS). The system was developed in MUMPS (Bowie and Barnett 1976) , a general purpose programming language with a native hierarchical database facility which was targeted towards applications in the healthcare domain. Test ordering protocols in LUMPS were represented in the form of production rules, which were encoded directly in MUMPS. The aim of LUMPS was to provide the "automatic reinforcement of locally agreed protocols of patient care, expressed as simple rules, which prompt rather than dictate" (Peters, M., Clarke et al. 1991). The main emphasis in LUMPS was to provide, for user-specified patient categories, from hospital wards to the clinical laboratory information system, personalised, editable laboratory medicine investigation protocols based on locally agreed guidelines and dynamically reflecting current pathology (Peters, M., Clarke et al. 1991). While it was recognised that the rules or protocols in LUMPS should be flexible, readily upgradeable and updatable, LUMPS did not facilitate interactive modification of, or addition of new rules or protocols. This work obtains its inspiration from the approach developed in LUMPS.

LUMPS uses the production rule paradigm to computerise problem-oriented or patient category based test ordering protocols for delivering a patient-specific test order plan, which a clinician can edit and/or modify. LUMPS differs from the two systems, BloodLink and Laboratory Advisor System (LAS), which are reviewed next, in that it issues patient-specific

suggestive prompts for test orders that would have been locally agreed and pre-defined for a given patient category without necessarily eliciting information from the user.

van Wijk *et al* (1999) developed a decision-support system, called BloodLink-Guideline, for ordering blood tests based on clinical guidelines designed by the Dutch College of General Practitioners for general practitioners (GPs) in the Netherlands. The GPs use the electronic patient record to activate BloodLink-Guideline to order blood tests (van Wijk, M.A.M., Mosseveld et al. 1999). When using the system, a GP initially selects the appropriate guideline, e.g., liver disease. BloodLink-Guideline then queries the GP about the reasons for requesting the tests with the objective of identifying an indication. Based on the indication, the system proposes the relevant tests. The GP decides whether or not to comply with the protocol and may also add tests to or remove tests from the proposed list. BloodLink-Guideline subsequently prints a patient-specific blood test request form that includes the necessary patient data, the indication, the tests requested, and the additional instructions for the laboratory. Finally, BloodLink-Guideline updates the patient record to show what tests have been requested. If the GP's indication cannot be established in BloodLink, the GP can select the option "other indication". If this option is selected, then the system is not able to provide recommendations for test ordering. Instead, the GP has to select the required tests by typing the initial letters of tests (van Wijk, M.A.M., Mosseveld et al. 1999). Blood link is a *pro-active system* that suggests certain tests to the clinician according to a given clinical protocol. The authors did not discuss how guideline information is represented in the BloodLink-Guideline system. The BloodLink-Guideline System computerises national and regional guideline information and provides guideline-based recommendations of test orders after having obtained a clinical indication or working hypobook on the patient by interviewing the clinician.

The Laboratory Advisor System (LAS) is a guideline-based expert system that works interactively with clinicians to assist them with test selection and result interpretation throughout the laboratory investigation of a patient (Smith and McNeely 1999). It uses its underlying information base to optimise the laboratory investigations for better care and low cost by optimising patient specific test ordering strategies, providing patient-specific result interpretation, and offering contexts-sensitive assistants throughout the process. In LAS, guideline information is represented using two formalisms: the standard production rule representation and an information representation scheme that is based on pattern recognition that conceptualises expertise as a highly developed pattern recognition skill and captures

information in “pattern-consequence" relationships. In LAS, patterns are relevant clinical information, and consequences are testing recommendations and interpretations (Smith and McNeely 1999).

LAS uses the production rule paradigm together with a pattern recognition approach to capture guideline information and uses the information to make appropriate recommendations based on patient-specific information elicited from the clinician. LAS is similar to LUMPS in its use of the production rule paradigm and to BloodLink-Guideline in eliciting patient-specific information from the clinicians in order to recommend which tests to order.

Bindels et al. (Bindels, de Clercq et al. 2000) developed a real-time automated reminder system aimed at changing physicians’ test ordering behaviour. The system, which is called *The Maastricht System* for the purpose of this review, uses practice guideline information and focuses on appropriateness of test orders. The approach of the Maastricht System is to critique the rationality of test orders at the moment the clinicians order a test. The system consists of five components: the information base, an order entry system, a reactive support module for issuing reminders, a passive support module that allows clinicians to request background information about the guideline, and a database for the electronic patient record (EPR) (Bindels, de Clercq et al. 2000). Guideline information is implemented as independent production rules, which are based on patient-specific data from the EPR and leads to a reminder if the corresponding guideline is not complied with. The decision to represent information using the production rule paradigm was made after studying regional and national guidelines in the Netherlands. To enable reasoning about the medical domain, an ontology built using Protégé (Musen, M. A. , Gennari et al. 1995) is used. Objects in the ontology are diagnostic tests, patient information, medical information and reasons for the test order. Unlike the BloodLink-Guideline system, the Maastricht system focuses on appropriateness leaving decision-making to clinicians. It reacts only if the test order is not in compliance with the clinical guidelines. The accuracy of the Maastricht system depends on rule management, i.e., with the maintenance of reminders in the rule base, and on the completeness of the medical data provided by clinicians, i.e., the complete electronic medical record.

The Maastricht System is similar to the other systems in its use of the production rule paradigm to represent guideline information. However, the Maastricht System takes a different approach in its enforcement of the guideline information. It monitors a clinician’s

test orders and uses guideline information to react with feedback when test orders do not comply with guidelines. The monitoring and reactive feedback occurs at the moment when the test orders are being made. Table 3.2 presents a summary of the review on systems that support CGP management for the domain of clinical laboratory test ordering.

Table 3.2 Literature review findings for the major systems that support the management of clinical guidelines and protocols for clinical laboratory test ordering

Guideline/Protocol System	Computational Formalism Used	SpEM Framework Support *Key:* √ - full support, * - weakly supported, **X** – no support,			
		Specification	*Execution*	*Manipulation*	
				Operation	*Query*
LUMPS (Peters, M., Clarke et al. 1991)	Production rule	*	√	*	**X**
BloodLink (Bindels, de Clercq et al. 2000)	Hybrid: logic, production rule	**X**	√	**X**	**X**
Maastricht (van Wijk, M.A.M., Mosseveld et al. 1999)	Hydrid: production rule, reactive rule	**X**	√	**X**	**X**
LAS (Smith and McNeely 1999)	Hybrid: production rule, pattern rercognition	**X**	√	**X**	**X**

The reviewed systems support the enforcement of guideline or protocol knowledge. Only LUMPS partially support the specification of protocols. Other systems do not explicitly support specification of protocols. The guideline or protocol does not exist as an explicit conceptual, logical or physical entity. In LUMPS, a guideline or protocol is identifiable as an explicit entity, which can also be manipulated by editing it. However, LUMPS did not provide a generic specification and manipulation languages; In overall, the SpEM Framework is inadequately supported as only the execution plane is supported by all systems while the specification and manipulation planes are either not supported or are partially or weakly supported. Furthermore, none of the systems provided a generic and unified framework and mechanism to support different guidelines or protocols from the ones they were designed to support.

Guideline support approaches for the clinical test ordering domain depend mainly on the traditional production rule paradigm for knowledge representation and take either the *pro-active* or *reactive* approach (Boran, O'Moore et al. 1996; O'Moore, Groth et al. 1996) to the enforcement of the guideline knowledge. The pro-active approach suggests test orders and allows the clinician to decide to accept, modify or reject the suggested test orders. The reactive approach performs real-time monitoring of the clinician's test orders and reacts with feedback whenever a test order represents non-compliance with the guideline, i.e., it critiques

test orders at the moment the orders are being made. Existing approaches and systems do not clearly separate the specification, execution and manipulation aspects of guideline knowledge management.

3.3.2. Guideline Models and Systems for the Domain of Diagnosis and Therapy Planning

The major works on computer-based support for the management of guideline and protocol information in the domain of diagnosis and therapy planning during the past decade are presented in Table 3.3.

Table 3.3 Diagnosis and therapy guideline models and systems

Guideline Approach/System	Organisation	Reference
DILEMMA/PRESTIGE	The Dilemma Consortium	(Thomson 1995; Gordon and Veloso 1996)
EON/Dharmma	Stanford Medical Informatics	(Musen, M.A. , Tu et al. 1996; Tu, S.W. and Musen 2001)
PROforma	Imperial Cancer Research Fund and Intermed Ltd, London	(Fox, Johns et al. 1996)
SIEGFRIED		
GLIF	Intermed Collaboratory	(Ohno-Machado, Gennari et al. 1998)
Asgaard/Asbru	Vienna University of Technology & Stanford Medical Informatics	(Shahar, Miksch et al. 1998)
GUIDE	Pavia University	(Quaglini, S., Stefanelli et al. 2000b)
PRODIGY	University of Newcastle-upon-Tyne	(Johnson, P.D, Tu et al. 1999)
GASTON	Medical Engineering Division at the Eindhoven University of Technology, the Netherlands	(de Clercq, Blom et al. 2000)
GLARE	Dipartomento di Informatica, Universita de Piemonte Orientale "Amedeo Avogadro", Alessandria, Italy, in collaboration with the Laboratorio di Informatica, Azienda Ospedaliera S. Giovanni Battista, Torino, Italy	(Terenziani, Molino et al. 2001)
Arden Syntax & Medical Logic Modules (MLM)	Columbia University	(Clayton, Pryor et al. 1989; ASTM 1992; HL7 1999)
HyperCare	Politecnco di Milano	(Caironi, Portoni et al. 1997)

In Table 3.3, the Arden Syntax for Medical Logic Modules (HL7 1999) and HyperCare (Caironi, Portoni et al. 1997) are different from the rest because they make use of the ECA rule paradigm. The following sub-sections present a brief review on each of the major guideline models and systems with the exception the Arden Syntax and HyperCare, whose review will be covered in Chapter 4 where the ECA rule paradigm applications are reviewed.

DILEMMA/PRESTIGE

DILEMMA (Thomson 1995) was an 1991-4 European Community (EC) AIM Programme while PRESTIGE (Gordon and Veloso 1996) was a project under the EC 4th Framework Health Telematics Programme. The DILEMMA Project produced a generic approach to the representation of knowledge from clinical guidelines and protocols, which was subsequently enhanced and implemented in the PRETIGE Project (Gordon, Herbert et al. 1996). The DILEMMA/PRESTIGE conceptual protocol and guideline model (CGPM) is an object model that defines: the kinds of objects or entities which may appear in a guideline or protocol; the

relationships between these objects or entities; and the attributes of these objects or entities (Gordon, Herbert et al. 1996; Gordon, Herbert et al. 1997). The types of objects defined include: the general concepts such as activities, acts and case-specific phenomena, e.g., diagnosis and symptoms; the protocol structure and version; and the expressions with several roles such as conditions defining entry-criteria, patient characteristics, attributes of activities and contexts of care, clinical procedures, and templates for data collection (Gordon, Herbert et al. 1997). The DILEMMA/PRESTIGE Model consists of two main components: the first describes healthcare in general, and the second describes clinical guidelines or protocols. The PRESTIGE Projects guideline authoring tools include: the guideline authoring and dissemination tool (GAUDI), which incorporates a terminology server and model (GRAIL and GALEN); and the Guideline Editing And Authoring Module (GLEAM) .

EON/Dharmma

The EON/Dharmma (Musen, M.A. , Tu et al. 1996; Tu and Musen 2001) clinical guideline model and system was developed at the Stanford Medical Informatics (SMI), Stanford University. The model uses a component-based approach and the system is a suite of re-usable software components for creating clinical guideline applications. Therefore, as stated in Section 3.3.3, the EON/Dharmma approach is a problem-solving method (PSM)-based approach to guideline modelling. The approach uses an extensible set of models among which the clinical guideline model is the core. The other models in the set are: the patient data information model, the medical-specialty (ontology) model and a temporal abstractions model. Definitions of decision-support services are based on a task-based approach. These decision-support services can be implemented using alternative/different techniques. In the EON guideline execution server, patient-specific recommendations are generated using formalised clinical guidelines and patient data linked together through the ontology of medical concepts in the medical-specialty model. The EON system also includes two further components: a temporal data mediator for supporting queries on temporal abstractions and relationships; and an explanation facility that provides explanation services to other components within the system.

PROforma

PROforma (Fox, Johns et al. 1996; Fox, Johns et al. 1998) was developed by the Advanced Computation Laboratory of Cancer Research in the UK. PROforma is based on the R^2L

language (Fox and Das 2000) and combines object-oriented modelling with logic programming (Fox, Johns et al. 1996). The PROforma guideline model strives to be expressive while using, by design, only a minimal set of modelling primitives. PROforma's guideline model consists of a task ontology that has four types of tasks, which are: actions, compound plans, decisions and enquiries (Fox, Johns et al. 1996). All tasks have common attributes that describe goals, control-flow, pre- and post-conditions.

SIEGFRIED

The SIEGFRIED (System for Interactive Electronic Guidelines with Feedback and Resources for Instructional and Educational Development) (Lobach, Gadd et al. 1997) approach uses a relational database to construct a generalized guideline knowledge base. The SIEGFRIED knowledge representation scheme was developed to capture guideline content and logic within the constraints of a relational database model (Lobach, Gadd et al. 1997). The relational database model for CGPs uses a hybrid of structured and procedural knowledge representation formalisms to represent guideline content and logic. In the SIEGFRIED system, a database schema based on a relational model is used for computerizing CGPs using a hybrid of structured and procedural knowledge representation schemes, which accommodated all necessary representational requirements (Lobach, Gadd et al. 1997). The SIEGFRIED knowledge representation scheme for CGPs conforms to a relational database model without compromising expressivity or completeness. This knowledge base was designed for use with Internet-based decision support applications. SIEGFRIED uses the Internet to present interactive CGPs that are customized to an individual patient and available at the point of care (Lobach, Gadd et al. 1997). The advantages of the relational schema-based guideline knowledge representation are:

1) ease-of-maintenance resulting from the availability of the database query language, the SQL;
2) The generic nature of the relational model permits standard accessibility of the clinical guideline knowledge to many applications; and
3) Since the medical record could be implemented using the relational model, it may share the same format as guideline knowledge, making it easier to address some of the problems of integration.

GLIF

The Guideline Interchange Format (GLIF) (Ohno-Machado, Gennari et al. 1998; Peleg, M, Boxwala et al. 2000) is a clinical guideline specification language. It is a product of collaboration among various research groups at Columbia, Stanford and Harvard Universities, which constituted the InterMed Collaboratory. The main aim of GLIF is the sharing of clinical guideline specifications among different healthcare organisations and software systems. As a result, GLIF builds on the useful and common features of other guideline models and emphasises on incorporating standards used in healthcare. For instance, GLIF uses a medical data model that is based on the Health Level 7 (HL7) Reference Information Model (RIM) (Schadow, Russler et al. 2000). Furthermore, the expression language of GLIF was initially based on the Arden Syntax (Hripscak, Luderman et al. 1994), an HL7 standard (HL7 1999). Currently, an object-oriented clinical guideline expression language, called GELLO (Ogunyemi, Zeng et al. 2002), is being considered as an HL7 standard and may subsequently replace the Arden Syntax as GLIF's expression language.

Asgaard/Asbru

The Asgaard/Asbru (1998) clinical guideline model and system is collaborative work between Vienna University of Technology and Ben Gurion University of the Negev, Israel. Clinical guidelines are specified using the Asbru language, which is a time-oriented, intention-based, skeletal-plan specification language (Shahar, Miksch et al. 1998). In the Asbru language, procedures in a clinical guideline are expressed as skeletal plans. The Asgaard system emphasises on execution-time flexibility in the achievement of particular intentions (Miksch 1999). Skeletal plans in the Asbru language are made more expressive by:

1) the characterisation of plan attributes such as intentions, conditions, and effects;
2) addition of a rich set of ordering constructs for plans; and
3) the definition of temporal dimensions for states and plans.
4) Bounding intervals are used in the language to express uncertainty in both temporal scope and parameters (Shahar, Miksch et al. 1998).

GUIDE

The GUIDE (Quaglini, S., Stefanelli et al. 2000b) modelling approach was developed at Pavia University and is sometimes referred to as the Pavia Model. GUIDE integrates clinical and organisational workflow issues (Dazzi, Fassino et al. 1997). It does so by addressing

communication, coordination and medical problems which are relevant in supporting the management of a clinical guideline or protocol in a healthcare organisation. The GUIDE modelling approach leads to the development of a patient workflow management system, called a *careflow management system* (CfMS) (Quaglini, S., Stefanelli et al. 2000b; Quaglini, S. , Stefanelli et al. 2001), from a detailed model of the medical work process and the organisational structure. The medical work process is represented through clinical practice guidelines while the organisational structure is expressed through an ontological description of the organisation (Dazzi, Fassino et al. 1997; Quaglini, S. , Stefanelli et al. 2000a). To be able to support the representation of sequential, parallel and iterative logic flows the Pavia guideline model, GUIDE, uses the Petri Net formalism. The major advantage of the Petri Net formalism, when applied to healthcare, is its ability to support the modelling of complex concurrent processes and to integrate clinical tasks specified in guidelines with the organisational models to manage patient careflow (Quaglini, S. , Stefanelli et al. 2001; OpenClinical 2003).

PRODIGY

PRODIGY (Johnson, P.D, Tu et al. 1999; Johnson, P.D., Tu et al. 2000) was developed at the University of Newcastle upon Tyne. The PRODIGY approach focuses on supporting clinical guidelines for the area of chronic disease management in primary healthcare. The PRODIGY guideline model is composed of two distinct components, which are: the *disease state map* to model decision-making. In the disease state map, a chronic disease is represented as a number of 'patient states'. Each patient state is called a scenario. At each state, a clinician has a number of choices of actions. Actions have outcomes, i.e., a patient remains in the same or moves to a different state at the next consultation; and a *consultation template* to model the care process which consists of actions and information management that are relevant whenever patient is seen. There is one consultation template for each scenario.

GASTON

The GASTON (de Clercq, Hasman et al. 2001) clinical guideline modelling approach was developed in the Medical Engineering Division at the Eindhoven University of Technology, in the Netherlands. In the GASTON approach, the guideline representation formalism uses an ontological representation to specify a guideline in the form of: domain ontologies, which hold domain-specific knowledge; and method ontologies, which hold primitive and complex

problem-solving methods (PSMs) (de Clercq, Blom et al. 2000). In the GASTON framework, the Ontology Editor is used to develop both the Method Ontology and the Domain Ontology while the Method Library contains all methods required by the clinical guideline and the Method Manager maps concepts in the Domain Ontology to knowledge roles in the Method Ontology (de Clercq, Blom et al. 1999).

GLARE

GLARE (guideline acquisition, representation and execution) (Terenziani, Molino et al. 2001) was developed by the Dipartomento di Informatica, Universita de Piemonte Orientale "Amedeo Avogadro", Alessandria, Italy, in collaboration with the Laboratorio di Informatica, Azienda Ospedaliera S. Giovanni Battista, Torino, Italy. GLARE is a modular approach for managing clinical guidelines. The GLARE approach uses a modular architecture that allows the separation between the *specification* and the *execution* of clinical guidelines (Terenziani, Molino et al. 2001). The GLARE representation language or formalism consists of two main different types of actions: plans or composite actions, and atomic actions, which can be queries, decisions, work actions and conclusions (Guarnero, Marzuoli et al. 1998). The order of execution of these actions are defined by control relations, which include: concurrent, sequence, alternative, and repetition constructs (Terenziani, Mastromonaco et al. 2000; Terenziani, Molino et al. 2001).

Findings and Discussion

Table 3.4 summarises the findings of the literature review on the support for the SpEM framework and the computational formalisms employed. Guideline support approaches and systems for the domain of diagnosis and therapy planning provide advanced and comprehensive modelling concepts and frameworks, and computational formalisms. However, these guideline support approaches provide guideline support mainly in terms of the specification and enforcement or execution of guideline knowledge and pay little or no attention to the comprehensive support of the manipulation, i.e., performing operations and querying of the guideline knowledge and information about the execution process of their instances.

Table 3.4 Literature review findings for systems that support the management of clinical guidelines and protocols

Guideline/Protocol System	Computational Formalism Employed	SpEM Framework Support **Key:** √ - full support, * - weakly supported, X – no support,			
		Specification	*Execution*	*Manipulation*	
				Operation	*Query*
DILEMMA/ PRESTIGE (Gordon, Jackson-Smale et al. 1994; Gordon and Veloso 1996)	*Network-based:* network of components, state-transition model of action execution	√	√	X	X
EON/Dharmma (Musen, M.A. , Tu et al. 1992; Tu, S.W. and Musen 2001)	*Hybrid:* network-based core model, Boolean criteria, temporal patterns and selected formalisms for suitable for each task components	√	√	X	X
PRO*forma* *(Fox, Johns et al. 1996)*	*Hybrid:* network of plans and procedures, declarative formal logic	√	√	X	X
SIEGFRIED (Lobach, Gadd et al. 1997)	*Hybrid:* Structured and procedural representation with a relational data model	√	√	*	*
GLIF (Ohno-Machado, Gennari et al. 1998)	*Network-based:* flowchart of structured actions and decisions.	√	*	X	X
Asgaard/ Asbru (Shahar, Miksch et al. 1998)	*Hybrid:* hierarchical skeletal planners with a library of various problem-solving methods.	√	√	*	*
GUIDE (also Pavia Model) (Quaglini, S., Stefanelli et al. 2000b)	*Network-based:* flowcharts based on Petri Nets	√	√	X	X
PROGIDY (Johnson, P.D, Tu et al. 1999)	*Network-based:* augmented transitions of patient states and decisions	√	√	X	X
GASTON (de Clercq, Hasman et al. 2001)	*Hybrid:* frame-based model with flowcharts and production rules	√	√	*	X
GLARE (Terenziani, Molino et al. 2001)	*Network-based:* a control network of actions and their composites	√	√	X	X

Manipulation of guideline knowledge and the information about its enforcement is important to allow flexibility and the ease-of-use of guideline support mechanisms. Flexibility and ease-of-use are the major determining factors in the acceptability of guideline systems by clinicians. In terms of the SpEM framework, the guideline systems and models reviewed in this section support mainly the specification and execution planes. With the exception of the

Asgaard/Asbru guideline system (Shahar, Miksch et al. 1998), most systems do not provide support for the manipulation plane.

3.4. Implications to this Study

The literature review revealed a number of important issues that need further research attention. First, it is necessary to develop a generic modelling and implementation framework and its associated specification and manipulation language for supporting the management of clinical guidelines/protocols. Second, instead of placing emphasis merely on the *specification* and *execution*, there is a need to comprehensively and explicitly support the *manipulation* (operations, querying) of the information on computerised guidelines/protocols and their executing instances. Thus, generic clinical protocols need not only to be declaratively specified, stored, and executed but also to be dynamically manipulated (i.e. operated on and queried) at the individual patient level, with both the specification and its instances being subject to the manipulation. Third, a clear line need to be drawn between *generic* guideline/protocol, and its *specific* instance. Most work within the clinical test-ordering protocol domain has concentrated mainly on developing expert systems that detect errors in test orders and abnormal test orders and test results and reason in order to issue alerts, reminders and pagers (Overhage, Tierney et al. 1997; Kuperman, Teich et al. 1999), without separating the guideline specification from other aspects of the system.

3.5. Summary

In summary, the literature review points to the need to address the limitations of current approaches to supporting the management of clinical guidelines by:

- supporting both the generic guideline knowledge and the specific instances of that knowledge resulting from the application of the generic knowledge to specific problem circumstances; and
- supporting dynamic customisation and manipulation to allow operations and querying of both the guideline knowledge and the objects, subjects and effects of its enforcement.

This work recognises that the problem of managing clinical guidelines as a type of the problem of managing knowledge and information for a given domain. This involves the information management tasks of acquisition, formal representation and specification,

storage, enforcement in solving domain problems, manipulation and dissemination. The development of a unified framework, a generic approach and its implementation mechanism for addressing the problem of the management of information for the case of clinical guidelines or protocols and similar applications is required. This work is an effort that is directed towards addressing this requirement.

Chapter 4 The Active Rule Paradigm and Active Database Systems

4.1. Introduction

In Chapter 2, the definition of the event-condition-action (ECA) rule paradigm was presented and the context of its use in this work for supporting clinical guidelines and protocols was also set. The introduction of the ECA rule paradigm in database systems was necessitated by the need to free individual applications from behavioural knowledge (Widom and Ceri 1996). This was achieved by pushing this knowledge into database management systems. Having behavioural knowledge in the database gives rise to *knowledge independence* because it freed applications from tasks like monitoring database events arising from activities or multiple users or applications, and periodically polling the database for events of interest (Paton and Diaz 1999).

An *active database management system* (ADBMS) is a database management system that incorporates an event-condition-action (ECA) rule mechanism and provides ECA rule support facilities that are stipulated in the Active Database System Manifesto (Dittrich, Gatziu et al. 1995). Passive database systems execute operations invoked in response to external requests from users or external applications. ADBMS extend passive ones by supporting ECA rules.

There is a subtle difference between active and reactive systems. On the one hand, active systems focus mainly on the task of monitoring changes in the state of a system through criteria evaluation that uses dynamic state data and information. On the other hand, the primary task of reactive systems is to coordinate activities through mainly real-time sensing of the environment for the attainment of some goal or state and usually functions with no explicit criteria evaluation. Knowledge-based systems differ from active and reactive systems in that their primary task is to reason using facts within the system in order to solve some problem. However, it is important to point out that the three tasks (monitoring, coordination, and reasoning) can overlap in each of the three systems. For clinical guideline management, systems that lie somewhere at the centre of active, reactive and knowledge-based systems are the most suitable. In healthcare, emphasis is placed more on assistance in

monitoring with the aim of prompting clinicians and issuing alerts and reminders as well as assisting with essential reasoning tasks that apply guideline knowledge in areas that benefit from computerisation. Furthermore, there is a need for tools to assist with monitoring, coordination, knowledge application and information exchange among clinicians and healthcare organisations. The ECA rule paradigm is capable of both monitoring and coordination as has been demonstrated in the literature (Eder, Groiss et al. 1994; Berndtsson, M., Chakravarthy et al. 1996).

The advantages of ECA rules in the form of triggers in database systems have been identified by Simon et al. (1995) and Appelrath et al. (1995). Triggers enable a uniform and centralised description and maintenance of domain knowledge such as business rules (Simon and Kotz-Dittrich 1995). The ECA rule paradigm provides a means to express event-action dependencies in active environments. In many application domains, event-action dependencies occur whereby an action is performed as a result of the occurrence of one or more events. For example, in a hospital, the action to *allocate a hospital bed* follows the occurrence of the *patient admission* event. By using the ECA rule paradigm, these event-action dependencies could be mapped directly into the system (Appelrath, H-J, Behrends et al. 1995). Triggers are reliable since they are automatically invoked whenever an appropriate event is issued by a transaction (Simon and Kotz-Dittrich 1995). ECA rule paradigm provides an opportunity for active behaviour to be modified and extended dynamically. This allows the customisation of an application. New behaviour and explicit control can be added when necessary (Appelrath, H-J, Behrends et al. 1995). Triggers are also expected to improve the performance of applications due to applicability of better optimisation techniques made possible by the centralisation of application semantics and their use as an effective tuning instrument to make applications run faster (e.g. by elimination of polling, and introduction of trigger-maintained materialised views) (Simon and Kotz-Dittrich 1995). Furthermore, exceptions can be represented as events in a system. Thus, the ECA rule paradigm provides an opportunity to handle exceptions in accordance with the users' expectations (Appelrath, H-J, Behrends et al. 1995).

This Chapter presents the state-of-the-art in the form of a review of the literature on the event-condition-action (ECA) rule paradigm and active databases (ADBs), the use of the ECA rule paradigm and active database guideline in various domains, and the use of the ECA rule paradigm and active databases in supporting clinical guidelines and protocols. The rest of the chapter is organised as follows: *Section 4.2* presents a review of the state-of-the-art in

the basic concepts and support of the ECA rule paradigm and active databases. *Section 4.3* presents *a* review of the applications of the ECA rule paradigm and active databases in domains other than the clinical guideline management domain. *Section 4.4* presents a review of approaches that make use of the ECA rule paradigm and active databases to support the management of clinical guidelines. *Section 4.5* summarises and concludes this chapter.

4.2. Dimensions of Active Behaviour

The main characteristics of active systems can be described in terms of the so-called *dimensions of active behaviour* introduced by Paton et al. (1999). These dimensions constitute a framework for describing active system functionality. The aspects of active behaviour that are characterised by these dimensions are the *knowledge model*, *the execution model*, and *rule management* (Paton and Diaz 1999). The following sub-sections outline the core concepts of active behaviour in terms of the dimensions of active behaviour. For a more detailed discussion of the dimensions of active behaviour and related concepts, the reader is referred to (Paton and Diaz 1999).

This model deals with what can be said about ECA rules in an active system (Paton and Diaz 1999). In other words, the knowledge model provides the structural characteristics of individual ECA rules as a means to define and specify the rules in an active system. The knowledge model have the following three main components (Paton 1999; Paton and Diaz 1999):

Event: The event specification defines the events that trigger the rule. An event occurs at a specific time point and is instantaneous. A rule can be processed immediately before or just after the occurrence of the event that triggers the rule. Events can be *atomic/primitive* or *composite*. The types of atomic or primitive events are: *database events* such as data manipulation events, transaction events, and method events; *time events*, which can be an absolute time point, a relative time point or periodic events; and *external or abstract events* originating from outside the system, e.g.; from users, external devices or application programs. *Composite events* are made up of primitive events combined using operators of the types: Boolean operators, history operators and interval-based operators.

Condition: is a predicate, which can be a database predicate, a database query that tests the existence of some data or information, or an external function that returns a Boolean value. A

rule condition may accept parameters from the event or pass its own parameters to the action of the rule.

Action: The action of a rule can perform a task such as updating the database schema or the rule-base, invoking some internal or external behaviour module, informing the user of some situation, aborting a transaction. The rule action can also use the *do-instead* construct to perform some alternative action.

4.2.1. Rule Execution Model

The execution model describes how a set of ECA rules are evaluated or handled by the active system at run-time. Paton and Diaz (1999) describe six dimensions of the execution model. **Coupling modes** is a dimension of the rule execution model, which determines when the ECA rule components are processed relative one another. The *event-condition coupling mode* determines when the condition is evaluated relative to the event that triggers the rule and this can be immediate, deferred or detached. The *condition-action coupling mode* determines when the action is executed relative to the evaluation of the condition and can be immediate, deferred, or detached. **Transition granularity** is a dimension that defines the relationship between the event occurrence and the number of rules it triggers. The transition granularity can be *tuple* when a single event occurrence triggers a single rule or *set* when several event occurrences are used to trigger a single rule. **Net effect policy** is a dimension that indicates whether or not the net effect of multiple event occurrences should be considered in triggering a rule. If the net effect is not considered, then each individual event occurrence is considered. **Cycle policy** is a dimension that determines what happens when events are signalled by the evaluation of the condition or execution of the rule action. The cycle policy can be *iterative* in which case rule execution is not suspended to allow responses to events signalled by the rule's condition or action. Alternatively, the cycle policy can be *recursive* in which case rule execution is suspended to allow response to events signalled by the rule's condition or action, i.e., rule triggering is allowed to cascade. The ***scheduling and priority*** dimension determines how multiple rules that are triggered simultaneously are handled. Principal tasks are the selection of the next rule to be fired and the determination of the number of rules to be fired. Rule selection can be made easier by assigning a priority to each rule using some priority mechanism. The last dimension is **Error handling,** which determines how errors that occur during rule firing is supported. Most modern database systems simply *abort* the transaction in which the error occurs. Alternatives include *ignoring*

the rule that causes the error, *backtracking* to the point when the rule started executing and restart or proceed with the transaction, or use some *contingency plan* to recover from the error, e.g., using an exception mechanism.

4.2.2. Rule Management Model

Paton and Diaz (1999) also introduced this class of dimensions of active behaviour, which deals with the operations that can be performed on rules, how the rules are represented, and programming support for the rules. Four dimensions are included in rule management (see Paton 1999). **Rule description** is a dimension that deals with how rules are specified. Rule description can be achieved by using a programming language, a query language (e.g., SQL), or objects**. Operations on rules** is another dimension of the rule management model. Mandatory operations are the create and delete rule operations. Other operations that may be supported include activate, deactivate and signal or fire a named rule. The dimension, **rule adaptability**, concerns when rules may be modified. Some systems allow modification of rules at *compile-time* others at *run-time.* Systems that support run-time adaptability may also allow rule actions to add or delete other rules. The **data model** constituting the rule environment is the last dimension for rule management. Since the data model associated with the active rule system affects the way the rule system is designed, it is an important dimension for rule management. When using ECA rules to support computerised clinical guidelines and protocols, ECA rule management is of special significance because, to support guideline knowledge management, the modelling and implementation primitives, the ECA rules, must be manageable on a full-scale. If rule management is fully supported in an active system, then using the ECA rule paradigm in the modelling and implementation framework for managing clinical guidelines would guarantee the full-scale manageability for the guidelines.

4.3. Active Systems

Modern database management systems support the ECA rule paradigm in the form of triggers e.g. Oracle, Ingres, Sybase, DB2, MS SQL Server, Informix, Interbase and AllBase. However, Li et. al (1999) observed that there was no modern DBMS that supports full active capability as stipulated in The Active DBMS Manifesto (Dittrich, Gatziu et al. 1995), although the premises of the active rule paradigm and database technology are now well

understood (Li and Chakravarthy 1999). Rule capability is provided in many modern systems, but the capability is not sufficient as it provides only basic triggering capability. In the next subsections, the DBMS *trigger systems or mechanisms* will be reviewed as a representative form of the ECA rule support.

4.3.1. ECA Rule Support in Modern DBMS's

In the SQL standard (Melton 2003), a trigger is a named ECA rule that is activated by a transition in the database state and must be created by using the CREATE TRIGGER statement, which is a specification of the trigger. A trigger specification consists of the trigger table, a triggering SQL operation (the event), a trigger condition, and a trigger action. The syntax of the creation of SQL Standard triggers is illustrated in Figure 4.

In the SQL Standard, trigger name must be unique within a schema and the subject table is required to be a base table (Melton 2003). The only triggering operations allowed are INSERT, DELETE and UPDATE statement.

```
<trigger-definition>::=CREATE TRIGGER <trigger-name> <trigger-action-time> <trigger-event> ON <table-name>
[REFERRENCING <old-or-new-value-alias-list>] <trigger-action>
<trigger-action-time>::=BEFORE|AFTER
<trigger-event>::=INSERT|DELETE|UPDATE [OF <column-name-list>]
<old-or-new-value-alias-list>::=<old-or-new-value-alias> …
<old-or-new-value-alias>::= OLD [AS] <identifier> | NEW [AS] <identifier> | OLD_TABLE [AS] <identifer> |
NEW_TABLE [AS] <identifer>
<trigger-action>::=[FOR EACH {ROW|STATEMENT}] [<trigger-condition>] <triggered-SQL-statement>
<trigger-condition>::=WHEN <left-paren> <search-condition> <right-paren>
<triggered-SQL-statement>::=<SQL-procedure-statement> | BEGIN ATOMC {<SQL-procedure-
statement><semicolon>} … END
```

Figure 4 The CREATE TRIGGER statement in the SQL Standard

The *trigger activation time* determines whether the trigger is activated before or after the triggering operation. The *trigger condition* is any SQL predicate, whose specification is not mandatory. The trigger granularity determines how many times the trigger is activated when its triggering operation executes and occurs at two levels: the *tuple- or row-level granularity*, which is specified by the FOR EACH ROW clause; and the *statement-level granularity*, which is specified by the FOR EACH STATEMENT clause. The transition tables and values are specified by the *OLD_TABLE/NEW_TABLE and OLD/NEW tuple references* respectively in order to allow the trigger action and condition to access the old and new states of the database. *Trigger priority* defines when a trigger is executed relative to other triggers. Although the SQL Standard trigger specification does not specify trigger priority, the

standard defines a default priority based on the time the triggers are created. Table 4.1 presents a summary of the support of these trigger features in the SQL Standard and the four main modern DBMS's: Oracle, Informix, DB2 and the Microsoft (MS) SQL Server.

It can be seen that Oracle and Informix offer the most comprehensive support while MS SQL Server offers the least support when all are compared with the SQL Standard. There are a number of limitations of the ECA rule support in DBMS trigger systems that have been identified in the literature (Kotz-Dittrich and Simon 1999; Li and Chakravarthy 1999).

Table 4.1 Trigger features supported by SQL3 and commercial database systems

Trigger Feature	SQL3	Oracle	Informix	DB2	MS SQL
Multiple events	N	Y	Y	N	Y
Trigger activation time	Y	Y	Y	Y	N
Condition present	Y	Y	Y	Y	N
Tuple-level granularity	Y	Y	Y	Y	N
Statement-level granularity	Y	Y	Y	Y	Y
Old/New tuple references	Y	Y	Y	Y	N
Old/New table reference	Y	N	N	Y	Y
Priorities	Y	N	N	N	N
Cascaded triggering	Y	Y	Y	Y	Y
Self triggering	N	Y	Y	N	Y
Maximum cascaded/self triggering depth	00	32	61	00	32
Explicit Authorisation	Y	Y	Y	Y	Y
Time Events	N	N	N	N	N

As can be seen from Table 4.1, there is a lack of support for time and temporal events, which are important in healthcare in general and in supporting the execution of clinical guidelines in particular. Furthermore, complex data definition is not allowed within trigger actions. While trigger actions are allowed to call stored procedures within the DBMS, only atomic values may be passed as parameters to these stored procedures. Another limitation of database triggers is that there is no direct access to other programs or external systems in the underlying operating system. The support for events in database triggers is limited to database operations INSERT, DELETE and UPDATE, which cannot be applied to more than one table. An event can not be named as a separate logical entity. Although one trigger can combine these events using the OR-operator, more meaningful composite events cannot be specified. User-level facilities to manipulate triggers are not directly available. Table 4.2 presents the review findings for support for the manipulation of database triggers.

All DBMS's support the creation and deletion of triggers. Only compile-time modification is supported by the Oracle and MS SQL Server DBMS's. Activation and deactivation are not explicitly supported except by the Oracle DBMS.

Table 4.2 Trigger management features supported by SQL3 and modern DBMS's

Management Operation on Triggers	SQL	Oracle	Informix	DB2	MS SQL Server
Creation	Y	Y	Y	Y	Y
Deletion	Y	Y	Y	Y	Y
Modification OR replacement	N	Y	N	N	Y
Query	N	N	N	N	N
Activation/ Deactivation	N	Y	N	N	N
Signal from external event sources	N	N	N	N	N
Separation of action execution and triggering processor or transaction	N	N	Y	N	N

Only one of the reviewed DBMS's, Informix, supports the separation of action execution and the triggering transaction. Furthermore, in all of these modern systems the only way to change rules is by recompiling the application code. In such systems rules are generally changed or modified at compile-time only. There is no support for dynamic management of ECA rules in all the modern systems.

4.3.2. ECA Rule Support in the Oracle DBMS

In this section, ECA rule support in the Oracle DBMS is reviewed. The Oracle database system has been selected here for a more detailed review because it provides more comprehensive ECA rule support than existing DBMS's.

The Dimension of the Knowledge Model in the Oracle DBMS

The knowledge model of the Oracle DBMS consists of three dimensions that correspond to the ECA rule components (Cyran 2002):

Event: The first part is the *triggering event* or *statement*, which can be one, two or all of DELETE, INSERT or UPDATE statement on the table. For an UPDATE triggering statement, affected columns can be optionally specified.

Condition: The second part of an Oracle trigger is the *trigger restriction*, which specifies a Boolean (logical) expression that must be true for the trigger to execute its action.

Action: The third part of an Oracle trigger in the *trigger action*, which contains the SQL statements and Oracle-specific language (PL/SQL) code to be executed when the triggering statement is issued and the trigger restriction evaluates to true. The trigger action can contain

SQL and/or PL/SQL statements, define PL/SQL constructs such as data structures and variables, and call stored procedures.

The Dimensions of Rule Management in the Oracle DBMS

Rule description: Oracle uses SQL and its extensions to describe or specify triggers. The first part of Figure 5 illustrates the syntax for specifying a trigger to be created in Oracle.

```
<Oracle trigger>::= CREATE [OR REPLACE]  TRIGGER< trigger name>
{BEFORE|AFTER}<trigger events>
On <table name>
[[REFERENCING<references>]
FOR EACH ROW
[WHEN<condition>]]
<PL/SQL block>
<trigger event>::=INSERT|DELETE|UPDATE {OF<column names>]
<REFERRENCE> ::=OLD AS <old value tuple name > |NEW AS <new value tuple name>
```

Figure 5 The syntax of a trigger in Oracle

The name of the trigger must be unique among all triggers within the database but not with respect to other schema objects such as tables.

Operations on rules: Oracle supports creation and deletion of rules. Modification and signalling operations on trigger are not explicitly supported.

Rule adaptability: Modifying an Oracle triggers can be achieved only by recompiling with the REPLACE option or by deleting the old trigger and creating a new one to take its place (Russell 2002). Consequently, Oracle supports the lowest level of trigger adaptability, i.e., compile-time adaptability.

Data model: A The Oracle database system supports the object-relational data model (Gietz and Dupree 2002). However, since Oracle triggers are defined on relations or tables (Cyran 2002), the effective data model for Oracle triggers is the relational model.

The Dimensions of the Rule Execution Model in the Oracle DBMS

Oracle uses the execution model whose algorithm is given in Figure 6 to maintain the proper firing sequence of multiple triggers and constraint checking. A single SQL statement can potentially fire up to four types of triggers: BEFORE row triggers, BEFORE statement triggers, AFTER row triggers, and AFTER statement triggers (Cyran 2002). A triggering statement or a statement within a trigger can cause one or more integrity constraints to be checked. Also, triggers can contain statements that cause other triggers to fire giving rise to

cascading triggers (Cyran 2002). An important property of the Oracle execution model is that all actions and checks done as a result of a SQL statement must succeed. If an exception is raised within a trigger, and the exception is not explicitly handled, all actions performed as a result of the original SQL statement, including the actions performed by fired triggers, are rolled back (Cyran 2002). Thus, triggers cannot compromise integrity constraints. The Oracle execution model takes into account integrity constraints and disallows triggers that violate declarative integrity constraints. It is important to be aware that triggers of different types are fired in a specific order. However, triggers of the same type for the same statement are not guaranteed to fire in any specific order (Cyran 2002).

1. Execute all BEFORE statement triggers that apply to the statement.
2. Loop for each row affected by the SQL statement.
    i) Execute all BEFORE row triggers that apply to the statement.
    ii) Lock and change row, and perform integrity constraint checking. (The lock is not released until the transaction is committed.)
    iii) Execute all AFTER row triggers that apply to the statement.
3. Complete deferred integrity constraint checking.
4. Execute all AFTER statement triggers that apply to the statement.

Figure 6 Algorithm for the Oracle trigger and constraint execution model (Cyran 2002)

For example, all BEFORE row triggers for a single UPDATE statement may not always fire in the same order. As a result, applications must be designed in such a way that they should not rely on the firing order of multiple triggers of the same type.

4.3.3. Limitations of the Oracle Trigger Mechanism and their Implications to this Study

ECA rule capabilities of the Oracle trigger mechanism have a number of restrictions. The following is an outline of the restrictions together within the implications to this study drawn from the clinical environment:

1. Oracle row triggers cannot access the table being altered by the triggering transaction. This is called the *mutating table (MT) problem* (Russell 2002).

   Implication I: This problem forces the use of set or statement level triggers, which can access the table being altered but cannot access the OLD state of the database. Only row triggers are able to refer and access the past state of data (Russell 2002). This may force the translation of ECA rules into row triggers. An example of a clinical domain rule that cannot be implemented due to the MT problem is: *When a new clinical lab result arrives,*

retrieve the last two results, for the same test for that patient, and determine if 2 of the 3 most recent results are above a stipulated value. Section 9.4 (c) describes how this limitation can be addressed.

2. Oracle triggers, by their definition (Cyran 2002), monitor only one table. The same trigger cannot monitor operations on several tables in a database.

   Implication II: In the clinical environment, the execution of one action may depend on a logical condition that involves attributes from more than one table. This forces one to create a distinct trigger for each table to be monitored and some form of a convergence mechanism to combine the results from the distinct triggers. An explanation of how this limitation can be resolved is found in Section 9.4 (c).

3. In Oracle, trigger processing is immediate (Cyran 2002) and only the immediate processing mode is supported.

   Implication III: Rules requiring triggers to be processed some time after and detached from the triggering transaction cannot be implemented. The only option is to adapt such rules to the immediate coupling mode. The reader is referred to Section 9.4 (c), for an explanation of how this limitation can be addressed.

4. Oracle triggers are executed in a fixed order and always have a lower priority than integrity constraints. In other words, triggers that violate integrity constraints can be prevented from executing and cause the whole transaction to rollback.

   Implication IV*:* This results in the possible occurrence of interference within trigger processing or between triggers and built-in constraints. This may have undesirable effects, if the DBMS trigger mechanism is to be used as a CGP execution engine. See Section 9.4 (c) for a description of the strategy for voiding possible problems that could arising from this limitation during protocol execution.

5. In Oracle, as in most commercial database systems, events are restricted to database operations on tables. Also atomic events can only be combined with the OR operator.

   Implication V: This means that any conceptual or domain-dependent event, such as *patient admission* or *discharge,* needs to be mapped to or represented by database operations on tables. This study has not addressed the important issue of a comprehensive event algebra to support composite events. This is left to future work.

6. Oracle triggers lack communication functionality with their environment.

Implication VI: Oracle triggers cannot control the processing of external actions or tasks. Also synchronising external actions, tasks or programs with Oracle triggers is difficult, especially when an ECA rule action is common to several tasks.

7. In the Oracle DBMS, as in most other DBMS's, management operations on triggers are part of the Data Definition Language (DDL) (Cyran 2002; Russell 2002) as opposed to Data Manipulation Language (DML).

   Implication VII: This limits the ability to effectively perform operations on and issues queries against triggers or ECA rules by both applications and users at run-time.

8. In the Oracle DBMS and other DBMS, triggers that specify time and/or temporal events are not supported.

   Implication VIII: All ECA rules that involve temporal events cannot be directly implemented by using the trigger mechanism of the DBMS.

4.3.4. Discussion

From the clinical guideline domain point of view, the ECA rule paradigm is useful in that it can be used to express guideline knowledge and enforce it using the ECA rule mechanism of a database system that holds the electronic patient record and the patient management workflow information. The specification and execution of ECA rules are supported, in a limited way, in modern database systems, such as Oracle 9i, where they are commonly referred to as *triggers*. To use these modern database systems to implement the ECA rule paradigm, it may be necessary to build extensions or enhancements to address the limitations of existing facilities in the systems and call upon system vendors to incorporate generic aspects of application requirements into these systems.

4.4. Applications of the ECA Rule Paradigm and Active Databases

ECA rules have been used for *database system extensions* such as supporting integrity constraints (Widom and Ceri 1996), for closed database applications such as monitoring sales in a stock control database (Simon and Kotz-Dittrich 1995), and for *open database applications* in which there is need to respond to situations outside the database such as warning clinicians of changes in patient's condition. As a result, applications of the ECA rules in databases are also commonly classified into two. *Internal applications* extend the

functionality of databases. Examples of such applications include: implementations of advanced transaction models (Geppert, A. and Dittrich 1993); dynamic displays of database objects (Diaz, Jaime et al. 1994); and database system monitoring and tuning (Graeser 1994). *External applications* use the ECA rules in active databases to support domain-specific behaviour that requires situation monitoring and reaction. Examples of such applications include: computer integrated manufacturing (Berndtsson, M. 1994); coordinating knowledge and discovery algorithms in a dynamic environment (Kawano, Nishio et al. 1994); software development process control (Jasper 1994); banking environments (Simon and Kotz-Dittrich 1995); and workflow and process management (Eder, Groiss et al. 1994).

From functional and behavioural points of view, Ceri at al. classified triggers into the following nine types: constraint-preserving, constraint-restoring, invalidating, materialized, meta-data, replication, extenders, alerters, and ad-hoc triggers. Ceri et al. further observed that for the nine trigger types and many applications, the primary purpose is to monitor and maintain some kind of constraint. This is in agreement with the observation that active systems's primary task is that of monitoring as opposed to coordination – the primary task of reactive systems -and reasoning – the primary task of knowledge based systems.

4.4.1. Applications of ECA Rules in Database Systems

Recently, Ceri et al (2000) noted that business rules, scheduling, supply chain management, web applications and workflow management constitute the majority real-world applications of active databases. ECA rule paradigm has been applied to a wide variety of applications in an equally wide variety of domains. Schwiderski (1996) used ECA rules with both primitive and composite event semantics that are based on the notion of "physical time" in a distributed environment to monitor the behaviour of distributed components of a system. To implement declarative conceptual integrity rules found in the development of information systems, Wu (1996) used the ECA rule paradigm based on an active mechanism of a database system . ECA rules in an OODBMS, the O2 System, have been used to support tasks such as: user notification, application access logging, organising related domain objects (e.g programs), tools communication, change propagation, and maintaining data consistency in the framework of the GOODSTEP project (Collet, Habraken et al. 1994). The GOODSTEP Project's main aim was to create a computer-aided software engineering platform. The AI community has investigated static and dynamic coordination protocols among agents. The database community has investigated system level support for coordination in distributed/federated databases and the specification and execution of relaxed notions of

transactions/activities. In an effort to combine these two approaches, Berndtsson et al. (1996) used pre-defined and dynamically created ECA rules to coordinate static and dynamic plans in the domain of cooperative problem solving. An ECA rule mechanism coupled to a relational database was used to detect cancer clusters in tumour registries (Appelrath, HJ, Behrends et al. 1994). ECA Rules were used to detect relevant events that determined when it was necessary to generate hypobook on clusters of cancer cases in both time and space. Kawano et al. (1994) integrated active database technology with machine learning techniques by using ECA rules as a data sampling and knowledge discovery initiators. The ECA rules triggered the data sampling or knowledge discovery process based on the importance or freshness of facts in the system (database). The rules in the database were also used to perform knowledge rule verification, modification and invalidation when certain conditions were detected in the knowledge discovery process (Kawano, Nishio et al. 1994). Kawano et al. (1994) also noted that the specification, refinement and assessment of appropriate conditions and actions of ECA rules may need the use of knowledge discovery tools, i.e., knowledge-assisted ECA rule specification. In a GUI used to display database objects, Diaz et al. (1994) updated dynamic displays automatically as changes occur to the database objects being displayed. ECA rules, being declarative and modular, were used to allow the support of dynamic displays with minimal changes to the GUI and the underlying database. Diaz et al. (1994) also used ECA rules to support dynamic interaction between the database system and external applications. In environmental systems there is a need for providing knowledge for reacting to certain situations that depend on measurement values. Gutleber et al. (1997) used ECA rules in a real-time database to reduce flooding of data from measuring instruments to central stations and to support the management of different alarm prescriptions on these stations. Eder et al. (1994) expressed workflow specifications in a graphical language, compiled them into ECA rules, and executed them in an active database-based system, thus, allowing dynamic execution of workflows to be handled by triggers of an active database system. In another effort in workflow management, Ceri at al. (1997) used ECA rules to support exception handling. Events of interest included data events (modifications to workflow data), external events (raised by external applications), workflow events (describe workflow evolution or progress in execution), and time events (absolute or relative time points). Data events were captured by low-level triggers installed in the DBMS (Ceri, S. , Grefen et al. 1997).

4.4.2. Discussion and Implications

The ECA rule paradigm in active databases, as can be observed from the literature and as noted by Ceri et al. (2000), is primarily used to address the problem of monitoring some form of constraints that can be expressed as logical criteria. Most of the applications are external applications in which the ECA rules are used to support the management of domain-specific knowledge. The work of Berndtsson et al. (1996) illustrates that ECA rules can support both monitoring and coordination tasks such as patient monitoring and patient workflow management (termed, *care flow*) respectively. The use of knowledge discovery techniques for automatic rule specification, refinement and assessment by Kawano et al. (1994) is interesting from the point of view of managing clinical guidelines modelled and implemented using ECA rules and active technology. The use of ECA rule as information filters (Gutleber, Schimak et al. 1997) can assist in addressing the problem of information overload experienced by clinicians in data intensive healthcare domains such as intensive care units (ICUs). Of special interest to this study is a new application domain for active databases, which addresses the problem of managing information and knowledge in clinical guidelines. Guideline knowledge can be represented as ECA rules. Such an application of ECA rules would fall under the type, *ad-hoc* triggers, identified in (Ceri, S., Cochrane et al. 2000). This new application may bring into light further demands for the incorporation of special ECA support requirements into modern DBMS. For instance, some of the important requirements from the healthcare application domain are comprehensive and high-level facilities for modularising, querying and dynamically manipulating ECA rules. The dynamic manipulation of the rules should occur through other ECA rules and either automatically with user concurrence or manually.

4.5. Use of ECA Rules and Active Databases to Support the Management of Clinical Guidelines and Protocols

Of special interest to this study are the guideline support approaches that make use of the ECA rule paradigm in database systems. The most significant effort that apply the ECA rule paradigm in supporting clinical guidelines/protocols are: the Arden Syntax and Medical Logic Modules (MLMs) (Jenders, R.A., Hripcsak et al. 1995); and HyperCare (Caironi, Portoni et al. 1997).

4.5.1. The Arden Syntax for Medical Logic Modules

The *Arden Syntax* is a language for encoding medical knowledge bases that consists of independent modules called the *medical logic modules* (MLMs). The Arden Syntax and MLMs constitute the first approach that made use of the ECA rule paradigm to support medical knowledge management. The Arden Syntax is currently the only standard for sharing and encoding medical knowledge among systems in various medical institutions (ASTM 1992; HL7 1999), which is an indication of the promise the ECA paradigm has as a viable technology.

A MLM is essentially an ECA rule, which is stored as a separate *ASCII file*. Each MLM is organised as a set of statements, called *slots,* which are categorised into *maintenance* information, *library* information, and the actual medical *knowledg*e (Clayton, Pryor et al. 1989). The maintenance category of slots hold information about the MLM such as title, filename, version, author, organisation and date. The library category of slots hold information that is important in archiving, searching and retrieval of the MLM such as its purpose, keywords, explanation and optional items such as links and citations. The knowledge slot is expressed in the ECA rule format and is the core of a MLM.

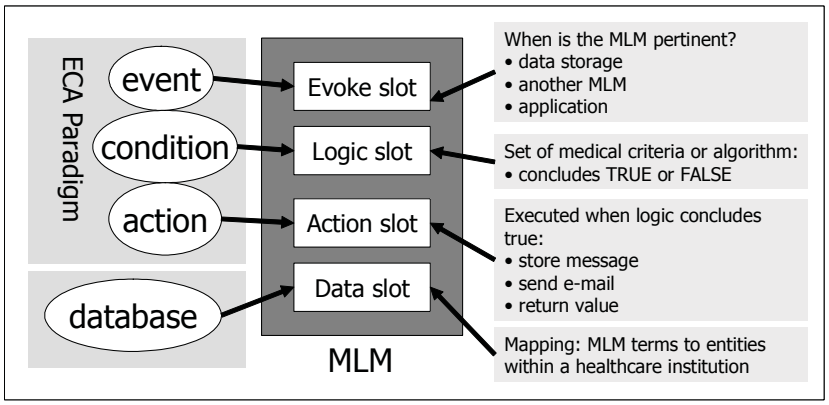


Figure 7 The core slots in the *knowledge* category of a Medical Logic Module (MLM) and the event-condition-action (ECA) rule paradigm

Figure 7 illustrates the core slots in the knowledge category of a MLM and how these slots relate to the components of the ECA rule paradigm. Other slots that are not shown in Figure 7 are the type, priority and urgency slots, whose purpose have been described the ASTM Standard (1992), which specifies the Arden Syntax for MLMs. Of interest to this study are the knowledge slots that form the basis of the execution of a MLM according to the ECA rule paradigm. As illustrated in Figure 7, the **evoke slot** specifies the events that trigger an

MLM execution. Examples of events include the passage of time, arrival of a piece of information and invocation by another MLM. The evoke slot corresponds to the event in the ECA rule paradigm. The **logic slot** specifies a set of medical criteria, which ends with one of two possible *conclude* statement: either *conclude true* or *conclude false*. The logic slot corresponds to a condition in the ECA rule paradigm. The **action slot** specifies the action that must be carried out if the logic slot concludes true. The action slot corresponds to the action in the ECA rule paradigm. The **data slot** maps terms in the MLM to medical record attributes in a database. The data slot corresponds to the database link, which is implicit in the ECA rule paradigm.

Figure 8 presents a example MLM taken from the MLM Library (Scherpbier 1995) of the Columbia-Presbyterian Medical Centre, New York City. This MLM is also a sample used for the ASTM standard for the Arden Syntax for MLMs (ASTM 1992). The medical relevance, accuracy or semantics of this MLM are not important here. The focus of this study is on the use of the ECA rule paradigm, by using the Arden Syntax, to express and enforce medical knowledge within a MLM. The MLM is triggered by a physician's order for a *CT study with contrast*, ct_contract_order. Once triggered, the MLM checks if the patient, for whom the order was made, has renal failure, and if so, the MLM issues an alert. The alert is intended to prompt the clinician to consider alternative contrast dyes instead of ordering the CT study with contrast, which is not suitable for patients with renal failure. The MLM uses previous results for the serum creatinine tests that were performed on the patient in order to determine the renal condition of a patient.

The MLMs have been applied to generating alerts, patient management suggestions, management critiques and diagnostic scores for a wide variety of clinical domains. Attempts have also been made to build complex care plans and clinical guidelines/protocols by chaining MLMs in such a way that the action of one MLM evokes the next MLMs (Starren and Xie 1994; Sherman, Hripcsak et al. 1995; Sailors, Bradshaw et al. 1998).

Since MLMs specifications are stored as individual text files, they cannot be queried or easily manipulated. For instance, in a study to quantify changes that occur as an MLM knowledge base evolves, 156 MLMs developed over 78 months were studied and 2020 distinct versions of these MLMs were observed. It was also found out that 38.7% of changes occur primarily in the logic slot while 17.8% and 12.4% of the changes occur in the action and data slots respectively (Jenders, R.A, Huang et al. 1998). In another study, it was found out that changes in laboratory testing can cause disruptions in MLM execution unless the

code of these MLMs is revised and modified (Jenders, R.A., Hripcsak et al. 1995). As a result, a limitation of the Arden Syntax, which is important and of interest to this work, is the lack of support for the manipulation, and querying and hence for maintenance of the MLMs specifications.

```
maintenance:
        title:  CT study with contrast in patient with renal failure;;
        filename:  astm_ct_contrast;;
        version:  1.00;;
        institution:  ASTM E31.15; SMS;;
        author:  Harm Scherpbier, M.D.;;
        specialist:  ;;
        date:  1995-09-11;;
        validation:  testing;;

library:
        purpose:
                Issue alert when physician orders CT study with contrast in patient with renal failure;;
        explanation:
                If physician orders CT scan with contrast, this rule retrieves most recent serum creatinine.  If the value
                is less than 1 week old, and more than 1.5, the system issues an alert to the physician to consider the
                possibility that his patient has renal failure, and to use other contrast dyes.
                ;;
        keywords:  ;;
        citations:  ;;
        links:  ;;

knowledge:
        type:  data_driven;;
        data:
                last_creat := read last {"Creatinine level"};
                last_BUN := read last {"BUN level"};
                ;;
        evoke:  ct_contrast_order;;
        logic:
                if last_creat is null and last_BUN is null then
                        alert_text := "No recent serum creatinine available.  Consider patient's kidney function
                        before ordering contrast studies.";
                        conclude true;
                elseif last_creat > 1.5 or last_BUN > 30 then
                        alert_text := "Consider impaired kidney function when ordering contrast studies for this
                        patient." ;
                        conclude true;
                else conclude false;
                endif;
                ;;
        action:
                write alert_text || "\nLast creatinine: "||last_creat|| on: "||time of last_creat || "\nLast BUN:
                "||last_BUN|| on: "||time of last_BUN ;
                ;;
        urgency: 50;;
end:
```

Figure 8 A example Medical Logic Module (MLM) in the Arden Syntax: CT Study With Contrast in Patients With Renal Failure (Scherpbier 1995)

4.5.2. The HyperCare Guideline System

HyperCare (Caironi, Portoni et al. 1997) is a prototype system that employs the ECA rule paradigm in the active object-oriented database, Chimera, to capture medical knowledge. HyperCare is the first guideline system to use an active database system for guideline management support. HyperCare does not provide a generic protocol specification model and was created specifically to manage a domain- and organisation-specific guideline for a specific medical condition. Consequently, example clinical protocol specifications used by HyperCare could not be found. HyperCare was designed solely for supporting clinical guideline compliance in the domain of essential hypertension.

The architecture of HyperCare consists of an object-oriented schema and an active computational paradigm implemented through ECA rules. The entities that make up the hypertension treatment domain are represented by an object-oriented schema through object classes. Such entities include physician, patient, drug, test, and visit. The ECA rules representing guideline knowledge are stratified into the following stratum in their order from top to bottom with rules in a higher stratum generating events that trigger rules in a lower stratum: a start stratum, diagnosis stratum, start-therapy stratum, decision stratum, increase-decrease dosage stratum, add-drop rule stratum, consistency (integrity constraints) stratum, patient visits stratum. The strategy for the stratification is based on the *event-based stratification model* (Ceri and Ramakrishnan 1996) that imposes modularisation, readability, maintainability and guarantees termination of the rules (Caironi, Portoni et al. 1997). The limitations of HyperCare are:

- The difficulty in managing the rules making up the protocol;
- The lack of support for dynamic manipulation, querying, versioning and customisation of clinical protocol specifications and instances; and
- It is an implementation of a specific guideline and does not attempt to provide a generic formalism to support similar protocols.

4.5.3. Review Findings

Arden Syntax and HyperCare both make use of the ECA rule paradigm to specify clinical protocols. The Arden Syntax allows the generic clinical protocols to be specified and executed. Protocol specifications are stored as programming language code in text files. Furthermore, there is no flexible support for the management of both specifications and their instances. HyperCare does not support the creation of generic clinical protocol specifications. Instead, the system was built for a specific clinical protocol, which it implements using ECA rules of an active database system. Both the Arden Syntax and HyperCare do not create patient-specific instances. Instead, rules in a protocol operate at a global level or have a global scope covering all patients.

4.6. Summary

This Chapter has presented the state-of-the-art in the ECA rule paradigm and active database systems. The main concepts and support for the ECA rule paradigm and active behaviour were presented. A review of the applications of active behaviour in various domains and in

the support for the management of clinical guidelines was undertaken. The study focuses on investigating the use of the ECA rule paradigm as a unifying concept that can be incorporated into both the conceptual modelling and the implementation frameworks of the management clinical guidelines or protocols. The ECA rule paradigm would offer the opportunity to make use of existing ECA mechanisms in modern database systems. Further benefits would be that the ECA rule mechanism can be combined with other existing technologies, such as web technologies and database systems, for supporting integration with medical vocabularies and the electronic medical record. In this Study, an approach that allows the management of ECA rule-based clinical protocols is adopted. The approach allows generic clinical protocols to be declaratively specified, stored, executed and dynamically manipulated. Both the specification and its instances are manageable on a full-scale.

PART II: APPROACH AND FRAMEWORK

Chapter 5 Framework and Approach for Supporting the Management of Clinical Protocols

5.1. Introduction

Clinical protocols contain domain knowledge that represents best practice in healthcare. The problem of incorporating clinical protocols into the daily routine used by clinicians is a subject of special interest in Healthcare Informatics and offers interesting challenges to the domain of computing and knowledge management. This chapter presents the generic framework, approach and method developed for supporting the management of clinical protocol or guideline information and knowledge. Full support for the management of clinical protocols can be provided in terms of clinical domain information and knowledge capturing, modelling, specification, storage, execution, manipulation and dissemination.

The rest of this chapter is organised as follows: Section 5.2 presents the problem of supporting the management of clinical protocols and the challenge that the problem poses. For the purpose of further clarification, Section 5.3 presents a brief review of the SPEM framework, which was introduced in Chapter 3, for supporting the management of clinical protocols. Section 5.4 presents the approach for managing clinical protocols and discusses the use of the event-condition-action rule paradigm within the SPEM framework. Section 5.5 presents the method for managing clinical protocols by first presenting the process for managing clinical protocols, then showing how this process fits into the framework, and, finally, identifying the enabling technologies that are necessary to accomplish the tasks that are described in the processes. Section 5.6 presents a discussion and review of related work. Section 5.7 summarises this chapter.

5.2. Problems and Challenges in Supporting the Management of Clinical Guidelines and Protocols

Ensuring clinician's compliance to clinical guidelines is a multi-faceted problem that involves, among many other aspects, cultural issues such as "cookbook medicine". IT support

is only one aspect to the solution of the problem of ensuring compliance to CGPs. As a contribution to this solution, there is a need to support and facilitate clinical protocols through the use of computer-based mechanisms. Figure 9 illustrates the main aspects of the problem. At the core of the problem, there is domain knowledge that exists mainly in the form of text based guidelines and human expertise. This domain knowledge needs to be captured and expressed in a generic format in order to allow its general usage and manipulation. To apply the knowledge to a specific problem requires that the knowledge be enhanced through customisation using clinical knowledge (patient data) in order to be applicable to the specific problem situation. As part of this problem, there is a need for a specification model and language and an execution and manipulation models and mechanisms. There is also a need to provide support for the full-scale management of this knowledge. Here *full-scale management* means that the knowledge and information must be specifiable and executable with the output of each of these aspects being able to be manipulated, that is, to perform operations and to issue queries. These requirements constitutes the core of the problem of the management of information and knowledge for supporting CGPs. In order to support the full-scale management of CGP knowledge, a number of aspects need to be addressed. The domain knowledge need to be specified. In order to provide support for managing clinical protocol, the protocol knowledge must be captured into a generic and formal specification. This requires the use of a formal specification model and language.

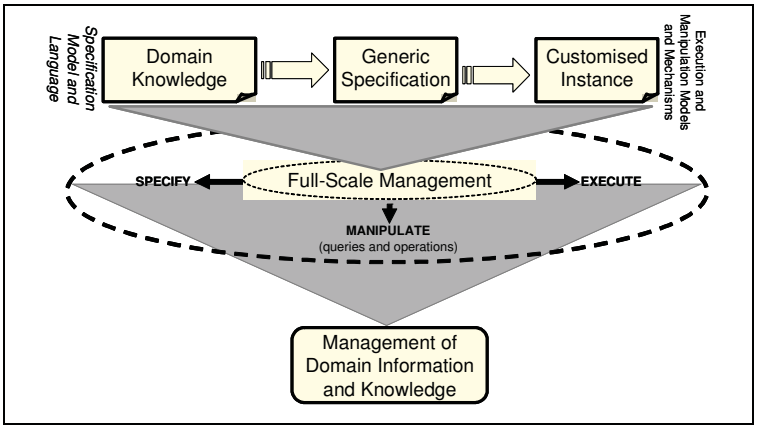


Figure 9 Aspects of protocol knowledge management

Once the specified domain knowledge in the form of the generic specifications are created, it needs to be stored. The method of the storage must allow the knowledge to be manipulated and queried. Before the knowledge can be used, it needs to be customised to suit existing

circumstances. The protocol knowledge needs to be applied to a specific problem situation. This requires the generic specifications to be customised or augmented with specific characteristics of the problem. For instance, domain knowledge in the form of clinical protocols needs to be customised at two important levels: the organisational level and the patient level. This customisation or augmentation process leads to the creation of the operational instance of the protocol. It is also important for the generic protocol specifications to be expressed using a formal model and language in order to make it possible to execute the customised instance of the protocol using a suitable computer-based execution mechanism. Furthermore, in order to achieve the full-scale management of knowledge in clinical protocols, provision must be made for the clinical protocols to be specified, executed and manipulated. The manipulation of both the specifications and execution aspects of the clinical protocols include three aspects. The first aspect involves performing operations on the knowledge and the effects of the knowledge's application. The second aspect involves querying or browsing the knowledge and the results of its application. The third and last aspect involves disseminating the knowledge and the results of its application to relevant places. Providing for the specification, execution and manipulation of domain knowledge and information in clinical protocols insures that the mechanism for supporting clinical protocols facilitates the incorporation of CGPs into daily clinical practice.

The complexity of information and knowledge management in the support for clinical protocols poses a number of challenges. *First,* a clinical protocol is a complex object that has multiple views. The protocol has both static and dynamic aspects that are also evolutionary in nature. The protocol is information that can be viewed from both a maintenance and usage viewpoint. The protocol is required to exist at both the generic and specific levels of information requiring transformations/translations back and forth between these two levels.

Second, the operations of addition, deletion and modification on parts of the specifications lead to the need to support versioning. Operations on specifications and patient plans give rise to the need for keeping the two in synchrony, that is, change propagation between specifications and patient plans, which are the instances generated from the specifications.

Third, a patient plan goes through the processes of creation, execution, manipulation and termination through its life. Termination occurs on completion of execution or truncation of the patient plan. During its life, the plan changes with time. Furthermore, due to these changes throughout its life, the patient plan becomes a complex entity whose state and

composition at time t1 may be different from those at time t2. An interesting challenge is to make these aspects of the plan subject to queries along the time.

Fourth, the protocol is a complex entity in the sense that it is composed of entities, which may also be complex. Furthermore, a protocol instance has a specification as well as an executing process. In other words, the patient plan has a static and dynamic aspect in the sense that it is an executing process and has a retrievable specification, which is independent of the executing process. When the operations are allowed to be performed on the static or dynamic aspects of a patient plan, any changes introduced must be propagated between the static and dynamic aspects.

Finding the solution to the problem of providing computer-based support for the management of computerised clinical guidelines entails the following:

- Developing an expressive and formal representation model for the clinical protocol knowledge;
- Automation of the enforcement of the protocol knowledge which is made possible by a formal model and representation model and language; and
- Sharing of the protocol knowledge, which is enabled by methods and mechanisms for customising of the knowledge to suit local and specific circumstances and distribution of the knowledge to locations where it is needed to be applied.

5.3. Review of the SpEM Framework for Managing Clinical Protocols

The framework for the management, i.e., the **Sp**ecification, **E**xecution and **M**anipulation, of clinical guidelines and protocols, **SpEM**, has been introduced in Chapter 3, Section 3.3. The aim of the SpEM framework is to support the full scale management of domain knowledge for computer-based clinical protocols. By full-scale management is meant the specification, execution and manipulation of the domain knowledge. Manipulation involves performing predefined operations and querying. The aspect of interest to this research, which has received little attention in the literature, is that of enabling these protocol specifications and their executing instances to be manipulated through operations and queries. In other words, the static specification and dynamic process of the protocol should be easy to manage. Since the SpEM Framework for supporting protocol management has been introduced in Chapter 3, this Section only briefly reviews the framework and provides some further explanation.

The major aspects of the framework of the management of protocol information are illustrated in Figure 10. The three planes, namely the specification, execution and manipulation of the protocol specifications and their instances constitute the core of the framework. Protocol instances are the individual patient care plans. Protocol specifications are created in the *specification plane*. In the *execution plane*, the customisation of protocols produces patient plans, which are then instantiated and executed. The protocol specifications and their instances are operated on and queried in the *manipulation plane*.

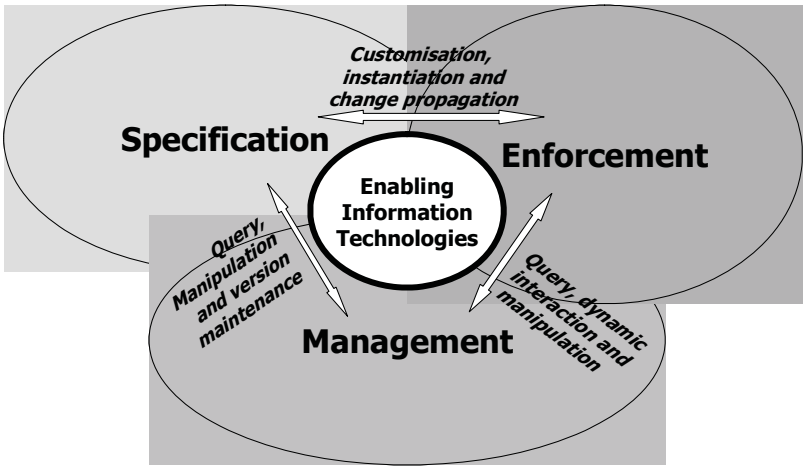


Figure 10 Main aspects of the SpEM framework for supporting the management of clinical protocols

The interaction between the specification, execution and manipulation planes of the framework consist of the manipulation of protocol specifications; the translation of protocol specifications to executing patient plans, which involves the customisation of protocol specifications and the enactment/execution of protocol instances; the manipulation of the executing protocol instances. At the core of the management planes are the enabling technologies that are based on the information technologies for supporting tasks in each of the three planes.

Within the SpEM Framework, the ECA rule paradigm is used in a number of ways. The model for the protocol specification uses the ECA rule paradigm as the main knowledge representation construct. The method of execution of protocol instances uses the ECA rule

mechanism that has been described in the active database literature (Dittrich, Gatziu et al. 1995). The mechanism for performing operations and querying specifications and their instances, while not based on the ECA rule paradigm, can be triggered by an ECA rule mechanism. For example, an ECA rule fired during the execution of a protocol can execute a task that may involve operations and queries that constitute manipulation within the framework.

Most frameworks in the literature incorporate the process of translating clinical protocols into formal specifications that are expressed in especially designed formal languages. Also, in these frameworks, some form of storage or persistence mechanism of the protocol specifications is provided. However, most existing frameworks do not pay much attention to the manipulation and querying of the stored protocol specifications. A mechanism for executing protocol instances is provided in almost all the works found in the literature. What makes the SpEM framework developed in this study stand out from other solutions is the emphasis on the manipulation and querying of both the stored protocol specifications and the executing protocol instances. The SpEM framework's uniqueness is based on that it addresses the problem of computer-based clinical protocol management in terms the three aspects of specification, execution and manipulation for supporting the management of clinical protocols. Most approaches found in the literature address specification and execution only and pay little or no attention to the manipulation aspect of clinical protocol management.

5.4. The MonCooS Approach to Supporting Clinical Protocol Management

The approach developed in this study for the management of information and knowledge in supporting computer-based clinical protocols has been named **MonCooS**, an acrogym for (**Mo**nitoring, **Coo**rdinating and providing **S**uggestions) The approach focuses on **Mo**nitoring, **Coo**rdinating and providing **S**uggestions to the clinicians. In the literature, the common practice is to make use of AI methods that strongly emphasise on assisting to domain experts with the task of reasoning and/or problem-solving (Musen, M.A. , Tu et al. 1992; Miksch 1999). The MonCooS approach makes use of protocol information in *monitoring* patient conditions and *coordinating* interventions for purposes of *suggesting* further appropriate clinical interventions such as ordering appropriate clinical laboratory tests whose outcomes

are also monitored. The aim is to provide a tool that assists domain experts while allowing them to perform the reasoning task.

The ECA rule paradigm plays a crucial role in the MonCooS Approach. First, an important advantage of making the MonCooS Approach database-based is that modern database systems already support, in a very basic way, the mechanism for monitoring and coordination in the form of the active rule mechanism. In other words, in a modern database system forms the basis for an execution engine for clinical protocols. Second, by being database-based, the approach can harness the excellent facilities available in database systems for manipulating information in the tasks of monitoring and coordinating. Third, a further advantage of the MonCooS approach being database – based is that future sharing of information is guaranteed by the generic nature of databases, e.g., tools already exist to map data from databases to XML for information exchange between systems.

The process illustrated in Figure 11 allows clinical protocols to be formally specified, stored, enforced or applied in problem solving, and manipulated through querying and operations.

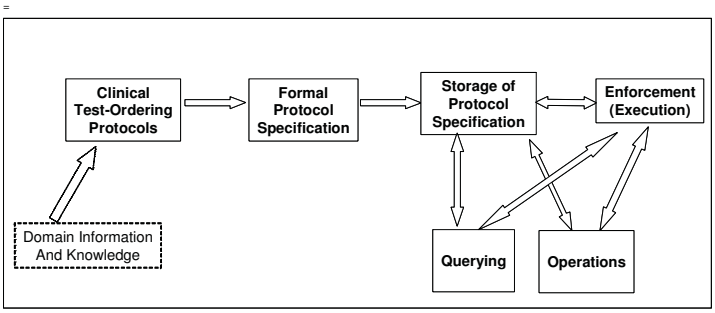


Figure 11 The process of supporting the management of clinical protocols

As illustrated in Figure 11, to comprehensively support the management of computerised clinical protocols, several aspects need to be incorporated and coordinated as components of the management process. Domain knowledge that exists in the form of expertise and literature on recent advances and discoveries in medical knowledge is the source of clinical protocols. The translation of this domain knowledge into clinical protocols is done by clinicians and is outside the scope of this book. Formal representation of protocols and creation of formal specifications and their subsequent storage is an important aspect of the computerisation of CGPs. The instantiation and execution/enforcement of computerised CGPs with respect to specific individual patient cases is a vital component of the

management of computerised CGPs. The manipulation of both the formal specifications and the enforcement process consists of the two aspects: querying; and performing pre-defined manipulation operations on them. The process illustrated in Figure 5.3 covers all aspects that ensure that the interaction and information related to the clinical protocol are manageable.

The SpEM framework for protocol management, presented in Section 3.3 of Chapter 3, and further explained in Section 5.3, is made up of the specification, execution, and manipulation planes. Figure 12 enhances Figure 11 by illustrating how the clinical management process fall into the three planes of the SpEM framework. The management process is fitted into the framework as follows:

- *Specification plane:* protocols are translated into formal specifications which are stored in a suitable form;
- *Enforcement plane:* the stored specifications are used to create patient plans that are executable by a computer-based execution mechanism; and
- *Manipulation plane:* the stored specifications and the executing patient plans are manipulated using pre-defined operations and queried.

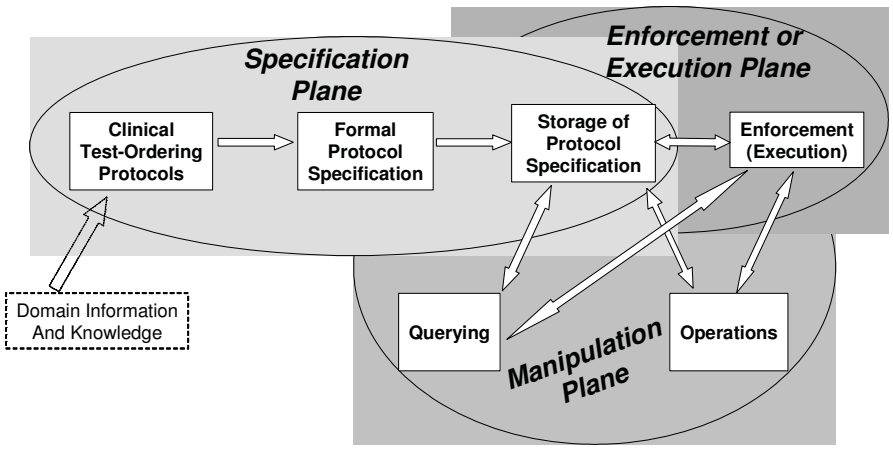


Figure 12 The clinical protocol management support process in the context of the SpEM framework

It is interesting to note that storage is at the intersection and, hence, plays a central role in the SpEM framework. This suggests the crucial role in which database technology can play in supporting protocol management. An interesting question is: To what extent can a database system support every process in each of the three planes? The answer to this question is presented next:

Supporting the specification plane: Formal protocol specifications are stored in the database. However, the process of translating guidelines to formal protocol specifications may not be directly supported by using database technology.

Supporting the execution plane: Important tasks in the execution plane are: the execution of protocols and the storage of information resulting from the execution. Execution can be supported by database technologies such as triggers, stored procedures and integrity constraints. Storage is the core function of a database system. Therefore, the whole of the execution plane can be supported through the use of database technology.

Supporting the manipulation plane: The manipulation plane involves queries and operations, which are performed on the information and knowledge that form part of managing clinical protocols. The database systems provide querying and operations on the data that they hold. Therefore, the manipulation plane can be fully supported by the use of database technology.

Figure 13 is an enhancement of Figure 12 by adding information on the enabling technologies to illustrate the method of supporting the management of clinical protocols.

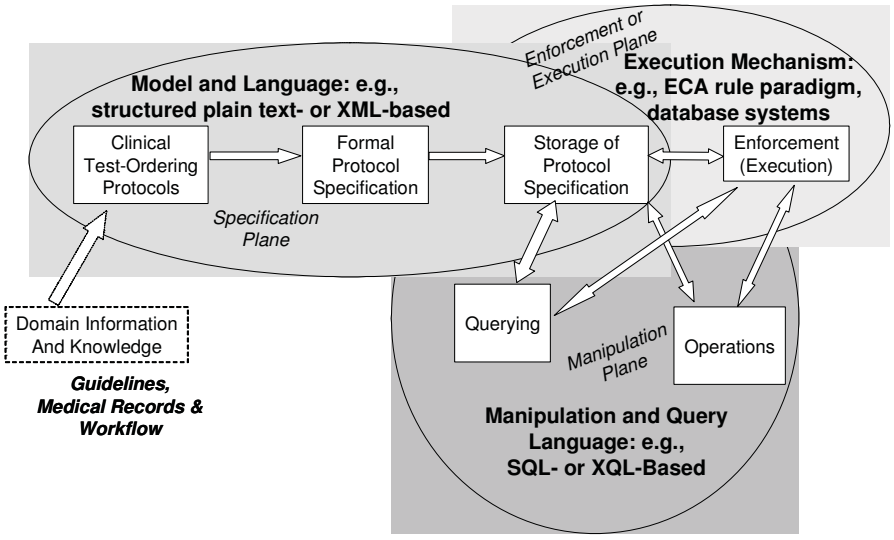


Figure 13 The enabling technologies for supporting protocol management

The method for supporting the management of protocols involves the provision, to the process within the framework, of following enabling technologies:

- Model and language for supporting the Specification Plane;

- Execution mechanism for supporting the Enforcement Plane; and
- Manipulation and query language for supporting the Manipulation Plane.

A declarative language, the **P**rotocol **LAN**guage, PLAN, together with its model, were developed. PLAN uses the event-condition-action (ECA) paradigm as the core representation construct for specifying clinical protocols. The storage of the ECA rule-based protocol specifications is achieved by the use of the relational database model. For each patient, the relevant protocol is customised and installed as an instance within the ECA rule mechanism of a database system. The execution of the patient plan proceeds according to the ECA rule mechanism which monitors events in the local patient record and the time points of interest to the protocol. Thus, the same database where protocol specifications and the patient record are held can also be used as the execution mechanism for the protocols. Provision is made to perform operations and to issue queries against the protocol and instance specifications and the instance's execution process and state. A suitable query and manipulation language is used for this purpose.

5.5. Summary

This Chapter has presented a description of the general problem of supporting the management of clinical protocols. At the core of this general problem, clinical guidelines need to be formally specified to create generic specifications, which later require customisation in order to be applied to a specific clinical problem scenario. This chapter has also reviewed the SpEM framework, which was presented earlier on in chapter 3, for supporting the management of clinical protocols. An important feature of this framework is the inclusion of a plane for the manipulation of information and knowledge as one of the core and essential aspects in addition to the usual specification and execution planes. This chapter has also presented the MonCooS approach for supporting the management of clinical protocols by using an active database-based approach that the places more emphasis on monitoring and coordination than on reasoning. The protocol management support method is centred on the knowledge and information database. This database is where specifications are held. The execution mechanism relies for its initial enactment, its progress and the information it generates on this database. It is also against this database that the manipulation of protocols and their executing instances through queries and operations is applied. Central to the SpEM framework, and the MonCooS approach, is the use of the ECA rule paradigm

for supporting the management of clinical protocols. The next three chapters will discuss in detail the MonCooS approach from the perspective of the three identified management planes: specification, execution and manipulation.

Chapter 6 Supporting the Specification of Clinical Protocols

6.1. Introduction

Clinical guidelines and protocols exist as natural language documents promulgating results of medical research or clinical trials. They may also exist as human expertise or as an unwritten part of organisational custom and culture. To support the effective management of knowledge in clinical guidelines and protocols there is a need to support the creation of computer-based specifications of clinical guidelines or protocols. These specifications should be *generic* so that they can be applied to different patients or to different organisations. The specification must be *formal* so that computational techniques could be used in supporting the management of these specifications. To support the specification of clinical protocols, the **P**rotocol specification **LAN**guage, **PLAN** (Wu, B. 1998), was developed. The aim of this chapter is to present the protocol representation model, a description of the language, PLAN, and the methodology for modelling protocols, which were developed for supporting the specification of clinical protocols by using the event-condition-action (ECA) rule paradigm.

This chapter is organised as follows: a brief background to PLAN is presented in Section 6.2; some definitions of terms and concepts as they are used in PLAN are presented in Section 6.3; the protocol specification model is presented in Section 6.4; the syntax of protocol specification language, PLAN, is presented in Section 6.5; the methodology for modelling protocols with domain expert involvement and specifying the resulting protocols in PLAN is described in Section 6.6; a discussion of issues in this chapter and related work is presented in Section 6.7; and, finally, Section 6.5 presents a summary of this Chapter.

6.2. Background to the Specification Language, PLAN

PLAN, the **P**rotocol specification **LAN**guage, was initially proposed by Wu (1998). PLAN is a generic and declarative language that uses the ECA rule paradigm to specify domain knowledge, which needs to be enforced by a computerised mechanism. In this book, PLAN is used for defining or specifying clinical protocols. In his original proposal, Wu (1998) stated

the aims of the design of the protocol specification language, PLAN, as being to allow the language to be:

- Easily usable by domain experts such as doctors and nurses in daily practice;
- Rich enough to specify a wide range of domain situations and tasks in the form of ECA rules;
- Flexible enough to describe different domains;
- Able to be implemented easily by using an Active Mechanism;
- Easy to integrate with systems and data that are in routine use within the application domain, in this case, healthcare; and
- Generic enough to be used in other domains with similar requirements (Wu, B. 1998).

The following sections present PLAN and the concepts and model behind PLAN as enhanced and refined in this work (Dube 2000b, 2000a; Wu, B. and Dube 2001).

6.3. Definitions of Terms and Concepts in PLAN

A **patient category** is a problem, disease or symptom-based classification of patients. Patient categories are created for the purpose of grouping patients with the same clinical problem. A single clinical protocol (defined next) is defined for each patient category. A clinical protocol contains domain knowledge that is applicable to the solution of the problem that forms the basis of a patient category. This research aims at providing automated assistance or support to the task of applying or complying with clinical protocols to a specified patient category. A patient is placed into a given category by the clinician who decides whether or not the patient satisfies the criteria for entry into that category.

A **clinical protocol** is a generic specification of a programme of clinical tasks/interventions to be applied to patients in a given patient category based on locally agreed or consensus clinical guidelines. As conceptualised in PLAN and its model, the clinical protocol is used as a template that is to be customised in order to construct a *patient plan* (defined next) for a particular patient in a patient category. A clinical protocol contains two main components: a set of criteria-based schedules to cover all the variations in the condition of patients in the patient category, and a set of protocol rules.

A **patient plan** is a description of performing a set of situation- or time-dependant actions for the care of an individual patient. A patient plan is derived from a specification of a protocol associated with a given patient category. The derivation of a patient plan from a

protocol involves customising the protocol using patient-specific attributes. Every patient plan is associated with an individual patient for a particular time duration. During its life time, a patient plan can be in any one of the states:

- Ready: when a plan has been created from the protocol specifications by customising and linking it to an individual patient;
- Active: when one or more rules in the plan can be fired and executed
- Suspended: when all rules in a plan are temporarily deactivated; and
- Terminated: when the plan expiry period has passed or then the plan has been stopped by a user.

A **static Rule** is a rule that performs a clinical task, activity or action subject to time being at a specified absolute value or within a specified time interval. A Static Rule can be regarded as an event-condition-action (ECA) rule with a condition that always yields a value of 'True'. The term *static rule* for describing a rule refers to the idea that the firing time of the rule is predetermined and definite on creation of the rule. Further to this, the rule is not associated with any logical event except a time event. In the specification of a Clinical Protocol, a Static Rule exists as either a Protocol Rule or a Schedule Rule.

A **dynamic rule** is an ECA rule that performs a clinical task, activity or action for the care of a particular patient, in reaction to some condition being satisfied after some event has occurred. In the specification of a clinical protocol, a Dynamic Rule exists as either a Protocol Rule or a Schedule Rule. A Dynamic Rule in a Patient Plan is an instance of a Protocol Rule, a Schedule Rule or a Global Rule, which is contained in a Protocol associated with a Patient Category to which the Patient belongs. The term *dynamic rule* refers to the fact whether or not the rule will fire and/or execute is determined dynamically depending on the situation at any point during the execution process.

The **state of a rule** indicates if the corresponding rule is applicable at any moment during the lifetime of the containing protocol or schedule. There are basically three types of rule states. In the *active state,* the rule is applicable now. An active rule can be in either the executing or the waiting sub-state. In the *inactive state,:* the rule is not applicable now. Sub-states include pending, stopped, finished or deleted.

Schedule: a Schedule is a set of static and dynamic rules that apply to a specific clinical variation in patient condition within a given Clinical Category. Schedules form part of the specification of a Protocol.

A **protocol rule** is a static or dynamic rule in a protocol specification that is independent of any schedule in the protocol. The scope of a protocol rule is entire protocol or a single patient category. The protocol rule dynamically monitors an event of interest and performs actions (order tests, page medic, modify schedule, etc.) based on certain inputs e.g. test results and vital signs data.

A **schedule rule** is a static or dynamic rule that is similar to a Protocol Rule only that its scope is the Schedule that is contained in a Protocol.

A **global Rule** is a static or dynamic rule whose scope includes all patient categories or all protocols defined in the system. Global rules are defined to apply irrespective of the protocol being followed for the patient. In other words, a global rule applies to all protocols and monitors every patient in the system. Thus global rules organisation-specific.

6.4. The Protocol Specification Model

The model of protocol specifications for use with the specification language, PLAN, is illustrated by means of the UML class diagram, in Figure 14 The figure shows the entities and relationships between the protocol representation constructs and the problem domain-specific entities. At a generic level, the model of protocol specifications consists of representation primitives, structure constructs, patient model and operational state representation.

Representation primitives form the basic building blocks in the protocol specification model. Structure constructs are high-level compound entities that are built by combining the representation primitives together. The patient model, while it is not explicitly part of the specification language, PLAN, forms the application domain basis and provides the vocabulary for the other components of the model. Operational state models the dynamic aspects of the support system such as the states of execution objects and processes, and domain objects, such as patients.

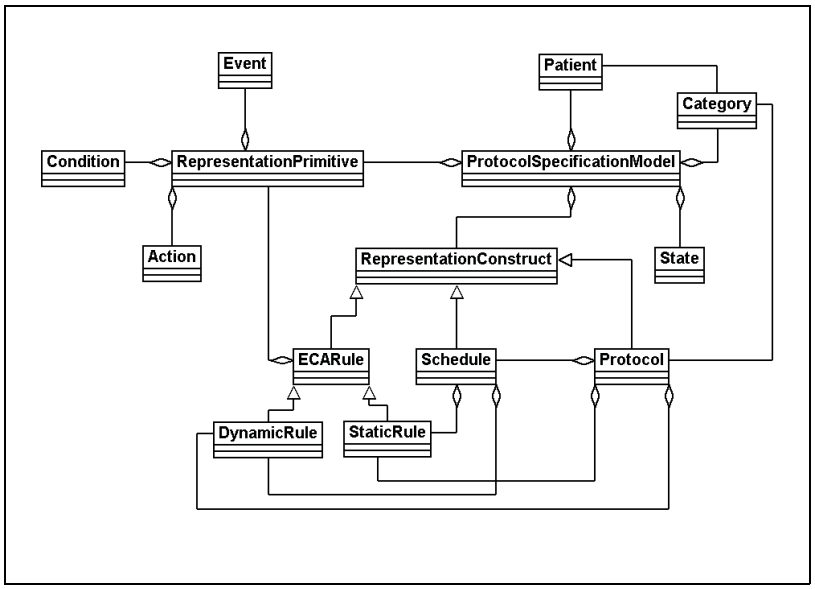


Figure 14 The detailed model of a protocol specification in terms of the UML class diagram

A protocol is associated with one and only one patient category. A patient category may contain many patients. Patients in the same category will be subject to the same protocol. For simplicity, a patient may belong to one category. However, in the real world, a patient experiencing co-morbidities may belong to more than one category and become subject to more than one protocol since a patient may suffer from more than one disease. For example, a patient with diabetes may also have renal and vascular diseases, and may be required to be assigned to the corresponding disease categories. The case for co-morbidities is left to future work.

Each patient will have a patient plan based on the general protocol for a given category. Such a patient plan will take into consideration the patient-specific circumstances. A protocol may be associated with many patient plans. A patient plan is associated with only one protocol from which it is derived. For simplicity, each patient will have only one patient plan at any time and each patient plan must be associated to one Patient. In the real world, it is necessary to allow a patient to have more than one patient plan each derived from the protocol associated with each of the categories to which the patient is assigned. The patient plan can be viewed as a customised version of the protocol.

A clinical protocol's logic is contained in one or more protocol rules (static and dynamic rules) as well as one or more schedules. A protocol can contain many schedules, each of which may not be mandatory for every patient. The logic of a schedule is contained in

one or more schedule rules (static and dynamic rules). Protocol and schedule rules differ only in scope. The scope of a schedule rule are the patients for whom the schedule applies. The scope of protocol rules is simply the entire set of patients in a category. In other words, protocol rules may also be viewed as category rules and are mandatory for all patients who are subject to the protocol. A protocol rule may not be shared by many protocols. Global rules, which may be static or dynamic rules, do not belong to a protocol but stand alone as rules that apply to all patients across all categories or the entire health care unit or organisation.

6.4.1. Protocol Representation Primitives in PLAN

This section describes the protocol representation primitives, which are the basic lowest-level building blocks for protocol specifications. Figure 15 illustrates the core representation primitives for PLAN.

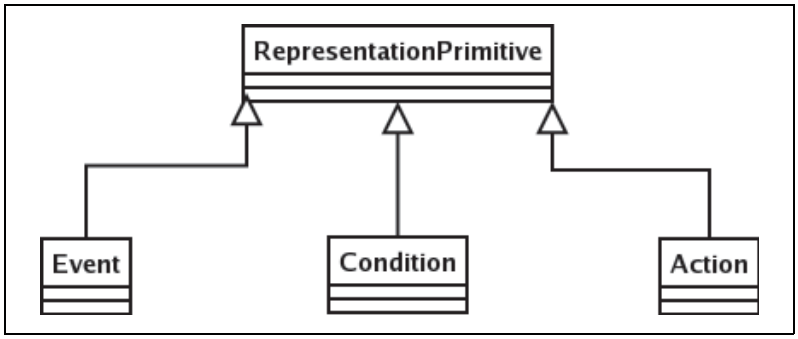


Figure 15 The core representation primitive constructs in PLAN

The Event

An *event* is an occurrence of interest in a given domain and requires some reaction, which may be manual or automated. From a theoretical perspective, a *primitive* event is instantaneous, atomic and bound to a specific point in time. This makes event detection easier as it eliminates contentious issues such as when to signal the occurrence of a day-long event. However, in real life, events can be long running activities consisting of one or more processes. This book focuses on events within the clinical domain and borrows ideas and concepts on clinical events found in literature, especially the work of Hripcsak et al (1996). A broad range of occurrences are covered by the generic term *clinical event.* Hripcsak et al (1996) researched into the design of a clinical event monitor and identified the following examples of simple clinical events: registration and administrative events patient visit,

admission, discharge and transfer; laboratory test-related events such as ordering and receiving results of tests; distribution of medication by the pharmacy in response to prescriptions; and scheduling of major procedures. Anything that can happen to a patient can be considered a clinical event. Event monitoring is an important task in the care of patients. Hripcsak et al (1996) also identified the benefits of automated monitoring of clinical events to include:

- The interpretation of laboratory results; warning clinicians about medication in cases of allergies;
- The detection of drug-to-drug interactions and side-effects;
- Automated prompts for suggesting a diagnosis or a new treatment option; and
- The co-ordination of complex tasks that are part of a clinical guideline.

Besides the simple events mentioned above, there are also other types of events. *Temporal Events* are a type of events that refers to occurrences of instances in time. The following are subtypes of the type, temporal events: absolute time events; relative time events: these are time points that occur relative to some reference time point (the zero time); and periodic time events. *Abstract Events* are conceptual events. This book will leave the support for temporal and composite events in clinical guidelines to future work. While the significance of these type of events in guideline systems is recognised, the prototype system developed in this book will initially support only simple clinical events and focus more on supporting the overall management framework.

The Condition

Clinical events trigger the logical criteria evaluation that leads to a determination of whether or not an appropriate clinical intervention is warranted. Examples of conditions in the clinical domain are: the presence of a disease, a result that exceeds a threshold and an age limits. Clinical criteria may be difficult to express as computable conditions. Such criteria may require eliciting the experts (clinicians) in order to evaluate them. Clinical protocols may be expressed as sets of criteria and actions. For purposes of this study, support is provided only for simple conditions that can be specified as logical expressions that are meaningful to the application domain. Within the ECA rule paradigm context, a *condition* is a Boolean expression that is evaluated when an event of interest occurs. A simple condition involves the comparison of a single attribute with an absolute value while a compound condition consists of conditions combined with the AND-OR connectives

The Action

An action is a set of operations meaningful to the application domain. The action of a rule may be to give suggestions, e.g., relating to clinical laboratory investigations and prescriptions; send messages of any of the types: alert, interpretation, maintenance, screen and patient state information; communicate with other systems such as workflow and patient record systems; and perform operations on other rules such as causing another rule to execute, scheduling the firing of other rules, terminate other rules, and adding or deleting another rule. Ideally, a single rule may perform several actions.

6.4.2. Representation Constructs in PLAN

Representation constructs in PLAN are illustrated in Figure 16. These constructs are named entities that are composed of the representation primitives. In this PLAN, the protocol representation constructs are the ECA rule, the schedule and the protocol. The next paragraphs describe these constructs

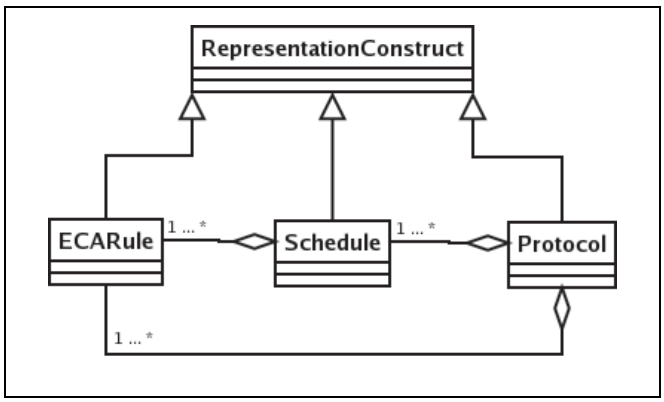


Figure 16 The structure of the representation construct in PLAN

The Rule

The rule is a protocol modelling construct that combines the three basic primitives, event, condition and action, into a single entity. A rule in the specification model is expressed in the form of the Event-Condition-Action (ECA) paradigm (Dittrich, Gatziu et al. 1995), with the semantics that the *action* specified in the rule will be performed when the rule is triggered by some *events* and the rule's *condition* is satisfied. In clinical protocol modelling, the only approaches that use the ECA paradigm as a modelling construct are the Arden Syntax for

Medical Logic Modules (Hripscak, Luderman et al. 1994) and HyperCare (Caironi, Portoni et al. 1997).

The Schedule

The Schedule is a protocol modelling construct that combines static and dynamic rules into a single module. However the schedule in a protocol is different from the schedule in a patient plan in that the schedule in a patient plan only holds static rules whilst in a protocol it holds both static and dynamic rules. This is so because when a plan is created, all rules are placed into one of two sets: the set of static rules, which becomes the plan's schedule, and the set of dynamic rules.

The Protocol

The protocol is the highest level construct in the protocol representation model for PLAN. It combines the set of protocol rules and the set of schedules into a single module, the protocol itself.

6.4.3. The Patient Record, Patient States and Execution States in PLAN

Patient record: In the model used in this Study, the patient record plays an important role. The ECA rules that make up the protocol are designed to monitor the patient record for events of interest. Our model assumes that the patient record and the ECA rule paradigm are based on the relational data model. It is the changes that occur within the patient record that determine whether or not rules in a patient plan will execute.

Operational state: Figure 17 illustrates the UML class model of the operational state for a clinical protocol.

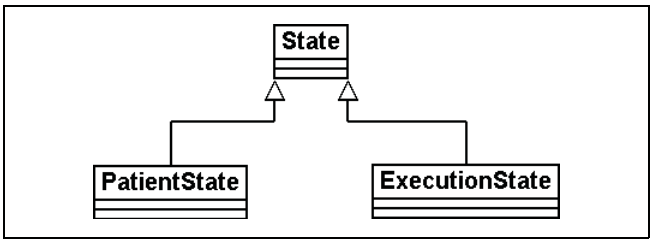


Figure 17 The UML class model of the operational state

The state of a patient within a given protocol is modelled during the specification of the protocol. The UML state chart is the tool used to model the patient states that are relevant to a protocol. The protocol rules are derived from the UML state chart. Section 6.6 presents the method developed for modelling protocols using the UML state chart.

The states of the execution of a protocol instance are regarded, as an important property of the execution mechanism. In general, patient states and execution states are closely related concepts to the extent that most systems model only one or the other but not both (Wang, Peleg et al. 2002). This study takes the approach that patient states are important in modelling the protocol knowledge while execution states are important in the protocol execution phase.

6.4.4. Discussion

This Section has presented the protocol specification model and its basic concepts and terms. The information representation primitives in PLAN are the event, condition and action. These primitives were described from the clinical domain perspective. The section also introduced the higher-level protocol representation constructs in PLAN, which are the ECA rule, the schedule and the protocol. The later two are essentially collections of ECA rules. The important role in PLAN of the patient record was highlighted. The patient record plays a central role in the execution of a PLAN-based protocol since the changes in the patient record drive the execution ECA rules. This study takes the approach in which the patient record and the ECA rule mechanism are combined within a database management system environment. Patient states may play a important role during the protocol knowledge modelling and specification phase. Patient states may also form the basis of the execution of a protocol. Execution states are more relevant during the execution phase of protocol instances. This section has described the protocol specification model on which PLAN is based by using the UML class model to illustrate the relationship among the protocol representation constructs and between these construct and relevant problem domain entities such as patient categories and patients.

6.5. The Protocol Specification Language, PLAN

This section describes the protocol specification language, PLAN, and presents an example of a clinical protocol specification expressed in natural language and in PLAN.

6.5.1. Description of PLAN

In PLAN, a protocol specification has the BNF syntax illustrated in Figure 18. (See Appendix A for a full listing of the Backus – Naur Forum (BNF) syntax of plan)

```
<protocol> ::= PROTOCOL <protocol_header>; <protocol_body>; END PROTOCOL.
<protocol_header> ::= <protocol_name>,<description>,<creator>,<category>;
<protocol_body> ::= <schedule_set> ; <protocol_rule_set>;
<schedule_set> ::=SCHEDULE_SET <schedule_list> END SCHEDULE_SET
<protocol_rule_set> ::= RULE_SET <protocol_rule_list> END RULE_SET
```

Figure 18 The PLAN syntax of a protocol

A protocol consists of a header followed by a body started and terminated by the words PROTOCOL and END PROTOCOL respectively. The protocol header consists of the name and description of the protocol and associates the protocol with its creator and the patient category.

```
PROTOCOL microalbuminuria;
   DESCRIPTION: protocol for micro-albuminuria patients;
   CREATOR: Dr John Doe;
   CATEGORY: MA1;
<set of schedules>;
<set of protocol rules>
END PROTOCOL.
```

Figure 19 Structure of a protocol specification in PLAN

For instance, Figure 19 illustrates an hypothetical example for the specification of a protocol header. The name of the protocol is *microalbuminuria*. This protocol was created by *Dr John Doe* for patient category *MA1.* The body of the protocol consists of the set of schedules and the set of protocol rules.

The Schedule: In PLAN, a schedule specification has the BNF syntax illustrated in Figure 20.

```
<schedule> ::= SCHEDULE <schedule_header>;<schedule_body> END SCHEDULE
<schedule_header> ::= <schedule_name>,<schedule_description>
<schedule_body>::= <entry_criteria>; <schedule_rule_set>;
<schedule_rule_set> ::= SCHEDULE_SET <schedule_list> END SCHEDULE-SET
<schedule_list> ::= <schedule> | <schedule>;<schedule_list>
<schedule> ::= <schedule_rule>| <schedule_rule>; <schedule_rule_list>
<schedule_rule> ::= <static_rule>|<dynamic_rule>
```

Figure 20 The PLAN syntax of a schedule

The Schedule specification is started by the word SCHEDULE, followed by the schedule header and body, and terminated by the words END SCHEDULE. The schedule header consists of the schedule's name and description while the schedule's body is a list of static and dynamic rules. For example, Figure 21 illustrates the structure of the specification of a schedule in PLAN.

```
SCHEDULE  microalbuminuria_sch,
   DESCRIPTION: micro-albuminuria schedule for patients with confirmed
         diabetes;
   ENTRY_CRITERIA,
      CONDITION: confirmed_diagnosis = DIABETES,
      DESCRIPTION: pre-condition for entry to the micro-albuminuria schedule;
  <list of schedule rules>
END SCHEDULE
```

Figure 21 Structure of the specification of a schedule in PLAN

In Figure 21, a schedule named *microalbuminuria_sch* that is applicable for patients who are confirmed diabetics is specified. The body of the schedule is a list of schedule rules, which are represented in Figure 21 by the place holder <list of schedule rules>.

The static rule: Figure 22 illustrates the BNF syntax of the specification of a static rule.

```
<static-rule> ::= <rule_header>,[<description>,]<time_events_spec>,<action_spec>
<rule_header> ::= STATIC_RULE  <rule_name>
<time_events_spec> ::= <zero_point>,<start_point>,<end_point>,<time_event>
<zero_point> ::= FROM: <identifier> | <domain_term>
<start_point>::= STARTING: <time_length>
<end_point>::= ENDING: <time_length>
<time_event> ::= ON: "{"<time_event_list>"}" <time_unit>  | ON_EVERY: <time_length>
 <time_event_list> ::= <integer> | <integer>, <time_event_list>
<time_length> ::=  <integer><time_unit>
<time_unit> ::= MINUTES | HOURS | DAYS | WEEKS | MONTHS | YEARS
```

Figure 22 The PLAN syntax of a static rule

The static rule's specification consists of a rule header, followed by an optional description, then time event specification and, finally, the specification of the rule's action. The header is made up of the label, STATIC_RULE, followed by the name of the rule. Time event specification consists of a zero time point, <*zero_point*>, a start time point <*start_point*>, an end time point, <*end_point*>, and a frequency interval, <*interval*>. Figure 23 illustrates the structure of a static rule in PLAN.

```
STATIC_RULE  ma1sr1,
DESCRIPTION: rule orders test during the period of the diagnosis of
microalbuminuria,
FROM: start_of_protocol,
STARTING: 1 WEEK,
```

```
ENDING: 3 MONTHS,
ON_EVERY: 1 MONTH,
DO: order_test ( 'A' );
```

Figure 23 An example static rule in PLAN

In Figure 23, the name of the static rule is *ma1sr1*. The static rule *ma1sr1* orders test 'A' every month for the three months from one week after the protocol is instantiated. Using a static rule, actions may be scheduled for one-time execution, or for repeated execution at regular intervals.

Table 6.1 Example specifications of the static rules in PLAN

DOMAIN EXAMPLE	FORMAL DEFINITION	GENERIC SPECIFICATION	PLAN SPECIFICATION
Order a test or perform appropriate action at a given absolute time point	sr1 = <d, a> where d - absolute date, a – action	**ON** 30-Jul-04 **DO** order anti-DCV	STATIC_RULE sr1 DECRIPTION: once-off order of a test FROM: start_of_protocol STARTING: 30-July-04 ENDING: 30-July-04 ON: 30-July-04 DO: order_test("anti-DCV")
Order a test or perform appropriate action at regular time intervals from a one specified time point to another	sr2 = <T_0, T, a>. where T = (T_1, T_2), T_1 is time length of the same unit as that of T_0 , T_2 is the either a time length or an absolute time point or a domain-dependent conceptual time point.	**FROM** date-of-conception **ON EVERY** 3 months **UNTIL** 9 months **DO** order "blood-test"	STATIC_RULE sr2 DECRIPTION: once-off order of a test FROM: date-of-conceptin STARTING: 1 week ENDING: 9 months ON: 4 weeks DO: order_test("blood-test")
Order a test or perform appropriate action at each point in a specified sequence of time points all measured from a specified time point	sr3 = <T_0, T, a> where T = (t_1, t_2, … t_n) and t_i are time lengths all of one arbitrary time granularity.	**FROM** date-of-admission **ON** {2, 3, 5, 8} days **DO** order {U, K}	STATIC_RULE sr3, DECRIPTION: repeated order of a test on irregular time points; FROM: date-of-admission ON: {2, 3, 5, 8} days DO: order_test("U,K")
Order a test or perform appropriate actions within a given interval optionally from a specified time point	sr4 = <T_0, T, a> where T = [T_1, T_2] is the time interval during which the action *a* is to be carried out.	**FROM** onset-of-pain ON PERIOD 4 TO 6 days **DO** order K	STATIC_RULE sr4a DECRIPTION: once-off order of a test within a given time interval FROM: start_of_protocol; STARTING: 4 days; ENDING: 6 days; ON: any day; DO: order_test("K");
		ON PERIOD 25-Jan-01 **TO** 30-Jan-01 **DO** order K	STATIC_RULE sr4b DECRIPTION: once-off order of a test within a given time interval FROM: start_of_protocol STARTING: 25-Jan-04 ENDING: 30-Jan-04 ON: any day DO: order_test("K")

In PLAN language, static rules are used to define schedules of clinical actions to be performed at certain points or periods in time depending on clinical requirements. It can be noted that static rules can be used to express generic scenarios in which actions need to be performed at specified time points relative to a starting time, which may be a conceptual time point.

Table 6.1 presents the example scenarios that need to be expressible using static rules in PLAN. From these example scenarios, it can be noted that static rules monitor time events and perform a specified action, such as prompting for, or issuing, a test order, on detecting the occurrence of a time event of interest. The time events may be specified to be a single absolute time point, regular or irregular time lengths measured from a single time reference point up to a specified time point, or a specified time interval or period measured from a specified time reference point or expressed as an absolute time interval.

The dynamic rule: As illustrated in Figure 24, a dynamic rule has two main parts : the Rule-Header and the Rule-Body. Rule-Header consists of the Rule-Name and the rule description. Rule-Body consists of the ECA component parts of the rule : the *event specification* defines the event which triggers the rule; the *condition specification* defines a logical expression about either patient's states or timing events; and the action specification defines the action or tasks to be performed when necessary. A possible operation from the domain of clinical laboratory test ordering protocols can be to suggest the order of a specified test.

```
<dynamic-rule> ::= <rule-header><rule_body>
<rule-header> ::= RULE <rule-name>,[<description>,]
<rule_body>::= ON: <event_spec>, IF: <condition_spec>, DO: <action_spec>;
<event_spec> ::= <event_name> ( [<parameter_list>] )
<condition_spec> ::= <condition> | <condition> {AND | OR} <condition_spec>
<action_spec> ::= <action> | <action>, <action_list>
<condition> ::= logical condition
<action> ::= <action_name> ( [<parameter_list>] )
```

Figure 24 PLAN syntax of a dynamic rule

Dynamic rule specifications exist in a protocol specification as a protocol and schedule rules whose scopes are the protocol and schedule respectively. In a patient plan, the dynamic rule exists simply as the plan's dynamic rule with no distinction regarding whether it belongs to a protocol or a schedule.

Figure 25 illustrates an example PLAN specification of a dynamic rule named *ma1sdr1*, which monitors the arrival of a clinical laboratory test result for a test named A and suggests the order of a further test named B if the incoming result is above 8.5.

```
RULE ma1sdr1,
        DESCRIPTION: rule to order test B if A result is abnormal,
        ON: result_arrival('A'),
        IF: A > 8.5,
        DO: order_test ( 'B' );
```

Figure 25 An example of a specification of a dynamic rule in PLAN

The Patient Plan: The protocol specification acts as a template that is used to create the patient plan. A *patient plan* is derived from tailoring a protocol to a specific patient in a particular category and is active for a finite time period. A patient plan is an instance of a test protocol that is relevant for a particular patient during a given time duration. A patient plan has the same syntax as a protocol except that it has the patient identification and/or the protocol from which it is derived. Figure 26 illustrates the syntax of a patient plan in PLAN.

```
<patient_plan> ::= PLAN <name>; <patient_detail>; <plan_body> END PLAN
<plan_body> ::= <static_rule_set>; <dynamic_rule_set>
```

Figure 26 The PLAN syntax of a patient plan

A Patient Plan has only one schedule composed from one or more protocol schedules whose selection is based on whether or not the schedules' entry criteria are satisfied by the patient. Protocol rules are instantiated to become dynamic rules within the patient plan. The two key components of a patient plan specification in PLAN are: a set of static rules followed by built from all the static rules selected from the relevant protocol, and a set of dynamic rules which is built from the protocol and schedule rule sets in the relevant protocol specification. Thus, the Patient Plan is essentially a set of rules which when triggered and executed, determine when clinical interventions that may be suggested with respect to an individual patient. The sequence of suggested actions may result from time alone as a stimulus in static rules. In addition the patient plan may employ dynamic rules which allow action suggestions to be sequenced or enabled in response to a combination of time and other asynchronous events which might occur during an episode of care.

ECA rules are the building blocks for higher-level constructs in PLAN: the schedule and the protocol. In the Arden Syntax (Hripscak, Luderman et al. 1994), the ECA rule is the highest level construct that stands alone as a module the medical logic module (MLM). It is interesting to note that it would be possible to define PLAN specifications using the Arden Syntax modules as the building blocks.

This section has presented a protocol specification language, PLAN. The design of the PLAN follows the ECA rule paradigm. This does not necessarily mean that the implementation of PLAN has use on an Active Database. However, it should be easier to implement the language if an Active Database is used. In this study, a prototype system

called, TOPS, that implements PLAN language by using the trigger mechanism of a modern database system was developed (see Part III of this Book).

6.5.2. An Example Protocol Specification in PLAN

Figure 27 presents the specification for the Protocol for Viral Hepatitis Testing (Protocol Steering Committee 1998) in PLAN. The structured natural language version of the viral hepatitis testing protocol, meant for clinicians, has already been presented in Section 2.4 of Chapter 2. Figure 27 serves to illustrate the use of PLAN in specifying a real life protocol for the purpose of providing clinicians with computerised assistance in applying the protocol to individual patients. The specification consists of three schedules each covering one of the suspected conditions among *acute viral hepatitis, hepatitis B carriers* and *previous or chronic hepatitis*. Each schedule consists of rules to suggest test orders appropriate for the suspected patient condition.

```
@PROTOCOL@ viral_hepatitis_testing;
DESCRIPTION: a protocol for ordering tests for patients suspected to
have the three conditions of acute hepatitis, heppatitis B carrier, and
previous/chronic hepatitis;
CREATOR: Dr John Doe;
CATEGORY: hepatitis_testing;

#SCHEDULE_SET#

^SCHEDULE^ acuteVH,
DESCRIPTION: a schedule for patients with a suspected condition of acute
viral hepatitis;

ENTRY_CRITERIA,
CONDITION: 'suspected_condition = acute_viral_hepatitis'
DESCRIPTION: this schedule is applicable to only those patients suspected to
have acute viral hepatitis;

STATIC_RULE avh1,
   DESCRIPTION: a rule to order the Anti-HAV on entry to this schedule,
   FROM: entry,
   STARTING: 0 minutes,
   ENDING: 5 minutes,
   ON EVERY: 4 minutes,
   DO: order ( 'Anti-HAV' );

RULE avh2,
DESCRIPTION: a rule to terminate execution if the anti_HAV result is
positive,
ON:  new_result('Anti_HAV') ,
IF:  result = 'positive',
DO:  stop();

RULE avh3,
DESCRIPTION: a rule to order the HBsAg on entry to this schedule,
ON:  new_result('Anti_HAV'),
IF:  result = 'negative',
DO:  order('HBsAg');

RULE avh4,
DESCRIPTION: a rule to order the HBsAg on entry to this schedule,
ON:  new_result('HBs Ag'),
IF:  result = 'posetive',
DO:  check_further_test_requests();

RULE avh5,
DESCRIPTION: a rule to order the HBsAg on entry to this schedule,
ON: new_result('HBsAg,),
IF: results = 'negative',
DO: oder('Anti-HCV');

^END SCHEDULE^

^SCHEDULE^ Hepatitis_B_Carrier
DESCRIPTION: a schedule for ordering test for patients who are suspected
Hetatitis B carriers;

ENTRY_CRITERIA,
CONDITION: suspected_condition = 'hepatitis_B_carries';

STATIC_RULE hbc1,
   DESCRIPTION: a rule to order the HBsAg on entry to this schedule,
   FROM: entry,
   STARTING: 0 minutes,
   ENDING: 5 minutes,
   ON EVERY: 4 minutes,
   DO: order('HBsAg');

^END SCHEDULE^

^SCHEDULE^ chronic_hepatitis,
DESCRIPTION: a schedule for odrering tests for patients with suspected chronic
hepatatitis;

ENTRY_CRITERIA,
CONDITION: suspected_condition='previous_hepatitis' OR
suspected_condition='chronic_hepatitis';

STATIC_RULE ch1,
   DESCRIPTION: a rule to order the Anti-HAV on entry to this schhedule,
   FROM: entry,
   STARTING: 0 minutes,
   ENDING: 5 minutes,
   ON EVERY: 4 minutes,
   DO: order('anti_HBc_total');

STATIC_RULE ch2,
   DESCRIPTION: a rule to order the Anti-HCV on entry to this schhedule,
   FROM: entry,
   STARTING: 0 minutes,
   ENDING: 5 minutes,
   ON EVERY: 4 minutes,
   DO: order('anti_HCV');

RULE ch3,
DESCRIPTION: a rule to order the two tests Anti-HBs and HBsAgs if the result
for Anti-HBc happens to be positive,
ON: new_result(Anti_HBc),
IF: result = 'positive',
DO: order('anti_HBs, HBsAgs');

^END SCHEDULE^

#END SCHEDULE_SET#

@END PROTOCOL@
```

Figure 27 The specification of the Viral Hepatitis testing protocol in PLAN

6.6. A Method for Protocol Modelling and Information Acquisition Using PLAN

This section presents a method for modelling clinical guidelines and protocol for the purpose of specifying them in PLAN.

6.6.1. Outline of the Method for Modelling a Protocol.

Protocol information is captured with help from local domain experts. The UML state chart is used as a tool for modelling the domain information. In the method presented in this section, the states of the patient in the context of a particular protocol are modelled with ECA rules defined as transitions between states. Once the clinical protocol is fully modelled and expressed in the UML state chart diagram, ECA rules can be extracted manually or automatically and then modularised in a hierarchical fashion using state and sub-state hierarchies in the state chart to create the protocol specification.

6.6.2. The UML State Chart as a Tool for Modelling ECA Rules

It has been shown that the state chart can be used as a tool for modelling ECA rules (Berndtsson, Mikael and Calestam 2001). The UML state chart models dynamic aspects of a single class and may need to be extended to allow a rule to be given a name as required in the active database manifesto (Dittrich, Gatziu et al. 1995). Further extensions may be required to allow modelling of composite events in the clinical protocol (Berndtsson, Mikael and Calestam 2001). Every transition in the UML state diagram corresponds to at least one ECA rule can be seen in Figure 28.

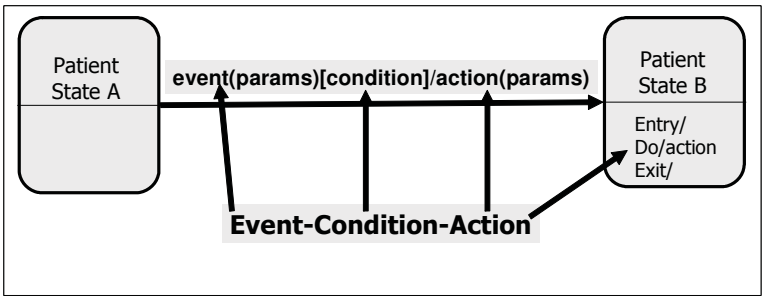


Figure 28 Capturing the ECA rules using the UML state chart transitions

6.6.3. Method for Creating the Protocol Specification

Figure 29 presents a summary of the steps for capturing domain knowledge for creating protocol specifications using the UML state chart as tool for modelling the knowledge in terms of the ECA rule paradigm. The method of modelling the clinical protocols involves the following steps:

Steps for the UML state chart-based method of modelling the clinical protocols

1. Interpretation and customisation of the text-based or flowchart guideline;
2. Identification of patient states in the context of the customised/localised protocol;
3. Specification of the events, conditions and actions or clinical interventions that make up transitions from one patient state to another;
4. Construction of a UML state diagram: the state chart diagram is constructed from the identified states and transitions;
5. Generation of protocol ECA rules from the UML state diagram; and
6. Specification of ECA rules into a suitable executable language.

Figure 29 Steps for creating ECA rule-based specifications of clinical protocols

Customisation: The domain experts, mainly clinicians, need to interpret their experience, current practice and the published text-based or flowchart guidelines in order to create a clinical protocol that is enhanced with local context.

Modelling: In the modelling approach adopted here, patient states are identified in the context of the localised protocol. The patient states are derived from the clinical logic of the guideline or protocol context and corresponds to the states in the state chart. The construction of a UML state chart that represents the protocol is based on these identified patient states. The specification of the events, conditions and actions that make up the protocol representation primitives are based on the transitions from one patient state to another in the state chart and the actions contained in each state.

Specification: The process of generating ECA rule specifications from the state chart can be manual or automatic. Every transition in the UML state diagram corresponds to at least one ECA rule. This process can be manual or automatic using UML-based tools such as Rational

Rose. Thus, a formal specification is created in PLAN based on the ECA rules that are generated from the UML state chart that is produced in the previous step. The schedules are created from rules generated from lower-level sub-states with the super-state representing the protocol.

Storage and Manipulation: A database of formal specification of protocol information is created. The ECA rule-based specifications are stored in the database in a manner that allows the ECA rules to be manipulated both individually and as collections making up the protocol specifications. In the prototype system presented in Chapter 9, the ECA rule -based specification of the clinical protocol is stored in a relational database, which permits the protocol knowledge to be manipulated, i.e., operated upon and queried, using the SQL.

This Section has presented a method for modelling clinical domain knowledge for the purpose of creating protocol specifications. UML state charts are created, with the help of a domain expert, from the clinical logic of the guideline or protocol. The rules that make up the protocol specification are obtained from the states and transitions of the UML state chart. In the literature, most clinical protocol modelling approaches model either the patient states or the execution states of the clinical protocol (Peleg, M. , Tu et al. 2002). In the approach presented in this Section, patient states are important for modelling domain knowledge for the purpose of creating protocol specifications. The execution states are considered to be the property of the execution mechanism.

6.7. Discussion and Related Work

PLAN is a specification language that is higher than database triggers and has the advantage of being independent from a specific product or trigger language. Specifications based on triggers are at a low level making such specifications more difficult to read and debug. Eder *et. al.* (1994) use a graphical description language to specify business processes or flow. They translate the resulting specifications into triggers of an active database. The same approach is taken here except that the language, PLAN, is not graphical. In Eder et al's work (1994), the whole description of a workflow process in a Workflow Description Language (WDL) is stored in rules and tables of database. Rules are automatically generated from the declarative specifications of the task and flows by the language compiler. The active DBMS is the workflow server and has the functionality described in the process specifications.

Each patient has his or her own rule set making the patient's plan. A similar idea is found in Appelrath et al's active repository (1995), which uses an active database for implementing the persistent and reactive parts of a process-centred software engineering environment. There, they identified the need for rule sets on a project basis, requiring extensions to their toolbox regarding multi-user and meta-programming capabilities (Appelrath, H-J, Behrends et al. 1995). The software engineering project is equivalent to the *patient* entity. However, their system could not support this phenomenon as it lacked multi-user support, i.e., the need for supporting several user groups each having its own set of rules.

Medical Logic Modules (MLMs) are ECA rules expressed in the Arden Syntax (Hripscak, Luderman et al. 1994). MLMs have been used to specify clinical protocols with no generic framework nor constructs at a higher-level than the ECA rules. As part of future work, it would be interesting to investigate the use of the Arden Syntax to specify PLAN rules.

6.8. Summary

This Chapter has presented a protocol specification model, which is based on the ECA rule paradigm. In this model, a clinical protocol is composed from modularised ECA rules, which are essentially templates that are used to create patient plans. The protocol specification language, PLAN, was described. PLAN is a simple and declarative language for specifying protocols as modules of ECA rules. Finally, this Chapter has presented the method for modelling protocol knowledge for the purpose of creating PLAN specifications. The modelling method first uses the UML state chart as a tool for capturing the domain knowledge, and then generates ECA rules from the state chart for use as the core protocol representation construct. The modelling method uses patient states and transitions between these states to determine what rules are relevant for inclusion in a protocol.

Chapter 7 Supporting the Execution of Clinical Protocols

7.1. Introduction

The main purpose of creating formal specifications of clinical protocols is to enable the execution of these protocols. The challenge is to provide an executable care plan for each patient that is appropriate for the management of the patient's clinical problem. It is also necessary to provide the clinician with protocol information at the moment when that information is most relevant, for example, at the point of care or the moment when new information on the condition of the patient becomes available. This chapter presents the conceptual approach and architecture for the enforcement of clinical protocols.

The rest of this chapter is organised as follows: Section 7.2 describes the approach to the execution of computer-based clinical protocols. The conceptual architecture for supporting protocol execution is presented in Section 7.3. The execution flow for supporting the execution of clinical protocols is presented in Section 7.4 while Section 7.5 presents the method for instantiating a clinical protocol for an individual patient thereby creating the patient plan. Section 7.6 presents the types of dynamic protocol management scenarios that need to be supported by the execution mechanism. Section 7.6 also describes the interaction between the real world and the protocol model. Section 7.7 reviews related work and, finally, Section 7.8 summarises this chapter.

7.2. The Approach to Protocol Execution

The execution of a clinical protocol involves the computer-based application of the protocol information to a specific clinical problem. The approach for the execution of event-condition-action (ECA) rule-based clinical protocols is illustrated in Figure 30 where it is presented in terms of the conceptual, logical and physical levels.

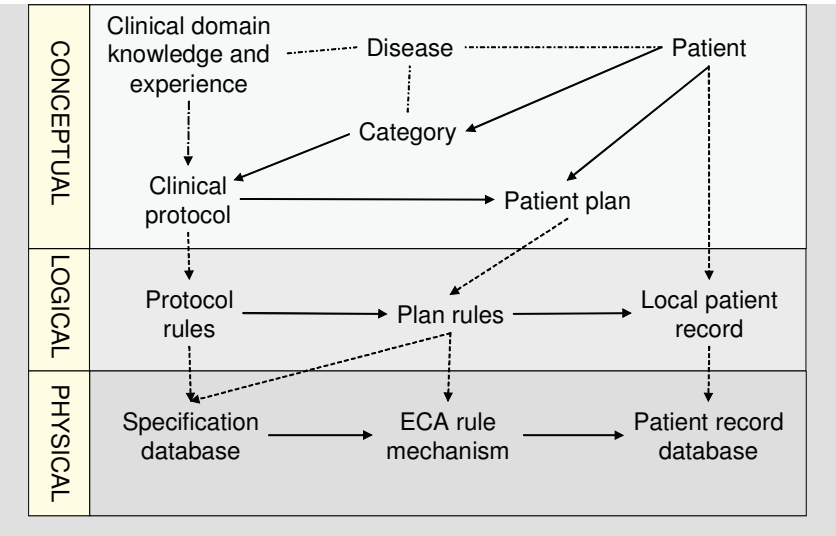


Figure 30 The approach to the executionof protocols

At the *conceptual level*, domain knowledge and information, which is usually in the form of natural language clinical guidelines, is expressed as a formal clinical protocol specification. In turn, the clinical protocol specification is mapped to a patient plan. This mapping customises the protocol specification to the needs of an individual patient. At the *logical level,* the protocol rules are mapped onto plan rules. Patient plan rules are derived from protocol rules during the protocol-to-plan mapping. At the *physical level*, the *ECA rules* in a patient plan are mapped onto a set of database triggers defined within the patient record database schema. These database triggers implement the execution mechanism for the patient plan. For the purpose of storage, the protocol and plan specifications are mapped onto a *specification database.* The manipulation operations are performed on the protocol specification database, the patient plans and patient data. Figure 31 illustrates the protocol enforcement approach, in the context of the SpEM management framework presented in Chapter 5, which consists of the three *planes*: specification, execution and management of the protocol specifications and their instances, the individual patient care plans. Within this framework and approach, a protocol specification is customised with patient-specific detail to create the patient plan. For instance, the protocol customisation process involves binding domain-dependent terms such as date-of-conception and patient-age in the protocol specification to actual values with respect to a specific patient. This creates a patient plan specification and occurs in the specification plane.

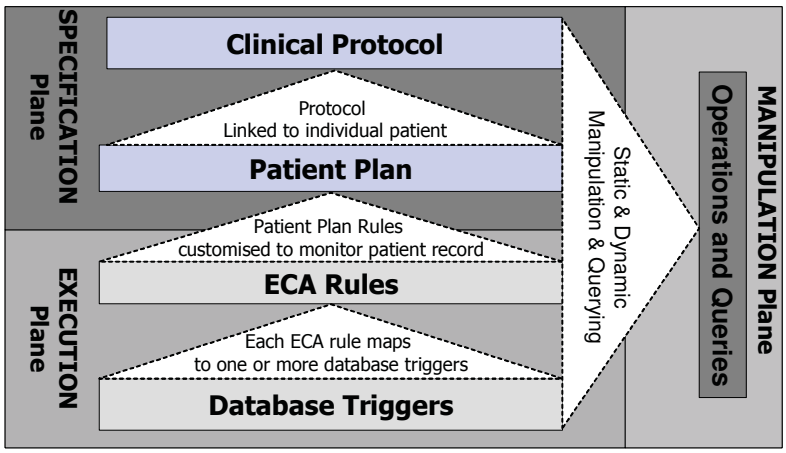


Figure 31 Framework and approach for the execution of clinical protocol

As part of the transition into the execution plane, the ECA rules in the patient plan are enhanced with hooks/references to patient data in the database. Each ECA rule in the patient plan is mapped to one or more database triggers to create a trigger set that implements the protocol logic for the specific individual patient. Patient plan execution proceeds according to the semantics of the ECA rule mechanism. Manipulation operations and queries can be applied dynamically to protocol specifications, the executing patient plan and to patient data.

7.3. Conceptual System Architecture for Supporting the Execution of Clinical Protocols

Figure 32 illustrates the conceptual system architecture for supporting the execution of clinical protocols. This architecture has been implemented in the prototype system to be presented in Chapter 9. In Figure 32, rectangular shapes denotes modules that are part of the architecture while rectangles with rounded corners denote external entities. The architecture is based on the wrapper principle. At the core of the architecture is a modern database management system (DBMS) with an ECA rule support mechanism, which is commonly referred to as the database trigger mechanism. Our prototype system, which is presented in Chapter 9, uses the Oracle DBMS as its core. This ECA rule mechanism of the DBMS serves as the clinical protocol execution engine.

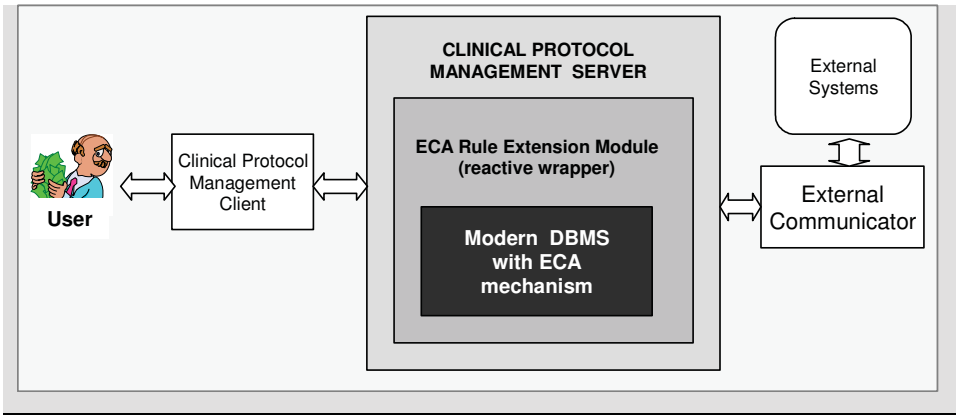


Figure 32 Conceptual system architecture for supporting the execution of clinical protocols by using active mechanism of a modern DBMS

An ECA rule extension module extends the basic ECA rule support within the DBMS. This ECA rule extension module is required to extend the ECA mechanism of the DBMS and provide features that are lacking within the database trigger mechanism. The clinical protocol management server provides the higher level functionality for delivering the protocol management operations. The protocol management client serves the purpose of providing the user with access to the management functionality. The user of system interacts with the clinical protocol management client, which interacts with the management server. The management client can be distributed and could be presented as a suitable user-friendly graphical interface. An external communicator module serves the purpose of linking the system to external systems such as the laboratory information system and patient record systems.

7.3.1. Advantages of the Conceptual Architecture

The use of a wrapper architecture that incorporates a DBMS that contains a basic ECA rule mechanism to provide support for the management of clinical protocols has a number of advantages. The capabilities of database system like safety, authorisations and, most importantly, recovery, are immediately available. The database is not only the blackboard for the execution process, but it is also the execution engine itself. The execution of protocols enjoys a high degree of concurrency because the architecture permits the increase in concurrency in a safe way. The usage of a standard modern database system also brings the benefits of a stable system that is available on different platforms. The tight integration of the

ECA-based protocol manager and the database could form a strong basis for easy integration with other health care applications such as care flow systems. Additional functionality, such as a distributed architecture, can easily be added later. Once a wrapper-based approach has been developed, it may be ported to other DBMS. Furthermore, the wrapper architecture allows for the ability to enhance and add active capability without the changing the client program.

From the clinical guideline and protocol support point of view, the conceptual a number of benefits. The target user can be the clinicians since no fully-featured programming language with complex structures need to be created. ECA rule-based protocol are meant to be written and used by clinicians with little or no training. ECA rules can provide explicit links to data, trigger events and messages to target users. The rules clearly define hooks to the clinical databases. The need for intelligent data–based monitoring of critical situations in patient care points to a number of further benefits that can be enjoyed from the ECA rule paradigm and database-based architecture. Round the clock physiological data collection can be attained through the use of the ECA rule-based mechanism. The difficulty in continuous monitoring and recording of generated data leading to mistakes that are not affordable in a critical environment, such as patient care, can be avoided by using ECA mechanism for automated continuous monitoring. The difficulty experienced by humans in keeping track of several parameters for a long time or in combining or synbooking many different parameter values for judgement or decision can be made easier by the database enhanced with the triggering ability of the ECA mechanism. Need for automated mechanisms that can handle repeated data collection and analysis for detection of alert situations can be easily met. The need for providing real time status alerts in order to save precious time as a way to assist domain experts (clinicians) in making decisions (treatment) can be addressed by the architecture. The need for alerts to occur, or for alert conditions to be checked, at the right moment, when the alert is relevant (e.g. when a doctor is proposing some medication - for adverse drug events alerts) is best achieved by this architecture.

7.3.2. Disadvantages of the Conceptual Architecture

An architecture that is based on the ECA rule paradigm within a modern DBMS as the core has a number of limitations. ECA rules in a modern DBMS have limited support through triggers. The main limitations of triggers include the lack of support for time triggers and temporal features, the inability to be applied to more than one table; no support for naming

and user-defined events; and lack of support for composite events within most existing DBMS trigger mechanisms. Furthermore, the action part of the ECA rule is implemented in DBMS as a stored procedure. Complex data definition in not allowed and only atomic values may be passed as parameters to stored procedures within the database. In addition to this, there is no direct access to other programs and external systems in the underlying DBMS and the operating system (OS). Another important limitation with ECA rules or active databases is the lack of development methods and tools, including the lack of transformation tools from higher level description formalisations to ECA rules. For instance, Petri Nets and state charts allow specification of event-action dependencies.

From the clinical guideline or protocol management support point of view, the ECA rule paradigm-based support for clinical protocol execution suffers several limitations. Guideline's overall logic is obscured by the detail of the individual ECA rule or decision module. ECA rules are suitable primarily for the task of monitoring and are very limited in their support for decision-making. They also do not model related decisions well leading to unexplained or complex interactions. Tu *et al* (2001) observed that chaining ECA rules as a method of modelling related decisions, sequencing of tasks and setting of goals breaks the desired modularity of Medical Logic Modules (MLMs), which are essentially ECA rules, and introduces maintenance problems of interdependent rules. The ECA rules, per se, as a mechanism for implementing protocols do not represent execution state or patient state, which are considered in the literature to be important primitives for guideline knowledge specification and execution (Pattison-Gordon, Cimino et al. 1996; Peleg, M, Boxwala et al. 2000).

7.4. The Execution Flow for Protocol Management

The aspects of the protocol management process that are of focus are the specification, customisation, installation and execution phases as well as the manipulation and querying that are applied to the four phases. Figure 33 illustrates the execution flow for supporting the management of protocols within the SpEM framework presented in Chapter 5.

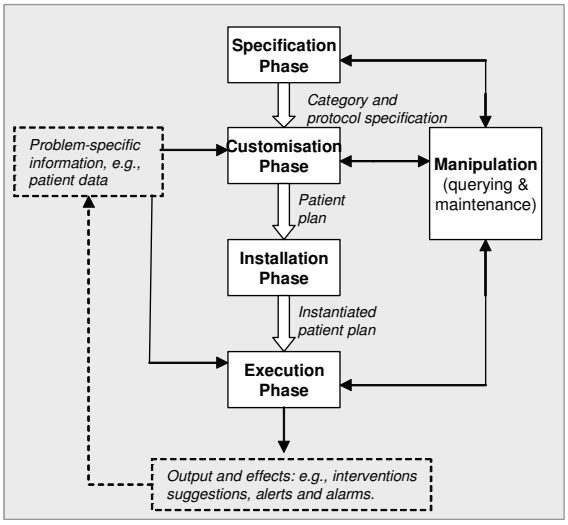


Figure 33 The execution flow for supporting the management of clinical protocols

The protocol management process consists of three main phases which are illustrated in Figure 33. The *specification phase* allows clinical protocols to be specified by using PLAN. The *customisation phase* is designed to ensure that the protocol is specific to given patient. The *installation phase* is responsible for the generation and creation of rule triggers within the DBMS for the implementation of the protocol's logic. The *execution phase* is the actual execution of a patient specific instance of the protocols. The *manipulation phase* enables operations and queries to be performed on objects in both the specification and execution phase. Thus, the *manipulation phase* conceptually permeates the other two phases. The next paragraphs discuss these phases in greater detail.

Specification Phase: During the protocol *specification* phase, 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. In the s*pecification phase,* domain knowledge, in the form of CGPs, is captured, formally represented and specified, and made persistent by storing it in a relational database. This phase requires the involvement of the domain expert, in this case, the clinician. The sources of the knowledge are mainly the domain expert and literature as interpreted by the domain expert. This phase results in the creation of problem categories with their associated domain knowledge (clinical protocol) specifications.

Customisation Phase: During the protocol *customisation* phase, 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. In the *customisation phase*, the domain knowledge is customised to a specific problem represented by the problem scenario object (PSO), in this case, the patient. This phase produces the instances for the individual patient.

Installation and Execution Phases: Figure 34 illustrates the detailed flow for plan installation and execution in the SpEM framework. During the test plan *installation* phase, the patient test plan is interpreted and set up to produce an instantiated patient test-ordering plan into the active DBMS. The 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, by the DBMS trigger mechanism. In the *installation phase,* the instances are installed, i.e., all the ECA rules are added to the rule engine and activated resulting in a ready-to-execute instance. The installation phase is tightly coupled to the *execution phase.* During the test plan *execution* phase, the test plan is executed. The test plan execution is driven by the ECA rule mode of operation.

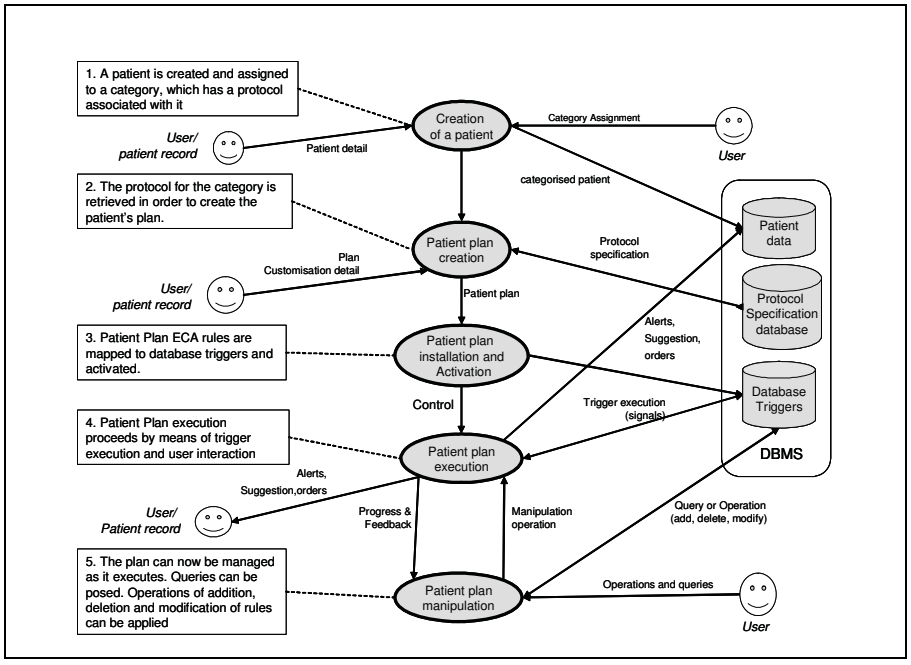


Figure 34 The execution flow for the creation, execution and manipulation of a patient plan in the SpEM framework

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. In the *execution phase,* the execution process proceeds in accordance with the ECA paradigm. The next section presents the queries and manipulation operations on specification and instances.

Manipulation Phase: There is a need to apply querying and manipulation operations to both specifications and the executing instances. The manipulation of the protocol and the patient plan constitutes the querying of the specifications and the history and state of plan execution. Manipulation also involves the dynamic addition, deletion and modification of the ECA rules that make up the protocol's logic. These operations allow adjustments and changes to be made to a protocol or a plan. The manipulation of the protocol and the plan specifications depend on how the rule and other plan components are specified and stored. As illustrated in Figure 33, manipulation of protocol information is relevant throughout the other phases of the execution flow.

The process of instantiating a protocol to create a plan is illustrated in Figure 34 and Figure 35. This process involves the criteria-based selection of schedules followed by the assignment of all the rules to one of two sets, i.e., the plan schedule containing static rules and the dynamic rule set. The evaluation of a schedule's entry criteria is done with respect to a specific problem scenario that is represented by a problem scenario entity (PSE) instance, the *problem scenario object (PSO),* which is the patient.

The algorithm for plan creation given the protocol specification is illustrated in Figure 36. In the clinical guideline and protocol domain, the PSO is the patient, who must satisfy the schedule's entry criteria in order for the schedule to be selected for incorporation into a patient plan. It should be noted that a protocol is not associated with any patient but with a problem-oriented (clinical) category while its instance - the plan - belongs to the PSO - the patient. The PSO must satisfy the category entry criteria in order for a protocol instance, or plan, to be created. In this work, we do not model the category entry criteria. Instead, this task is assumed to be the preserve of the domain expert decision-making process such as clinicians, in the case of a patient, who decide on the appropriate diagnosis and places the patient into a clinical problem category. Consequently, the algorithm in Figure 36 assumes that the PSO satisfies the entry criteria for the problem category for which the protocol was defined.

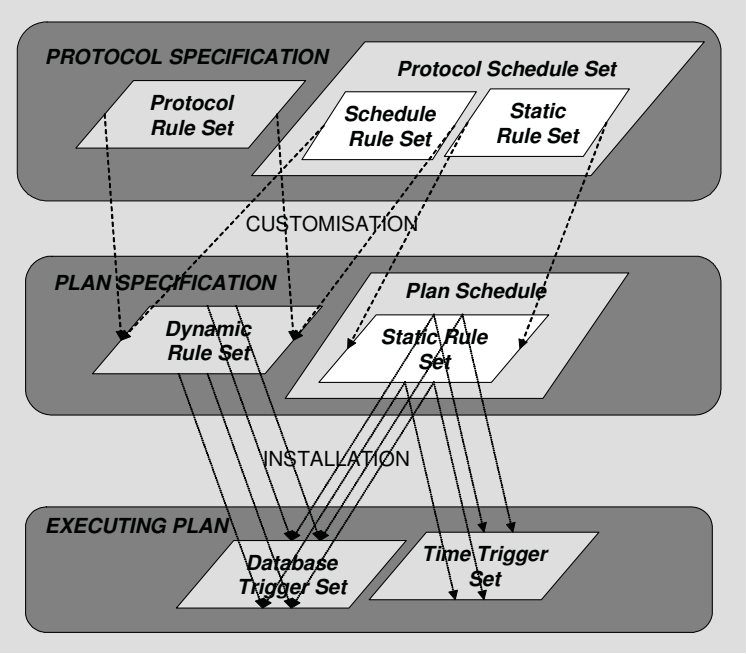


Figure 35 Components of protocols and plans and the mappings between the specifications and execution planes

Input:
Protocol (**Pr**); and
Problem Scenario Object (**PSO**)

Output:
Plan (**Pl**)

Algorithm:
1. Initialise:
   a. selected protocol schedule set **S** to be an empty set;
   b. plan **Pl** to contain:
      i. an empty plan schedule **PS**;
      ii. an empty plan dynamic rule set **PLDR**; and
      iii. a link to the **PSO**
2. For schedule **s** in **Pr** do:
   a. Evaluate entry criteria **CR** for **PSO**;
   b. If **CR** holds then add **s** to **S**;
3. For each **s** in **S** do:
   for each rule **r** in **s** do: addToPlan**(r);**
4. For each rule **r** in the protocol rule set **R** do: addToPlan(**r**);
5. Return **Pl**;

```
addToPlan(Rule r)
Begin
a. Customise r  w.r.t PSO;
b. If r is a static rule
        then Add r to PS;
        else if r is a dynamic rule
                then Add r to PLDR;
End
```

Figure 36 Algorithm for creating the protocol instance – the plan

Once the plan has been created using the algorithm in Figure 36, the plan is installed mapping each plan rule onto one or more database triggers. As illustrated in Figure 35, dynamic rules in a plan are mapped onto database triggers while static rules are mapped onto time triggers, which may be handled by a separate mechanism if the DBMS does not support time triggers.

The plan has a specification that is separate from that of the protocol specification. During execution, the plan is an evolving and changing object. Some of the changes experienced by the plan affect its specification and can also potentially affect the protocol specification through a background change propagation process. Thus, for every plan, the following hold with respect to the rule content of plan and protocol specifications:

a) either $plan \subset protocol;$
b) or $plan \cap protocol \neq \emptyset$.

The contents of a plan may change over time during its execution as rules are deleted, added, modified. The state of a plan may also change over time. Rules deactivated and activated remain in the plan and do not affect the contents of a plan. Consequently, plan P^I after time t_1 will be plan P^{II} and after a later time t_2 , it will be P^{III} thus,

$P^I \xrightarrow{t1} P^{II} \xrightarrow{t2} P^{III}$ where t_1 , t_2 are time intervals. P^I, P^{II} and P^{III} denote a plan at different time points.

The rule content and execution status of plan P at these different time points may or may not be the same. It is useful to enable information about the temporal evolution of a patient plan to be queried and replayed.

7.5. The Dynamic Management of Protocols

Management of a protocol, that is, the ability to query, add, delete, and modify components of both the specification and the running instances of the protocol is essential for the acceptability and sustainability of a computer-based protocol management system. Graphical visualisation of the protocol will greatly aid and simplify the task of protocol management. Figure 37 illustrates the types of dynamic protocol management scenarios that are necessary and the interaction between the real world and the protocol model.

Random changes and adjustments arise due to the need to correlate the patient's condition and the patient's executing test ordering plan. Some changes in patient's condition are reflected in previous test results as well as a clinician's observations, both will be

contained in the patient medical record. Random changes and adjustments provide flexibility based on adaptation of protocol specification and instances.

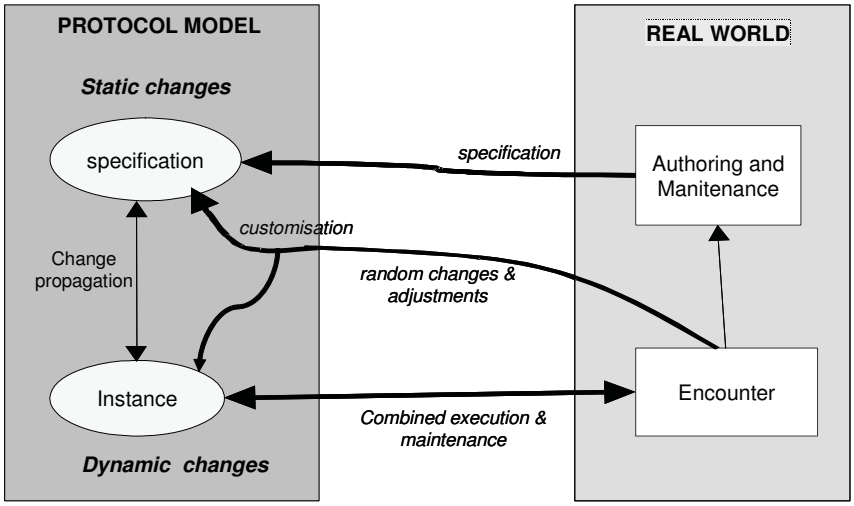


Figure 37 Dynamic protocol management: the interaction between the protocol management model and the real world - dynamic and static changes and interaction

Combined planning and execution is required when protocols cannot be specified completely in advance or when a complete specification of a protocol is inappropriate in the circumstances obtaining. Combining clinician's planning and protocol execution give rise to dynamically evolving protocol instances where decisions on tests to be ordered may be taken on the basis of already received previously ordered tests combined with further clinical observation. *Specifications* are created as definitions of new clinical protocols that are associated with new patient categories. There is a need to provide for continuous improvement in the form of updates of a protocol specification due to advances in medical knowledge and corrections of errors on already existing specifications. *Customisation* involves the adaptation of a protocol specification to a specific individual patient. This is realised by determining the schedule of tests whose filter conditions are satisfied by the patient, and then extending and refining this schedule based on patient-specific requirements.

7.6. Discussion

An architecture that allows ECA rules to be specified and executed for clinical protocol management can be used to provide a generic, portable and flexible mechanism for clinical protocol management. The clinical protocol management mechanism must be “adaptable” to allow easy extension or adaptation to handle new clinical protocols. It should be “portable”

in order to be easily re-used for different clinical protocols. It should be "generic" so that it cannot be tied to a particular database implementation.

Active database characteristics: Chaudhry *et al* (1998) used an active database in the implementation of a multi-step control of semi-conductor manufacturing. In this work, they identified important active database characteristics that were necessary to satisfy their application requirements. These characteristics are also important in the use of active databases for clinical protocol management and they include:

- Allowable event sources to include external massages and method invocations;
- Allowable actions must include sending messages to applications outside the database;
- Event structure must allow the definition of composite events; and
- Conditions to include function calls to allow external data analysis (Chaudhry, Moyne et al. 1998).

While in Chaudry et al's work (1998) time-constrained rule execution was not a strong requirement, supporting the management of clinical protocols strongly requires time triggers as well as temporary queries.

The Challenges of the creation of triggers to implement ECA rules: In the installation phase of executing clinical protocols, patient plans are mapped to one or more database triggers. Owens suggests two objectives to be attained when designing ECA rule based trigger code (Owens 1994). The first objective is the completeness of the ECA rule enforcement, which involves the identification of all events to which rule enforcement logic must respond. Data integrity could be compromised if completeness is not assured. Completeness is especially important if a single ECA rule is implemented by triggers from different tables. This happens when an object is constrained by an attribute of a related object – the constraining object- which is stored in a different table. The second objective is the maintainability of the trigger architecture. As ECA rules are enforced with multiple triggers for a single rule, and not always on a single table, we want the final trigger architecture to be maintainable and capable of responding to rule changes. According to Owens, the key to maintainability is to encapsulate highly cohesive procedures and functions into a re-usable, testable and manageable system, where one can trace from rule description to a procedure, and then back from a procedure to a rule. This allows rule tuning for domain changes and efficient management of large number of rule-based requirements (Owens 1994).

Customisation of clinical protocols: A rule set is customised so that each rule monitors a single patient. This guideline customisation that is based on the customisation of ECA Rules is justified by the need to take both medication information and patient status into consideration when specifying a rule (the rule language grammar should make such a provision). In most systems, all patients are monitored with the same rules and yet there is a need for each patient to be monitored with different alert rules according to his / her specific condition. Allowing ECA rules to be customised for each individual patient may have an negative impact on system performance due to an enlarged rule-base.

7.7. Summary

This chapter has presented the conceptual approach and architecture for supporting the enforcement of clinical protocols. By the use of the approach and architecture presented in this chapter, the execution of clinical protocols can be attained by means of a computer-based mechanism. The approach makes use of the ECA rule paradigm within database systems to drive the execution mechanism. The architecture is based on the wrapper principle in which rule support within the database system is extended within the wrapper. Protocol management functionality is provided at a higher level layer. The chapter also places the enforcement of protocols within the context of an execution flow for the support of protocol management. This chapter has also presented the method of creating an instance of a protocol, the patient plan, which forms the basis of the process of enforcing a protocol. Once the plan is created and executing, the interaction between the protocol management model and the real world occurs through a number of scenarios which were also presented in this chapter.

Chapter 8 Supporting the Manipulation of Protocol Information and Knowledge

8.1. Introduction

One of the important aspects of supporting computerised clinical protocols is to support the ability to dynamically perform manipulation of clinical protocol specifications and the execution process. As already pointed out, manipulation refers to performing operations as well as issuing queries against the clinical protocol information and knowledge. The subjects of manipulation include clinical protocol specifications, the plan execution process and the patient. Users should be allowed to pose various types of queries to obtain information about objects and their components in the system. When clinical protocols are specified, stored and later executed with respect to a specific patient, the maintenance operations of addition, modification and deletion need to be supported for specifications as well as for executing protocol instances. The clinical protocol information and knowledge associated with clinical protocol specification and execution needs to be made available through querying. Providing the ability to query the information and knowledge enhances the support for the flexible management of the protocols.

This chapter aims at presenting the framework and approach together with a language for supporting the manipulation of clinical protocol knowledge and information. Section 8.2 presents the framework for supporting manipulation. Section 8.3 presents the approach and method adopted in providing for manipulation within the overall protocol management framework presented in chapter 5. Section 8.4 presents the language, called TOPSQL, for querying and operations on the protocols and associated information. Section 8.5 presents a review of related work together with a discussion of the implications to this work. Section 8.6 summarises this chapter.

8.2. Framework for the Manipulation of Protocols

We now proceed to look at the framework for the manipulation (performing operations and querying) of information and knowledge associated with clinical protocols. We also identify

the views from which manipulation of protocol information and knowledge can be performed.

A clinical protocol, which is composed from sets of event-condition-action (ECA) rules for managing a patient, must be allowed to be dynamically manipulated. This means that the specifications, the executing instances (processes) and the effects (outputs) of the clinical protocols can be queried and operated on at any point in time. For this to be possible, it is necessary that the ECA rules that act as building blocks of the clinical protocols must also be dynamically manipulated. Consequently, the framework for manipulating protocols is based on the management model for active rule behaviour (Paton and Diaz 1999). The management model, as presented by Paton and Diaz (1999), has the four dimensions summarised in Table 8.1.

Table 8.1 Summary of the dimensions of the management model for ECA rules

DIMENSION	DESCRIPTION - CONTENTS OF DIMENSION
Description	Definition language, query language, or objects
Manipulation*	*Execution operations:* activate, deactivate, fire/signal
	Operations: add, delete, modify
	Query: retrieve, display, and navigate.
Adaptability	compile-time or run-time
Data model	relational, extended relational, deductive, object-relational, or object-oriented

In Table 8.1, the dimension marked (*) was presented as "operations" by Paton and Diaz for rule management. The term "manipulation" is preferred here in order to be in line with the terminology adopted in the framework presented in Chapter 5. The term "operations" is reserved for the operations of addition, deletion and modification. Within the framework, the dimension of the ECA rule management model are extended to the conceptual entities, which are composed of sets and subsets of ECA rules. The conceptual entities are the protocols and the plans. Table 8.2 presents the framework for the manipulation of ECA rule-based protocols. The aspects of the manipulation functions for protocols, patient plans and ECA rules are described in the next paragraphs.

Manipulation of Protocol Specifications: First, the specification should be able to be stored. Second, the stored specifications should be retrievable for the purpose of executing them with respect to an individual patient. In other words, the protocol specification should be allowed to be customised and linked to a patient to create a patient plan. Third, components of the specifications and the executing instances should be allowed to be

manipulated. Fourth, the specifications and instances should be queried, navigated and visualised down to component-level.

Table 8.2 Manipulation Framework for ECA Rule-based Clinical Protocols

DIMENSIONS OF THE MANIPULATION MODEL		DESCRIPTION FOR ECA RULE-BASED CLINICAL PROTOCOLS
Description	Definition Language	PLAN: a declarative language to define ECA rule-based clinical protocols and patient plans.
	Query Language	SQL and TOPSQL: Language to query both static and dynamic aspects of protocol rules down to individual ECA component level.
	Objects	a) *Inside the DBMS:* rules are schema objects described in the system catalogue b) *External to DBMS:* rules can be objects and so are their event-condition-action components
Operations	Activate	Applicable to 1) the patient plan, 2) the base schedule and 3) every rule
	Deactivate	Applicable to the same objects as "Activate"
	Add	a) Required for rules as well as sets of rules such as protocols, plans, and schedules; b) Applicable to specifications as well as to the instances of these specifications; c) Support for change propagation required between specifications and their instances; d) Action of a rule should be allowed to perform these operations on other rules.
	Delete	
	Modify	
	Signal/fire	Allow a rule to be invoked implicitly or explicitly by the user or by another rule
Adaptability	Compile-time	Allow changes to patient plan during its creation from the protocol specification
	Run-time	a) All operations to be dynamically applicable to both specifications and their instances at run-time; b) Support for one rule to manipulate or perform the rule operations on itself or other rules
Data Model	Relational	Definitions or specifications use the relational model; Execution mechanism is that of a relational DBMS.
	Extended Relational	None
	Deductive	None
	Object-Relational	Query, views and navigation of a protocol and patient plans will use object-relational features
	Object-Oriented	Modules (external to DBMS) that communicate with rules use the object-oriented data model

Manipulation of Patient Plans: The *static aspect* of a plan is essentially the protocol specification. It should be pointed out that operations on the plan must take into consideration the resulting effects on a patient. Operations on the protocol specification may bring about changes that may need to be immediately propagated to patient plans derived from it. The *dynamic aspect* of a plan refers to the plan's executing process. The plan has states and a life-span. Plan components can be manipulated through change propagation from changes made to a protocol. The plan's dynamic aspect can be queried along its state and time dimensions, which require history and snapshot maintenance. Graphical navigation or browsing and visualisation facilities could make manipulation easier. Another important feature that could enhance the ease of management of protocols is that for *re-playing* the execution of a plan for time periods that have already occurred.

Manipulation of ECA Rules: The ECA rules in a plan have states and life-spans. The plan rule sets should also be queried down to the event, condition and action components. The rules should be allowed to be added, deleted and modified in a dynamic fashion. A human user or another ECA rule can add, delete or modify another ECA rule. Another desirable feature that could aid in the management of the ECA rule-based protocols is that of allowing the execution process and, consequently, rule activity to be visualised.

The manipulation framework is supported through the problem domain, audit and explanation, manipulation, process and temporal views of managing protocol knowledge, which are illustrated in Figure 38 These five types of views have the following functionality: *The domain view* supports manipulation and queries that satisfy the requirements of the problem/application domain e.g. the management of clinical conditions by clinicians and patients. *The audit view* provides support for auditing the system and explanation of events and actions performed.

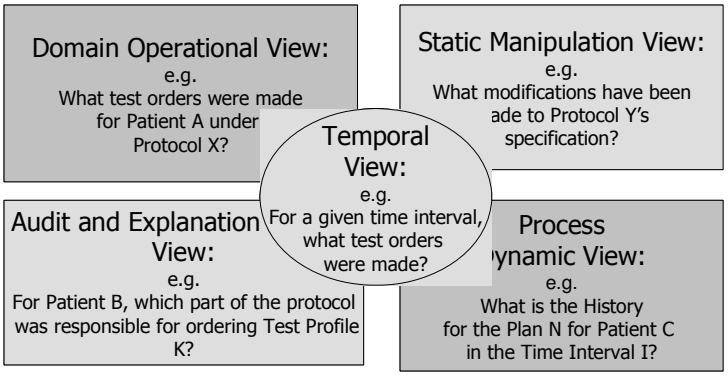


Figure 38 The view for the management of protocol knowledge

The management view provides support for administering the system through manipulation of specifications, instances and domain objects. *The process view* provides support for monitoring and controlling instance execution. *Temporal view* is a time-based view of the specifications and executing processes. The temporal view permeates the other four views. The manipulation language for supporting the these views for managing clinical protocol specifications and their instances is presented in Section 8.4.

Protocol specifications are created, parsed/compiled and the resulting protocol attributes are stored in the database. The stored protocol specifications are retrieved and customised using patient-specific attributes to create a patient plan specification. The patient plan execution is based on the ECA paradigm. The attributes of the patient plan specification and the results of its execution are also stored in the database. This approach forms a good basis for supporting the querying and manipulation of all aspects of clinical protocols using the SQL. This section has presented the concepts and framework for the manipulation of protocols. The manipulation framework is based on the management dimensions for active rule behaviour (Paton and Diaz 1999), which is extended to higher level domain entities such as patient plans. This manipulation framework is supported by information and knowledge management views that cover requirements for the problem domain, auditing, manipulation

and temporal queries. The next section presents the approach for accomplishing the manipulation framework.

8.3. Manipulation Approach

This section presents the approach and method for supporting the manipulation of ECA rule-based protocol specifications and their executing instances. The need for manipulation arises from the need to access protocol specifications, instances and objects for purposes of update, modification, replacement and obtaining information. The categories of operations and queries that are useful to perform on aspects of clinical protocols are illustrated in Table 8.3.

Table 8.3 Manipulation of protocols

Manipulation Object	PROTOCOL MANIPULATION Q-query, C-create, M-modify, D-delete, ADT-activate/deactivate/terminate, ✓ - defined, ✖ - undefined								
	Static				Dynamic				
	Queries	*Operations*			*Queries*	*Operations*			
		C	M	D		C	*M*	D	ADT
Category	✓	✓	✓	✓	✓	✓	✓	✓	✓
Protocol	✓	✓	✓	✓	✓	✓	✓	✓	✓
Patient	✓	✓	✓	✓	✓	✓	✓	✓	✓
Patient Plan	✖	✖	✖	✖	✓	✓	✓	✓	✓
Schedule	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rule	✓	✓	✓	✓	✓	✓	✓	✓	✓
Event	✓	✓	✓	✓	✓	✓	✓	✓	✖
Condition	✓	✓	✓	✓	✓	✓	✓	✓	✖
Action	✓	✓	✓	✓	✓	✓	✓	✓	✖

The protocol manipulation in Table 8.3 is categorised into *static* and *dynamic* aspects. The static aspect of protocol manipulation is targeted towards the specifications or definitions. Dynamic manipulation is targeted towards the history and process of the execution of patient plans. Within both the static and dynamic aspects of manipulation, there are *queries* and *manipulation* operations. Static operations and queries are applicable to the specification elements. Dynamic operations and queries are applicable to the executing instances. Manipulation operations are for creating (C), modifying (M), deleting (D), and activating or deactivating or terminating (ADT) elements. These manipulation operations include:

- manually activating or de-activating instances and associated protocol rules;
- creating specifications or their components;
- modifying the existing specifications and instances;
- deleting static and dynamic aspects of executing instances; and
- terminating an executing instance.

The operations and queries are achieved through the use of the query and manipulation language, TOPSQL, which is presented in the Section 8.4.

Manipulation of the category: A category is subject to both static and dynamic manipulation. Static manipulation is applied to the category specification or definition. Dynamic manipulation is applied to protocols, patients and executing plans within a category. Dynamic queries for a category retrieve information about patients and patient plans within the category. For instance, one may pose queries such as: *List currently active plans in a given category;* and *How many patients entered the category during the last two weeks?* The manipulation operations Activate, Deactivate and Terminate, when applied to a category, affect all patient plans within the category. The deletion of a category means the deletion of its protocol and all patients in the category together with their plans from the protocol system. Deletion should not be interpreted to mean physical deletion, instead it should be taken to mean flagging or labelling with *deleted* label or flag such that queries can still be applied while adding patients and patient plans to the deleted category will be disallowed.

Manipulation of the protocol: Static manipulation of a protocol affects only the protocol specification. Dynamic manipulation of a protocol affects both the protocol specification and all the patient plans derived from the protocol. When additions, deletion and modifications are made to the protocol, these changes may need to be propagated to the executing patient plans. It is worthy pointing out that a *version* concept for the management of the protocol specification is important but its investigation will be left to future work. The manipulation operations Activate, Deactivate and Terminate, when applied to a protocol, affect all patient plans derived from the protocol and is equivalent to the same operations on a category. Static queries on protocols are targeted towards the specification whereas dynamic queries on a protocol are targeted towards the patient plans that are derived from the protocol.

Manipulation of the patient: Both static and dynamic manipulation are applicable to the patient. The patient element can be created (C) to become a subject of the category and its protocol. The patient can also be deleted (D) if he/she is no longer the subject of the category or protocol. When a protocol is deleted, its associated patient plan is also deleted. Again, deletion is intended to mean flagging instead of physical removal. The dynamic modification (M) operation can also be performed on the patient. Activation, deactivation and termination operations on a patient affects the patient's plan, i.e., a patient is activated/deactivated/terminated when the corresponding patient plan is activated/deactivated/terminated within the system.

Manipulation of the patient plan: A patient plan, being a executing instance of a protocol, is subject to dynamic manipulation. Static manipulation of the patient plan is undefined since it has only a transient specification, which exists only during the creation of the plan. The manipulation of a patient plan will be performed separately without affecting the protocol from which the plan is derived.

Manipulation of the schedule, the rule and rule components: Static manipulation applies only the specification of the schedule and rule or its component in a protocol. Dynamic manipulation of a schedule, rule or rule component applies only to the plan schedule, rule or rule component. Since each of the ECA rule components cannot be executed alone as a separate module outside the rule context, the operations activation, deactivation and termination are undefined.

The effects of manipulation operations on an executing plan may be complex and require careful consideration. The effects are of two types: the effects and dynamics of rule insertion, deletion and modification in an already executing plan, and change propagation between plans and protocol specifications. Table 8.4 summaries the approach adopted for handling the effects of dynamic operations on test plan execution.

Table 8.4 Effects of manipulation operations on an executing plan, schedule and rule

		Effect (E) on execution state	
		Freeze/Deactivate	*Terminate*
Operation (O)	*Add*	(schedule, plan)	(plan, plan) (schedule, schedule)
	Modify	(schedule, plan)	(dynamic rule, dynamic rule)
	Delete	(schedule, plan)	(x, x)

The rows in Table 8.4 are dynamic operations and columns are effects on the dynamic operations on the executing plan. The entries of the table are given in the form (x,y), where x,y ∈ {plan schedule, rule} and (x,y) has the semantics that when the dynamic operation along that row is performed on x, then first perform the effect for that column on y. For example, to *add* a new schedule, *freeze* the executing plan and to *add* a new plan, *terminate* the currently running plan. It can be noted that 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.

What to do with a rule that could have been fired during a dynamic modification operation is also an issue that requires special attention. However, this issue does not arise when adding a new plan since no rule exists and is likely to fire during the process.

8.4. The Manipulation Language: TOPSQL

This section presents the protocol manipulation language, which has been named, TOPSQL - the **TOPS Q**uery **L**anguage. TOPS, the Test-Ordering Protocol System, is a prototype system that is presented in Chapter 9. The aims for the desing of TOPSQL are:

- easy to read and understand;
- easy to be used by domain experts ;
- easy to define a simple formal mapping to SQL;

The manipulation language, TOPSQL, consists of two main aspects, which are illustrated by using the Backus-Nuar Form (BNF) in Figure 39. The first aspect is the query language for querying the protocol specifications, the patient plans and their execution history. The second aspect of the manipulation language provides the manipulation operations on specifications and patient plans.

```
<TOPSQLstatement> ::= <TOPSQL-query> | <TOPSQL-operation>
```

Figure 39 The high-level syntax of the TOPSQL statement

In the next sections, the two aspects of TOPSQL are described. These aspects are the query language for protocols and patient plans and the language for manipulation operations in TOPSQL.

8.4.1. Queries on Protocols and Patient Plans in TOPSQL

The TOPSQL query is specified in the form of a SELECT statement whose syntax is similar to that of the SQL. Figure 40 illustrates the syntax of the TOPSQL SELECT statement for specifying queries on protocols and plans in BNF. The purpose of the SELECT statement is to retrieve information about the target item, <select-item>. The target item has, from the problem domain's perspective, a relationship with the reference item <reference-item>. The reference item is the source link subject to which the select item's information is to be retrieved. The target item must also satisfy the condition specified by <condition-spec>. The

result of the SELECT statement is the objects whose type is denoted by <select-item>, and satisfies the query

```
<TOPSQL-query> ::= SELECT <select-item> [SPEC] {FOR | FROM | IN} <reference-item> WHERE
[TARGET: <condition-spec>; SOURCE: ] <condition-spec>
<select-item> ::= {<target-obj-type> | <domain-dependent-obj-type>}
<reference-item> :: = {<source-ref-obj-type> | <domain-dependent-ref-obj-type>}
<target-obj-type> ::=  EVENT | CONDITION | ACTION | RULE | SCHEDULE | PLAN | PROTOCOL|
CATEGORY
<domain-depenedent-ref-obj-type> ::= TEST| RESULT | TEST-ORDER | PATIENT | …
<source-ref-obj-type> ::= RULE | SCHEDULE | PLAN | PROTOCOL| | CATEGORY
<condition-spec>  ::=<condition>|<time-interval>
<condition> ::= <SQL-condition>
<time-interval>::=<timestamp>,<timestamp>
<timestamp>::=<year>-<month>-dayOfMonth><blankspace><hour>:<minute>:<second>
```

Figure 40 Syntax of the TOPSQL query: The SELECT statement

condition. In the next paragraphs, the three main components of the SELECT statement are described in more detail.The query target, <select-item>, must have some form of relationship with the query source, <reference-item>. This relationship should be natural, clearly defined and important within the application domain. For instance, within the clinical test-ordering application domain, every test order is made with respect to a specific patient. Thus, an order for a test can be a query target while the patient with respect to whom the order is made can be a query source in a TOPSQL query.

Select item: The item to be retrieved, <select-item>, is the subject of the query statement. As illustrated in Figure 40, the <select-item> is either the target object type, <target-obj-type>, or the domain-dependent object type, <domain-dependent-obj-type>, which is meant to be retrieved by issuing this query. The target object type is one of the types defined as the basic components of the protocol specification model and includes the rule and its ECA components, the schedule, the protocol and the plan. The domain-dependent object type represents objects that are part of the problem domain. For example, in the domain selected for this Study, domain dependent object types include clinical laboratory test profiles, orders and results.

Reference item: In SQL, selected items are a list of attributes or columns of some reference relational tables specified in the FROM clause. In TOPSQL, instead of reference tables, a reference item, the <reference-item>, which is either a source reference object type, <source-ref-obj-type>, or a domain dependent reference object type, <domain-dependent-obj-type>, is specified. The selected item should have some form of dependency relationship or association to the reference item. Such a relationship must be meaningful and important in the system or domain. Typical relationships between the selected item and the reference item are:

- Selected item IS CONTAINED IN the reference item, e.g., an event is part of an ECA rule, and a schedule is part of a protocol;
- Selected item BELONGS TO the reference item, e.g., a plan is created for a patient, and a protocol is defined for a category.

The query condition: The query condition, <condition-spec>, is specified over the attributes of the selected item and also covers the attributes of the reference item. The simple condition generally involves the comparison of the relevant attribute to an absolute value. The compound condition would consist of two or more simple conditions connected by the Boolean connectives AND and OR. The query condition filters the items to be selected for retrieval. Only the items that satisfy the query condition are retrieved. To answer a TOPSQL query, the query processor must first apply the source condition to identify the source object. The source object is then used to determine the target object which should satisfy the target condition. The target clause specifies the condition that filters the query target and can be a logical condition over the query target's attributes or a time interval or window. The source clause is a condition that filters the query source. The following is an example of a TOPSQL query:

SELECT *ORDER* FOR *PATIENT*
WHERE
TARGET: 2004-*7-16 17:48:30, 2004-7-16 17:51:25;*
SOURCE: *PATIENT_ID=61*

This query reads: *Select all (test) orders made within the time interval from 17:48:30 to 17:51:25 on 2004-7-16 with respect to a patient whose ID is 61*. The target of this query is the ORDER object since the query is seeking for information on what (test) orders were made. The query source is the PATIENT object because we are focusing on a specific patient and we proceed from what we know about the patient, i.e., the PATIENT_ID. The target condition, in this example, is the time interval [17:51:25, 17:51:25] on a specific date, 2004-7-16, which is applied to the query target. It should be pointed out here that the granularity of the time interval can be arbitrary. The source condition identifies the specific query source, the PATIENT object.

Typical examples of queries: Table 8.5 presents some examples of various types of queries that can be specified using the query and manipulation language, TOPSQL. The

queries presented in Table 8.3 are of two types: The first type of TOPSQL queries can be directly translated into one or more SQL queries. The main difference between TOPSQL queries and their SQL equivalent is that TOPSQL queries specify entities as SELECT items while SQL queries specify attributes of the entities. TOPSQL queries return a set of objects in the form of attribute values that constitute specifications of the objects. The second type of TOPSQL queries are more complex and cannot be directly translated into SQL queries. These two types of queries are discussed in the next paragraphs. In Table 8.5, Q1 to Q6 are simple TOPSQL queries that can be translated into SQL queries. These queries can be answered directly by using the one or more SQL queries against either the tables or the views in the database. Also in Table 8.5, Q7 to Q9 are more complex TOPSQL queries that may not have a direct translation to one or more SQL queries.

Table 8.5 Examples of TOPSQL queries

QUERY	TOPSQL	SQL
Q1: For a given patient and day or time/date interval [t1, t2], what *test orders* were made?	SELECT *actions* FOR patient WHERE patient.id=k AND time > t1 AND time < t2	1. Select plan.id n From pl_plan Where (patient.id = k); 2. select rule_id, action, date_executed from pl_history_vw where (plan_id = n) AND (exec_date > t1 and exec_date < t2)
Q2: Which *rule* triggered a given *test order*?	SELECT *rule* FOR order WHERE order.id = n	select rule_id from pl_plan_rule_order_vw where pl_plan_rule_order_vw.order_id = n;
Q3: Given the *test order*, what *time* was the order issued?	SELECT time FOR order WHERE order.id = k	select exec_date from pl_plan_rule_order_vw where pl_plan_rule_order_vw.order_id = k
Q4: Given a *test order*, what is the *result* of the ordered tests?	SELECT result FOR order WHERE order.id = k	select test_id, result_id, result, result_date from patient_order_test_result_vw where order_id = 40
Q5: For a given *category*, which is the *protocol*?	SELECT protocol FOR category WHERE category.id = k	select id, name, date_created, creator_id, schedules n_schedules, protocol_rules n_rules, description from pr_protocol where category_id = 1
Q6: For a given *patient*, what was the *plan*	SELECT plan FOR patient WHERE patient.id = k	select id, name, protocol_id, date_created, current_state, state_change_date from pl_plan where patient_id=6
Q7: What was the *reaction* to a given *result*?	SELECT reaction FOR result WHERE result.id = k	No direct SQL equivalent (may include the firing and execution of other rules)
Q8: For a given *patient*, what was the *plan* at a given *time* point t or interval [t1, t2] ?	SELECT plan FOR patient WHERE (patient.id = k AND time = t) SELECT plan FOR patient WHERE (patient.id = k AND time > t1 AND time < t2)	No direct SQL equivalent (may require maintenance of *snapshots*)
Q9: For a given plan OR patient, show what happened during the time interval from time t1 to time t2	SELECT replay FOR plan WHERE (plan.id = k AND time > t1 AND time < t2) OR SELECT replay FOR patient WHERE patient.id = n	No direct SQL equivalent

The query Q7 retrieves information on the immediate *reaction* given to the specific event, i.e., the occurrence of a new *result.* In order for the system to answer query Q7, it will need to keep track of rules fired and executed as a result of a given event, e.g., "result-arrival". The query Q8 retrieves the plan or information about the state of a plan at a given time point, t, or a given time interval, [t1, t2]. To answer queries like Q8, the system needs to capture snapshots of every executing plan at every instant. It is interesting to note that it is possible that a snapshot may not have been taken at an arbitrary time, t, or during an arbitrary time interval, [t1, t2]. In such a case, the system may not be able to answer query Q8 unless a policy on how to handle such a situation is adopted. Figure 41 illustrates the policy adopted for guaranteeing that all queries of the type of Q8 always return a result.

In the Figure 41, **s***i* is the plan snapshots taken at time **t***i*. For example, plan snapshot s2 is taken at time t3. It can be noted that, in Figure 41, there was no plan snapshot taken at time t2. It can also be noted that there was no snapshot taken within the time interval [t4, t5]. If a TOPSQL query selects the plan at t2 or [t4, t5], the query could return no result.

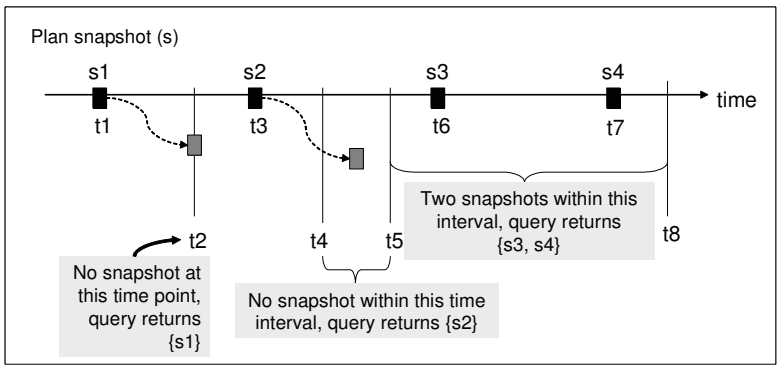


Figure 41 The policy adopted for snapshots and time intervals in TOPSQL queries for an executing plan

Since it is known that plan snapshot s1 existed at time t2 and plan snapshot s2 existed during the interval [t4, t5], the policy for guaranteeing that queries that involve such time points and intervals as t2 and [t4, t5] always return some results can be stated as:

> *If no snapshot exists in the database for a plan at a given time point or interval specified in a TOPSQL query, then the query returns the plan snapshot that was taken at a time point occurring closest before the time point or interval being sought in the query.*

The query Q9 in Table 8.5 triggers the re-play of the execution of a given plan, or a given patient's plan, during the specified interval of time. Queries like Q9 require a special

mechanism that maintains well-formatted execution logs, queries the execution log, and uses the query results in a simulation process that represents re-playing the plan that produced the execution logs. The replaying of a plan involves the simulation of event occurrences and actions execution that result from the firing of rules from which the plan is composed.

8.4.2. Manipulation Operations in TOPSQL

This section presents the manipulation requirements associated with both specifications and instances, the operations required to meet the manipulation requirements, and the language and syntax for manipulating protocols and their instances.

Operations on Specifications of Protocols and Plans in TOPSQL

Protocols and patient plans have specifications on which manipulation operations can be applied. Figure 42 illustrates the general syntax, in BNF, of the manipulation operations in TOPSQL.

```
<TOPSOperation> ::= <ADDcmd> | <DELETEcmd> | <EDITcmd> | <DISPLAYcmd> | <LISTcmd>
<ADDcmd> ::= { ADD | INSERT } <tops-object-type> TO <tops-object-type>”.”<attribute> <attribute-vale> AS “(“
<PLAN_spec>“)” | <tops-object-name>
<DELETEcmd> ::= DELETE <tops-object-type>”.”<attribute> <tops-object-name> FROM <tops-object-
type>”.”<attribute> <tops-object-name>
<EDITcmd> ::= EDIT <tops-object-type>”.”<attribute> <attribute-value> [FOR <tops-object-type>”.”<attribute>
<attribute-value>]
<DISPLAYcmd> ::= DISPLAY <tops-object-type>”.”<attribute> <attribute-value> [FOR <tops-object-
type>”.”<attribute> <attribute-value>]
<LISTcmd> ::= LIST <tops-object-type> WHERE <condition>
```

Figure 42 The BNF syntax of manipulation operations on static aspects of protocols

Manipulation operations include: the add command for adding new components to either protocols or plans; the delete operation for deleting a component from a protocol or a plan; the edit operation for allowing the modification of existing components of protocols and plans; display command for retrieving the specification of protocols and their components; and the list command for listing components without giving detailed specifications.

Creating Protocols and Plans: Protocols are created through the compilation of a protocol specification expressed in protocol specification language, PLAN, which has been presented in Chapter 6. Plans are created from the instantiation of a protocol by customising the protocol specification with respect to an individual patient. No explicit operations are provided for creating protocols and plans.

Adding or inserting a protocol or plan components: Once a protocol or plan is created, new rules or schedules may need to be added to it. The addition or insertion operation allows new protocol or plan components to be added to existing protocols or plans. An example ADD command is illustrated in Figure 43. In this example, a rule named r1 is to be added to a patient plan named p1234. Rule r1 monitors the event result-arrival and if it has occurred, checks if the result is above the normal value and if it is, the liver-investigation is suggested. It should be pointed out that the component to be added must be specified in PLAN.

```
ADD rule TO plan.name pl234
AS {
        RULE r1,
        ON result-arrival,
        IF result > normal-value
        DO suggest("liver_investigation")
}
```

Figure 43 Example ADD statement in TOPSQL

Deleting a protocol or plan component: It is important to support the deletion of components that are no longer required from a protocol or a plan. The delete statement must specify the type and name of the component to be deleted and the type and name of the entity from which the component must be deleted. Figure 44 illustrates an example of the delete statement. In this example, the rule named r1 is to be deleted from the plan named p1234. The DELETE statement can be performed against both the plan and the protocol specifications.

Modifying a protocol or plan object: To modify a protocol or plan component, the EDIT command is used. The command specifies the type and attribute and its value of the component to be modified and also optionally of the entity of containing the component to be deleted. Figure 44 illustrates an example of the EDIT command, which specifies that a rule named r1 for plan named p1234 needs to be retrieved for modification.

```
DELETE rule.name r1 FROM plan.name p1234
EDIT rule.name r1 FOR plan.name p1234
DISPLAY rule.name r1 FOR plan.name p1234
LIST rules WHERE plan.NAME = p1234
LIST rules WHERE patient.ID = 1234
```

Figure 44 Examples of DELETE, EDIT, DISPLAY and LIST statements

Display any item: The purpose of the DISPLAY statement is to retrieve and display to the screen the specification of the specified item. The DISPLAY command needs to specify

type and name of the object to be displayed as illustrated in Figure 44. In this example, the specification of a rule named r1 belong to the plan named p1234 is to be displayed.

List names of items: Sometimes it is useful to list items without showing their detailed specifications. For instance, one may want to list all plans, protocols, patients, rules or actions by name that exist within the system. Figure 44 presents two examples of typical LIST commands. A LIST command needs to specify the item type to be listed and the condition that must hold for each item in the list. The first example in Figure 44 lists, by name, all rules in the plan named p1234. The second example lists, by name, all rules in a plan that belongs to a patient whose ID number is 1234. If the plan associated with patient ID 1234 is p1234, then the two commands should produce exactly the same list.

Operations for Manipulating Dynamic Aspects of Executing Patient Plans in TOPSQL

Patient plans have static specifications as well as dynamic aspects in the form of the execution process and states. It is important and useful to provide a way to allow both the static and dynamic aspects of patient plans to be controlled and manipulated. The specifications of a plan can be manipulated by using the same operations as those for manipulating protocols. The execution of a plan is dynamically manipulated by using the commands that are presented in Figure 45.

The Figure presents the BNF syntax of the most important operations for manipulating the execution of a plan. These operations are ACTIVATE, DEACTIVATE, and STOP. The next paragraphs describe the three commands.

<DEACTIVATEcmd> ::= DEACTIVATE {rule | plan | patient} WHERE <tops-object-type>”.”<key-attribute> “=” <key-attr-value>
<ACTIVATEcmd> ::= ACTIVATE {rule | plan | patient} WHERE <tops-object-type>”.”<attribute> “=” <attribute-value>
<STOPcmd> ::= STOP {rule | plan | patient} WHERE <tops-object-type> <tops-object-name>

Figure 45 The BNF syntax of manipulation operations on the dynamic aspects of protocols

Deactivating a plan or its component: When a new plan is created from a protocol for a patient, it is activated automatically on installation. When a plan is deactivated, it exists but cannot monitor the patient nor can it execute any appropriate action since all its rules are inactive. Certain situations may arise during the care of a patient that may render it useful to deactivate a currently active plan. The DEACTIVATE command in TOPSQL makes possible the deactivation of a plan or any rule in an active plan. Figure 46 presents an example of the DEACTIVATE command.

```
a) DEACTIVATE rule WHERE rule.name = "r1"
b) DEACTIVATE plan WHERE plan.name = "p1234"
c) DEACTIVATE patient WHERE patient.id = 1234
```

Figure 46 Example DEACTIVATE command in TOPSQL

In Figure 46, a rule named r1 is to be deactivated in the first example and a plan named p1234 is to be deactivated in the second example. The third example indirectly deactivates the plan that belongs to a patient whose ID is 1234. Since a plan is composed of rules, deactivating a plan means every rule in the plan is deactivated.

Activating a plan or its component: There are situations in which it may be useful to activate a previously stopped or deactivated plan or rule. This is accomplished by using the ACTIVATE command. Since a deactivated rule or plan already exists within the system, the ACTIVATE command simply activates then to enable them to execute. Figure 47 presents two examples of the ACTIVATE command.

```
a) ACTIVATE rule WHERE rule.name = "r1"
b) ACTIVATE plan WHERE plan.name = "p1234"
c) ACTIVATE patient WHERE patient.id=1234
```

Figure 47 Example ACTIVATE command

In Figure 47, the first command activates a rule named r1 while the second command activates a plan named p1234. The third command indirectly activates a plan that belongs to a patient whose ID is 1234. Activating a plan is achieved by activating every rule in that plan.

Terminating or stopping a plan or its component: It is important to provide a user with the means of terminating or stopping a patient plan if it is deemed necessary. This is the purpose of the STOP command. The syntax and semantics of the STOP command are similar to those of the ACTIVATE and DEACTIVATE commands.

This Section has presented the protocol manipulation method in terms of the protocol manipulation requirements and the language, called TOPSQL, for expressing the queries and operations that address these requirements. In addition, this Section has also presented a description of how the effects of manipulation operations on an executing plan can be managed. The next Section presents the strategy for implementing the manipulation requirements and method presented in this Section.

8.5. Related Work and Discussion

Querying Protocol Information and Knowledge: An important aspect of the management of test ordering protocol and plan specifications and test plan instances is the ability to query the static protocol and plan specifications and the executing test plan instances. Issuing various queries to several relations in the protocol specification and execution databases and then combining these answers into one would be the suitable approach to providing answers to user queries. This problem is seen to be similar if not identical, to the problem of answering queries using views, also known as query rewriting or folding. Query rewriting or folding is the process of determining whether and how a query can be answered using a given set of resources (Qian 1996). Resources for answering queries include: materialised views; cached results of previous queries; or queries answerable by other databases. Gryz (1998a; 1998b) addresses the problem of Query Rewriting using Views for conjunctive queries and views in the presence of Inclusion Dependencies and both Inclusion Dependencies and Functional Dependencies. Most of the work in answering queries using views focuses on developing strategies that are targeted towards implementation within the DBMS and are invisible to applications and users. This work takes an application domain perspective. The TOPS query processor determines how to answer TOPSQL queries by a simple mapping of the query entities to their corresponding relational entities and views in the TOPS database and generates the appropriate set of SQL queries. Each TOPSQL query may span more than one relational table or view. The TOPS query processor must determine how to answer a given TOPSQL query by using the set of existing views and tables in the database. Currently, the TOPS query processor is very simple and still needs further work to exploit ideas from the research in query rewriting.

Support for ECA Rule Management in ECA Rule Systems: The management of the collection of rules in a database system is an important requirement within the framework for supporting the management of clinical protocols. For a large collection of rules retrieving a single rule becomes difficult. Therefore, there is a need for mechanisms for allowing the posing of queries against the rule-base. There is also need for facilities to manipulate (add, delete, modify) the rules to support the evolution of the collection of rules in the system. Hence it is interesting to review the extent of the support for rule manipulation in systems that support ECA rules.

Creation and Deletion of Rules: All active database systems support the creation and deletion of rules. What differs in database systems is whether or not these operations are allowed to occur while the database is online. Some systems assume that create and delete operations on rules are performed when the database is disconnected or off-line. Other systems allow the operations to be performed while
the database is online or processing other transactions. In the later case, there is a strong requirement for a special concurrency control mechanism to be provided.

Activation and Deactivation of Rules: The activation and deactivation of rules are common operations supported by most database systems. Since rules are persistent and may have a long lifespan, these operations are important for the management of rules because they allow some rules to be temporarily switched off without deleting them.

Explicitly Firing Rules and The Signal Operation: There are situations where the support of the so-called abstract or user-defined events is required. When these events occur, the signal operation is explicitly invoked to notify the rule system of the (external) event occurrence. Most modern commercial database systems do not support the signal operation.

Rule Modularisation and Stratification: Clinical protocols and patient plans are made up of rules. There is a need for a mechanism that allows rules to be logically grouped together to form a single entity that may have a separate lifespan. Some research prototypes, such as POSTGRES (Stonebraker, Hanson et al. 1988) and Starburst (Widom and Ceri 1996), introduced the concept of *rule sets* to allow rules to be grouped together or to be modularised. These rule sets could be created or deleted. Rules could also be added or removed from the rule sets. In POSTGRES, one command could be used to activate or deactivate all rules in a rule set. In Starburst, only rules in a particular set could be invoked for processing. In object-oriented systems where each rule is a first-class object, grouping of rules is natural because the usual structuring mechanism of classes and hierarchies (inheritance) are available to both rules and data. The addition or deletion of a rule can render a rule set incorrect. An evolution support mechanism is required to determine the effect of rule addition or deletion. Rule management can be aided by the stratification technique (Baralis, Elena , Ceri et al. 1996). In this technique, rules are partitioned into disjoint strata. The designer can then abstract rule behaviour by reasoning locally on each individual stratum separately and then reasoning globally on the behaviour across strata. The partitioning is done based on some criteria and should result in disjoint subsets of independent rules. Correctness criteria are established at a higher level of abstraction. Termination is an example of correctness criteria for which three

approaches were proposed for stratification: behavioural, assertional and event-based (Baralis, Elena , Ceri et al. 1996). In modern commercial database systems, grouping together of rules is not supported at all.

Querying Rules in a Database: In almost all database systems, rules are treated as named system objects such as tables in relational database systems. As a consequence, there are no comprehensive commands or languages that are provided for retrieving (and manipulating) individual rules or rule sets. Where such an attempt has been made, only very simple commands are available.

In most database systems, rules are stored as system objects. Further to this, very limited information relating to the rules is stored in the database system's catalogues. System objects are usually not the target of query languages such as the SQL for relational database systems. As a result, the query language may not be able to express useful queries on rules. For instance, a query may be required that cross-references rules and data. The following are examples of such queries:

- Which rules refer to column AGE of table PATIENTS in their condition?
- Which rules modify column DOSAGE of table MEDICATION in their action?
- This type of rules requires the access to the internal structure of the rule condition and action.

Object-oriented systems have the advantage that if rules are treated as first-class objects, as in HiPAC (Dayal, Blaustein et al. 1988), they can be queried using the standard query language for objects, e.g., the Object Query Language (OQL). In such a scenario, rules could be viewed in the same way as data within the database.

8.6. Summary

This chapter has presented a framework and approach for the manipulation of protocols and their executing instances the patient plan. The framework closely follows the management of ECA rules according to the dimension of the management model which was proposed by Paton and Diaz (1999). This chapter also present an approach to the manipulation of protocol that addresses the static and dynamic aspect of the objects or subject to be manipulated. The static aspect of manipulation deals with specification while the dynamic aspect deals with the execution process and the information it generates. This chapter also presented the

manipulation language, TOPSQL, which provides manipulation operations and querries to be performed on the protocol specifications, patients and patient plans.

PART III: DESIGN AND EVALUATION

Chapter 9 TOPS : Design and Implementation

9.1. Introduction

A prototype system called TOPS, the **T**est **O**rdering **P**rotocol **S**ystem, for managing clinical protocols within the domain of clinical laboratory test ordering by clinicians has been implemented. TOPS uses the SpEM framework and the MonCooS approach for supporting the management of clinical protocols to implement the functionality to specify and store protocols, permit the creation and execution of patient plans and support the manipulation of protocol specifications and patient plans. This chapter presents the design and implementation of TOPS.

The chapter is organised as follows: Section 9.2 presents the background to the requirements for TOPS. Section 9.3 presents the requirements specifications for TOPS. Section 9.4 presents the design of TOPS in terms of the functional, object and dynamic models. The section describes each of the core components of TOPS in detail. Section 9.5 describes the design of the support for the SpEM framework and the MonCooS approach. The sections also outlines how the design of TOPS tackles the challenges due to the lack of the comprehsive support for ECA rules in modern DBMS. Section 9.6 presents the overall architecture of TOPS. Section 9.7 presents a review of the design TOPS and compares it to related work. Finally, Section 9.8 summarises this chapter.

9.2. Background to the Requirements for TOPS

This section presents a background to the application requirements for TOPS. There is a need for a system that supports the management of computerised clinical protocols. The required system should provide a computer-based environment that allows users to specify clinical protocol knowledge, which the system represents formally; create executable instances of the specified clinical protocols for individual patients, which the system executes using a suitable mechanism; and manipulate, i.e., query and perform operations on, the knowledge and information relating to clinical protocol specifications and executing instances.

Within a clinical setting consisting of clinicians who are responsible for managing in- or out-patients who suffer from chronic diseases, e.g., diabetes mellitus types I and II, there is

a need to support the management of computerised clinical laboratory test-ordering protocols. The aim of such support is to improve the quality, efficacy and effectiveness of patient care as well as containment of costs. Local consensus-based test-ordering protocols for problem-based patient categories need to be specified in a formal manner and stored for later use. To each categorised patient, the appropriate test-ordering protocol needs to be applied with the necessary customisations. This application of the test-ordering protocol to the patient needs to be monitored and controlled over time. Since there may be many categories each with many patients, the monitoring and control of the application of the test-ordering protocol for each patient cannot be easily done by clinicians without automated assistance.

The underlying problem to be addressed by TOPS has two significant features that are worthy being highlighted. *Firstly,* the problem arises in the context of chronic diseases. Chronic diseases are usually associated with the three characteristics: they sometimes last for a lifetime; they progress with time - the patient either getting worse or getting better with time; and they need to be managed through monitoring and control. Clinical laboratory tests are one of the major means of monitoring chronic diseases. *Secondly,* the problem presents two levels of abstraction in the sense that there is a need to define a generic protocol for each category of patients. There is also a need to provide a more specific protocol or plan that is customised to suit the individual patient in the category. In other words, the protocol must occur at the category level and also at the patient level. This gives rise to the two levels of abstraction.

9.3. Requirements for TOPS

This section presents the application domain and technical requirements for TOPS. At a high-level, the clinical domain requirement is to provide computer-based assistance to healthcare professionals in the specification, storage, execution, manipulation and querying of domain knowledge and/or information for supporting the management of clinical test-ordering protocols for problem-based clinical categories of patients. This high-level requirement can be presented in terms of the following two major areas in which this assistance can be provided:

Specification: Assistance can be provided for healthcare professionals to specify and manipulate a computerised *test-ordering protocol* for a particular *category* of patients, e.g., patient categories for diabetes mellitus or its complications such as micro-albuminuria or

proteinuria. This assistance needs to be presented in terms of the *creation, storage,* and *manipulation,* i.e., the query and performance of operations on, the test-ordering protocol knowledge specifications for different categories of patients.

Execution: Assistance can also be provided for healthcare professionals to dynamically create and manipulate a *patient test-ordering plan* for an individual patient. This *patient test-ordering plan* is obtained for the patient from a test-ordering protocol of the particular *category* to which the patient belongs. The assistance to healthcare professionals needs to be presented in terms of the *creation, storage, execution,* and *manipulation*, i.e., the *query* and *performance of operations* on the individual patient test-ordering plans.

There are two interesting aspects that need to be understood about these two important domain requirements:

The levels of assistance required: It is very important to notice the relevance and difference of the two levels of assistance: at the first level, a test-ordering protocol is a generic specification of clinical protocol knowledge for a particular patient category; and at the second level, an individual patient will only be associated with a patient test-ordering plan, which is merely an instance of the more general protocol.

The emphasis on manipulation: It should be noted that in meeting the specification and execution requirements, it is important to provide for the manipulation, i.e., the issuing of queries and operations on the information and knowledge resulting from both the specification and execution tasks.

9.3.1. List of Requirements

From a technical perspective, the main requirements can be listed as follows:

1) A representation model is needed to represent the protocol or guideline knowledge;
2) a specification language is needed for test-ordering protocols and patient test-ordering plans;
3) a manipulation language is needed to query and perform operations on the information and knowledge associated with the test-ordering protocols ;
4) Software tools are needed to support the specification, storage, query and performing operations on computerised test-ordering protocol specifications; and
5) Tools are also needed to support the creation, from protocols, of patient clinical test-ordering plans and provide the mechanism for their execution and dynamic manipulation.

9.3.2. The UML Use Case-Based Requirements Model for TOPS

The TOPS Use Cases are illustrated in Figure 48. There are three system actors, the Administrator (Protocol Designer), the clinician (Patient Care Provider), the laboratory information system (LIS). There are five main use cases, namely, 1) create category, 2) create protocol specification, 3) perform manipulation (of protocol or plan), 4) create plan and 5) execute plan. The remaining six use cases each either extends, is included or generalise one of these main uses cases. The next paragraphs present descriptions of the use cases illustrated in Figure 48.

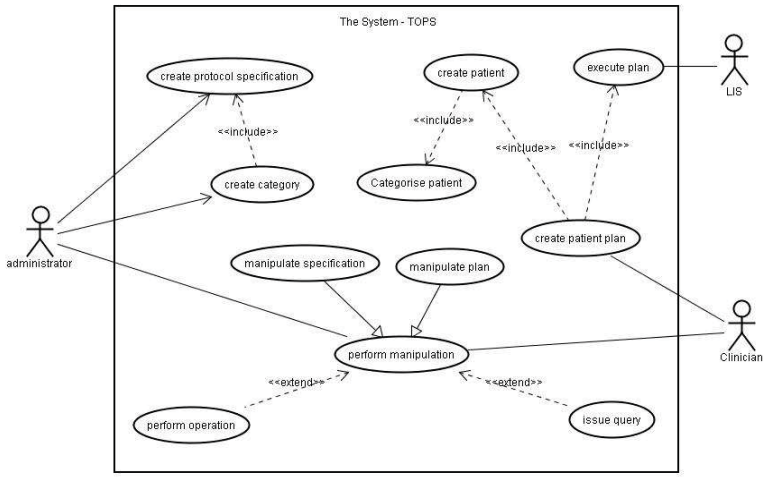


Figure 48 Use Cases for TOPS

Create category: The administrator actor creates the clinical category by providing a specification of the characteristics of the clinical problem being represented by the category. Each time a new category is created, its associated protocol must also be specified.

Create protocol specification: In the *create protocol specification* use case, the administrator actor creates the protocol specification for a category that has already been created. The specifications are expressed in PLAN language and are stored in a relational database table.

Create patient plan: The *create patient plan* use case is initiated by the Patient Care Provider, the clinician, who must as a prerequisite also create and categorise a new TOPS patient. The patient's plan is then created from the protocol associated with the patient's category. Customising the test protocol to the clinical circumstances or requirements of the patient require access to the patient's medical record. There is also a need to update the patient record with the resulting plan.

Create patient: This use case is included in the *create plan* use case. As part of creating a plan, a new patient must be created within the system. However, the system must also allow for a plan to be created for an existing patient provided that patient is re-categorised.

Categorise patient: The categorisation of a patient is important in the system because it determines the protocol that will be relevant for the patient. Consequently, the patient is categorised on being created within the system. In other words, an uncategorised patient cannot be allowed to exist in the system. It is important to point out that it is the clinician who makes the decision to place a patient into a category and the system only accepts this decision.

Execute patient plan: A patient plan is automatically activated soon after it has been created. For this reason, the *create patient plan* use case includes the *execute patient test plan* use case. During the test plan execution, test orders are sent to and their corresponding results are obtained from, the Laboratory Information System (LIS).

Perform manipulation: The administrator and the clinician can perform manipulation of either the protocol specifications or the patient plans. The clinician can also browse the issued orders and received laboratory results through the *perform manipulation* use case since the patient's local medical record is updated accordingly with the orders issued and results received. In the *perform manipulation* use case, the clinician queries the execution of the plan and can also modify the components of the plan. Dynamic modification of the patient test plan is an important aspect of the system, which needs special attention since it brings in the issue of dynamic modification of ECA rules which has received little attention in research related to the ECA rule systems in active databases. In the *perform manipulation* use case, the administrator actor queries, retrieves and modifies protocol specifications contained in the database.

Issue query: The issuing of queries is performed as part of the manipulation of protocol specifications and patient plans. The *issue query* use case is the specialisation of the perform manipulation use case. The querying is done using the TOPSQL, which has been described in Chapter 8.

Perform operation: operations are performed as part of the manipulation of protocols and patient plans. The perform operation use case is the specialisation of the perform manipulation use case.

9.3.3. Discussion of TOPS Requirements

The major requirement for TOPS is that of providing automatic application, at category level, of locally agreed clinical laboratory test-ordering protocols, customisable to individual patient circumstances. The satisfaction of this requirement provides, from both the clinical and laboratory operational standpoints, computerised protocol-based ordering of clinical laboratory tests. Excluded from the requirement is the provision of any attempt at human reasoning that leads to automatic clinical decision-making or diagnosis. TOPS's technical requirements include providing a specification language to specify investigation protocols, a database for storing these protocols, an execution mechanism based on the ECA rule paradigm, and a language to manipulate the specification and the execution process. When clinical guidelines are specified, they presuppose a clearly defined clinical problem to be addressed and their recommendations include the specification of well-defined patient categories to which the recommendations apply. TOPS' requirements do not include that of automating the task of deciding to which category an incoming patient should be assigned. Instead, TOPS's requirement is to leave this task to the domain expert. TOPS is required to accept categorised patients to whom it applies the protocol for the category to which the patient has been assigned and creates executable plans for these patients.

In summary, this section has spelt out the problem to which TOPS serves as a solution. The nature and characteristics of this problem has been described. The Section has also exposed the requirements that TOPS must satisfy in order to attain its aim and objectives. These requirements were described from both the application domain and the technical perspectives. The next Section focuses on describing the design and implementation of TOPS.

9.4. The Design of TOPS

This Section presents the design of TOPS. The model of TOPS from the functional, object and dynamic modelling perspectives is presented. The section describes the TOPS protocol specification database as well as the design of the TOPS mechanisms for the specification, execution and manipulation of protocols. The section also presents the architecture of TOPS. Finally the section ends with a discussion of the design of TOPS and a brief summary.

9.4.1. The Functional Model of TOPS

The functional model of TOPS is described in terms of a data flow diagram and describes what the system does. A data flow diagram (DFD) is a network representation of the system showing the functional relationships of the data that are computed by the system. The DFD is used to present a description of the high-level functions of TOPS. Figure 49 illustrates a DFD of TOPS, showing the main functional processes, data flows, the main data stored as well as the external entities of the system. The processes illustrated in Figure 49 are as follows:

1. *Managing patient categories:* The Category Designer creates a new category, which is stored in a data store, which can be queried and modified.
2. *Creating a protocol specification:* The Protocol Designer creates a new protocol specification, which is stored in a data store.
3. *Managing a protocol specification:* The Protocol Designer may query and modify existing protocol specifications.
4. *Creating a patient plan:* An individual patient's test ordering plan is generated from a test protocol for the clinician. The process of building a patient test plan required data from the patient's medical record.
5. *Managing a patient plan:* The clinician queries and gets responses on test plans. The Clinician can retrieve and update or modify the patient test plan specifications.
6. *Executing a patient plan:* During test plan execution, plan rules are set up for monitoring, execution and feedback on execution is produced. When a patient test plan is executed, appropriate test orders are issued and test results are received from clinical laboratory.

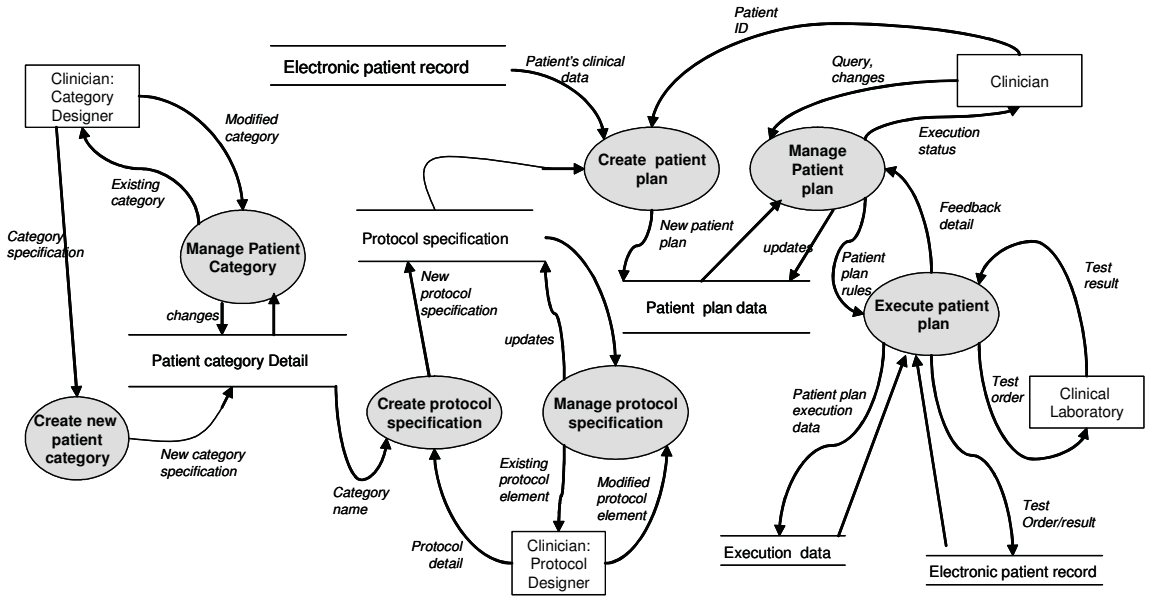


Figure 49 Data flow diagram for TOPS with a focus on the domain of clinical laboratory test-ordering protocols

Table 9.1 Table of data flow for the DFD of Figure 9.3

FUNCTIONAL MODULE	INPUT	FROM	OUTPUT	TO	COMMENT
Create patient category	Category specification	Designer	Formatted category specification	Patient category data store	Designer creates a new category, which is stored in a data store
Manage patient category	• Changes • Query	• Designer • Category data store	• Formatted changes • Query response	• Category data store • Designer	Designer queries and modifies existing category specifications
Create protocol specification	• New protocol specification • Category name	• Designer	• Formatted new protocol specification	• Protocol specification data store	Designer creates a new protocol specification, which is stored in a data store
Manage protocol specification	• Changes • Query	• Designer • Protocol data store	• Formatted changes • Query response	• Protocol data store • Designer	Designer queries and modifies existing protocol specifications
Get patient test plan	• patient ID • patient's medical record • protocol specification	• Clinician • Electronic patient record • Protocol specification data store	• New patient test ordering plan	• Patient test plan data store	An individual patient's test ordering plan is generated from a test protocol for the clinician
Manage patient test plan	• Query • Patient test plan specification • Execution feedback	• Clinician • Patient test plan specification data store • "Execute test plan" process	• Query response • Changes • Test plan rules	• Clinician • Test plan specification data store • "execute test plan" process	The clinician queries and get responses on test plan, the test plan specification is retrieved and updated, test plan rules are submitted for execution and feedback on execution is received
Execute patient test plan	• Test plan rules • Test results	• "Manage test plan" process • Clinical laboratory	• Test orders • Feedback on execution • Execution state data	• Clinical laboratory • "manage test plan" process • execution state data store	Patient test plan is executed, appropriate test orders are issued and test results are received from clinical laboratory

Table 9.1 presents a detailed description of each process illustrated in Figure 49 in terms of the inputs and where they are coming from, and the outputs and where they are going to from the process.

9.4.2. Entity-Relationship and Object Models for TOPS

This section presents the static model of TOPS in the form of an entity-relationship model for the most significant entities in the system and the object model for the most significant classes within the system.

The TOPS Entity-Relationship Model

Figure 50 illustrates the entity-relationship model for TOPS in the notation of Chen (1976). The entity-relationship model in Figure 50 expresses that patients are placed into clinical categories. A separate protocol is specified for each category. For each categorised patient, a patient plan for ordering clinical investigations is created as an instance of the category's protocol.

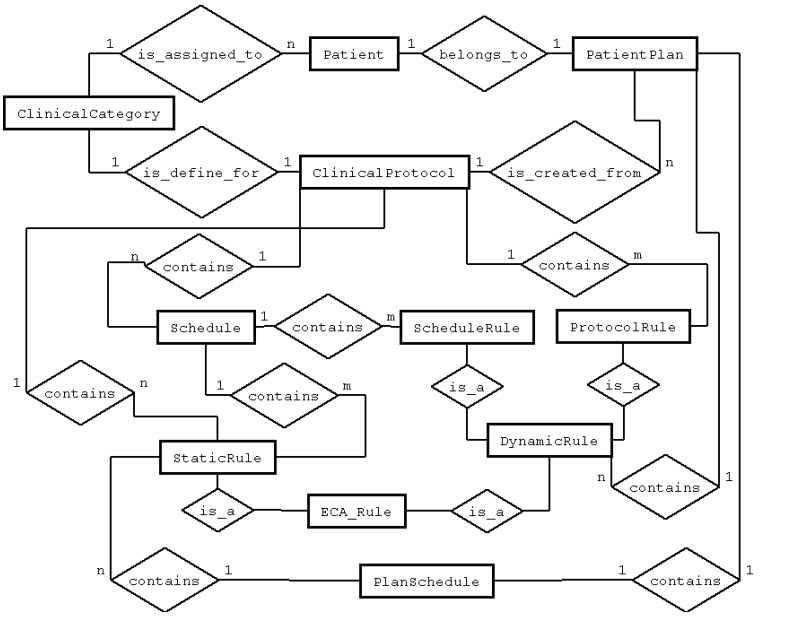


Figure 50 The entity-relationship model for the specification of the ECA rule-based protocols.

The clinical protocol specifies circumstances for ordering each laboratory tests through a set of protocol rules and schedules. Each schedule is composed of a set of static rules and schedule rules. Protocol rules and schedule are the two types of the dynamic rule. The dynamic rule and static rules are two types of the generic ECA rule. When the protocol is

instantiated with respect to a patient to create the patient plan, only two rule sets are created: the set of static rules, which form the plan schedule, and the set of dynamic rules, which is created from the sets of schedule and protocol rules.

The Object Model for TOPS

Figure 51 illustrates the object model for the prototype system TOPS, which is made up of the following components:

The TOPS patient: The Patient class provides for the specification of patient demographics as well as a link to the patient's category. The Patient class also provides methods for adding the patient details to the database, managing the patient and creating the patient plan. The PatientHistory class allows the system to maintain the history of the patient in the database while the PatientState class provide facilities for maintaining the state of a patient within a protocol execution process.

The clinical category: The Category class has attributes that specify the clinical category for which each protocol is defined and to which each patient is assigned.

*The clinical protocol specification***:** The Protocol class models the protocol specification and its instances represent complete specifications of protocols. The Protocol class has attributes whose types are of the following classes: ProtocolHeader, PScheduleSet, PSRuleSet and PDRuleSet. The ProtocolHeader class holds the attributes of the protocol. The PScheduleSet class is a container for the set of schedules within the protocol. Each schedule is an instance of the PSchedule class and contains, as attributes, an entry-criteria, a set of static rules and a set of dynamic rules. The entry–criteria are a special type of a condition (PCondition class) that must be satisfied by a patient in order for the schedule to be selected for inclusion in a plan. The PSRuleSet class is a collection of static rules while the PDRuleSet is a collection of dynamic rules, which are not part of any of the schedules in a protocol. Each element in a PSRuleSet collection is an instance of the PSRule class, which is a static rule. Also, each element in a PDRuleSet collection is an instance of the PDRule class, which is a dynamic rule. From Figure 51, it can be seen that both the PSRule class and the PDRule class are specialisations of the Rule class. The Rule class has, as its attributes, an action of type PAction and a condition of type PCondition.

The patient plan and its execution mechanism: The TOPlan class, which is a specialisation of the GenericPlan class, serves the purpose of an intermediate mechanism from the protocol specification to the executing plan within the DBMS' trigger mechanism.

The GenericPlan class, and, by inheritance, the TOPlan class, has two important attributes: the first one is the schedule of type Schedule class and the second one is a set of the type DRuleSet class, which is a container for instances of dynamic rules of the type DRule class. The Schedule class contains a schedule rule set of type SRuleSet class, which is a container for static rules of type SRule class. At the implementation level, static rules are implemented by using both time triggers and Oracle triggers. Time triggers are implemented in a Java-based mechanism. Dynamic rules are implemented through Oracle database triggers. Appendix J presents an illustration of how a rule from the case study in Chapter 10 is translated to the Oracle database trigger. The resulting database trigger incorpates appropriate customisations.

The system database access mechanism: To access the database, TOPS makes use of the three classes: SQLOp, TopsDBAccess and TDBC. The SQLOp class dynamically generates SQL statements required to accomplished tasks that need to access the database. The TopsDBAccess class manages connections to the database. The TDBC class uses the Java Database Connectivity (JDBC) to create connections to the appropriate database server.

The mechanism for rule communication with modules that are external to the DBMS: The execution of triggers within the database system is complemented by the ECA rule extension mechanism outside the DBMS. The link between triggers in the DBMS and extension modules outside the DBMS requires a communication link, which cannot be achieved through JDBC. Outside the database system, a listener, an instance of the DBMsgListener class, listens at a secure port and on detecting an incoming connection, it invokes the reader, an instance of the DBMsgReader class, to read the message received. Once message reading is complete, an instance of the DBMsgProcessor class analyses the message and invokes an instance of the ExternalAction class in order to execute the external action required. The DBMsgListener functions in the same way as an HTTP server. Every trigger that executes within the DBMS invokes a Notifier that connects to the DBMsgListener using the same strategy as an HTTP client. On establishing the connection to the DBMsgListener, the Notifier sends attributes of the patient and the ECA rule to be executed outside the DBMS. Appendix J presents further details on the TOPS mechanism for allowing database triggers to communicate with applications outside the DBMS.

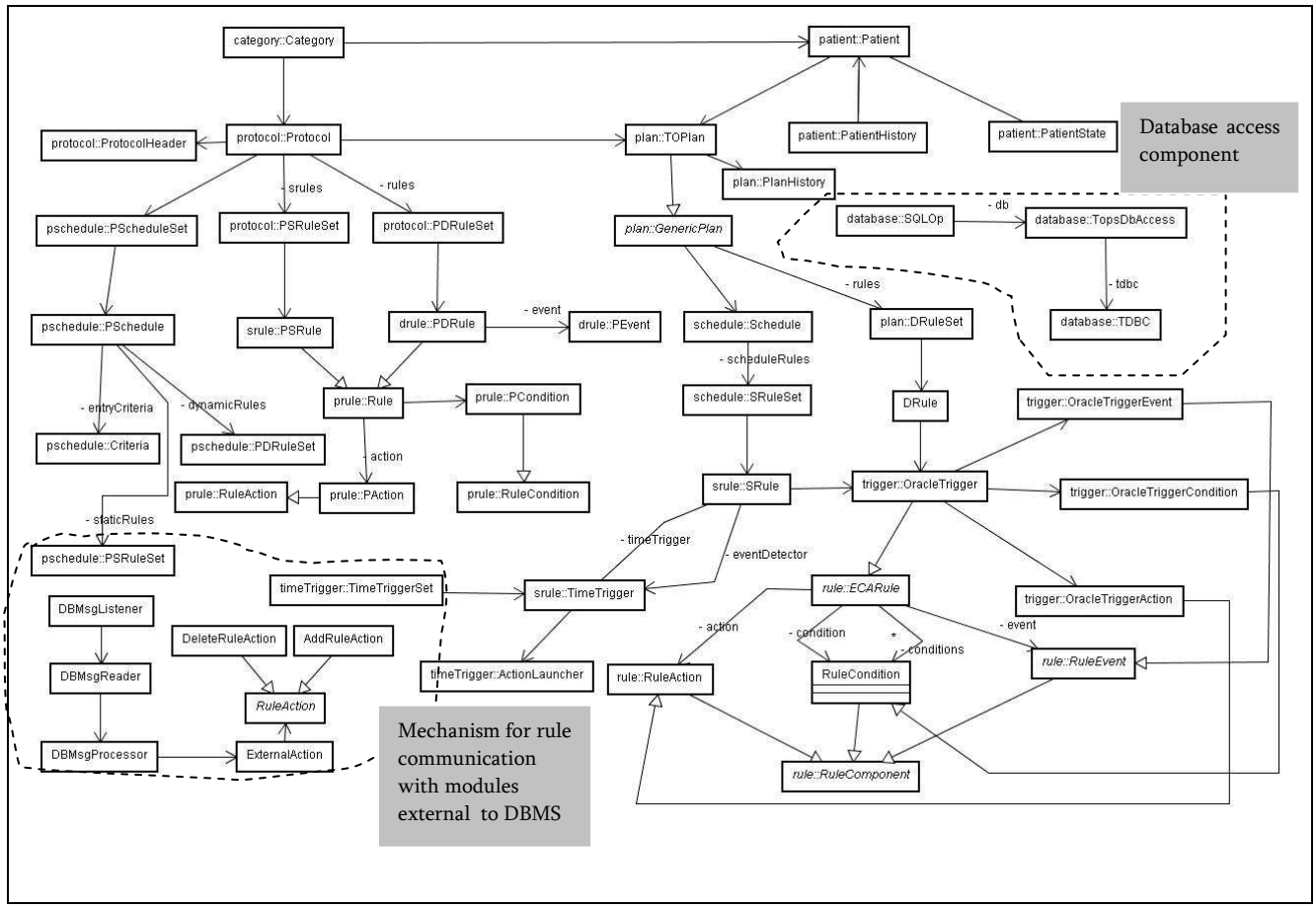


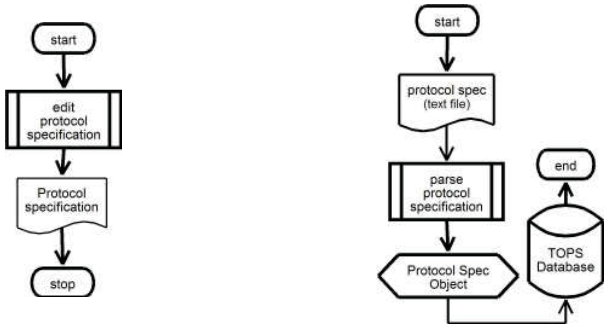
Figure 51 The core object model for TOPS incorporating the Category, Patient, Protocol and Plan classes

9.4.3. Dynamic Model of TOPS

This Section presents the dynamic models of important processes in TOPS. The UML sequence diagram is used to model the key functionality of TOPS.

Creating a Protocol Specification

A protocol specification is created using a traditional text editor. The specification is written in PLAN language described in Chapter 6. The flow chart for the process of creating a protocol specification in TOPS is illustrated in Figure 52.



(a) Flow chart for the creation of a protocol specification in TOPS

(b) Flow chart for the process of parsing a protocol specification

Figure 52 The dynamic model for the protocol specification in TOPS

The result of the protocol editing process is a plain text file, which will be the input to the PLAN language parser. The main outputs of the parser are an instance of the Protocol class, which is an object-oriented representation of the protocol specification, and relational database version of the specification. If it is given a Patient class instance, the protocol object can permit the creation of a plan for the patient.

Creating the TOPS Patient

For any plan to be created and executed, first a TOPS patient must be created. The category to which a patient is assigned must exist prior to the creation of a new TOPS patient. The sequence diagram in Figure 53 illustrates the process of creating a patient in TOPS.

To create a TOPS patient, a message, select(), is sent to the TOPS category object to allow a category to be selected from those available. The patient will be assigned to the selected category. Next, after obtaining patient demographics, a <<create>>Patient() message is sent to create a new object instance of the Patient class, which sends the add() to itself to add the patient to the TOPS database. At this point, new TOPS patient will have been created and is ready to have a plan created for him/her using the protocol associated with the category to which the patient belongs.

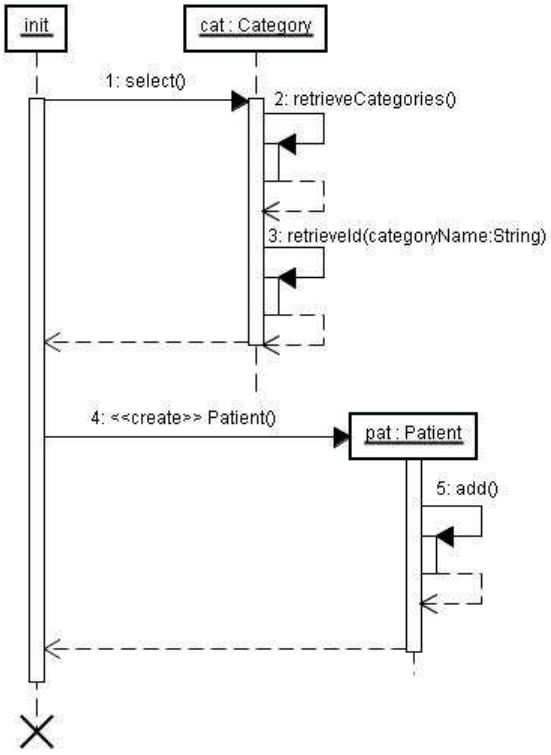


Figure 53 Sequence diagram for creating a TOPS patient

Changing the Category of the TOPS Patient

As has been noted earlier, a TOPS patient is categorised on creation and cannot exist in TOPS without being associated with a TOPS category. It is permissible to assign an existing patient to a new category if it is necessary to apply a new protocol to the patient. The sequence diagram in Figure 54 illustrates the process of changing the category of the patient in TOPS.

First, the select() message is sent to a Category object to allow for a new category to be selected. Second, the list of all categories defined within the system is retrieved through the message, retrieveCategories(). Third, the ID number for the selected category is retrieved from the database by using the retrieveId() message. Fourth, a setCategoryId() message is sent to the Patient object so that it can update the ID number for the new category to which the patient has been re-assigned. From this point on, the patient is associated with this new category and any attempt to create a plan for this patient will automatically use the protocol that is associated with this new category.

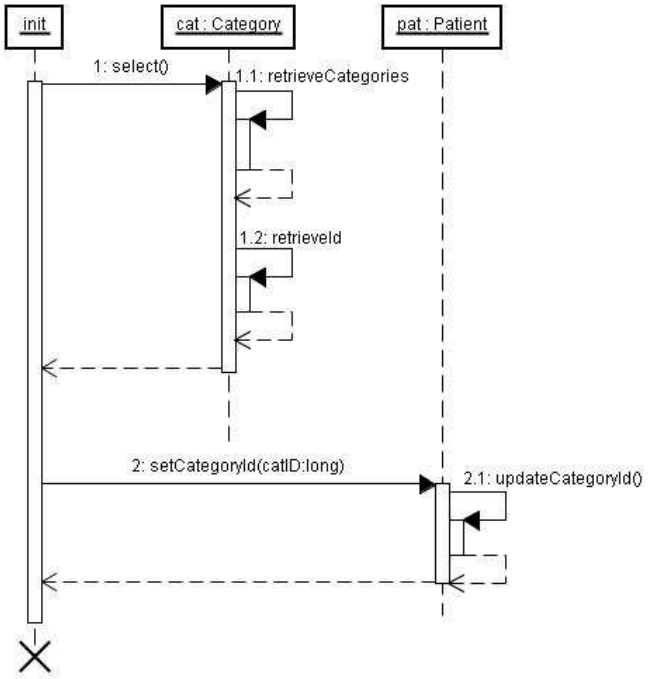


Figure 54 Sequence diagram for changing the category of a TOPS patient

Creating the TOPS Patient Plan

When a patient has been created in TOPS, a plan can be created for the patient. The flow chart for the process of creating a TOPS patient plan is illustrated in Figure 55. An appropriate protocol specification is retrieved from the TOPS database into the protocol object, which provides for methods to manipulate the specification including that for creating a patient plan from the specification. The process of creating a patient plan produces a plan object, which installs and activates the plan in the TOPS database.

The sequence diagram in Figure 56 illustrates the process of creating a patient plan in TOPS. The initiating urgent sends a message to create the patient plan, createPatientPlan(), to the patient plan manager, PlanManager, instance. The PlanManager then performs the following actions:

1) An instance of the Category class is created, <<create>>Category();
2) The message, select(), is sent to the Category object to allow a category to be selected and its ID number to be retrieved from the database;
3) The message, getId(), is sent to the Category object to retrieve the category's ID number;
4) If the Patient object is not supplied as a parameter to the createPatientPlan() message, an instance of the Patient class is created;

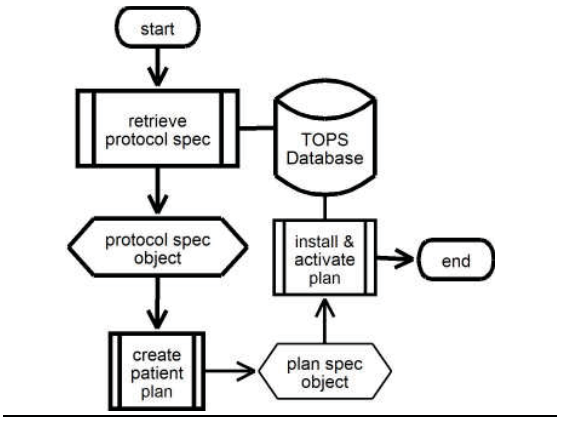


Figure 55 Flow chart for the process of creating a TOPS patient plan

5) A plan object is created as an instance of the TOPlan class, <<create>>TOPlan(); and
6) The message, create(), is sent to the plan object with the patient object as a parameter. This starts the process of creating a patient plan from a protocol specified for the category to which the patient belongs.

To create a plan from a protocol, the plan object creates an instance of the Protocol class, <<create>>Protocol() and sends a toPlan() message to convert a protocol specification into a TOPS patient plan. To achieve this, the protocol object proceed by performing the following actions:

1) The message, toPlanDynamicRuleSet(), is sent to the dynamic rule set object, drSet, which contains the dynamic rules in the protocol. A similar message, toPlanDynamicRules(), is sent to the object containing a set of protocol schedules. The output of these two messages is a combined set of the plan version of all the dynamic rules that were contained in the protocol;
2) The message, toPlanStaticRuleSet(), is set to the object containing the set of static rules in the protocol, psrSet, to create the plan version of the protocol static rules;
3) The message, toPlanSchedule(), is sent to the object instance of the PScheduleSet class, which is a set of schedules in the protocol. This message has the effect of the creation of a plan schedule that contains only static rules.
4) The PScheduleSet class creates an instance of the plan schedule, <<create>>PSchedule(), which contains static rules from the protocol schedules and also from the protocol static rules. The plan static rules are created by sending the message, toPlanStaticRuleSet(), to protocol and schedule instances of the PSRuleSet class.

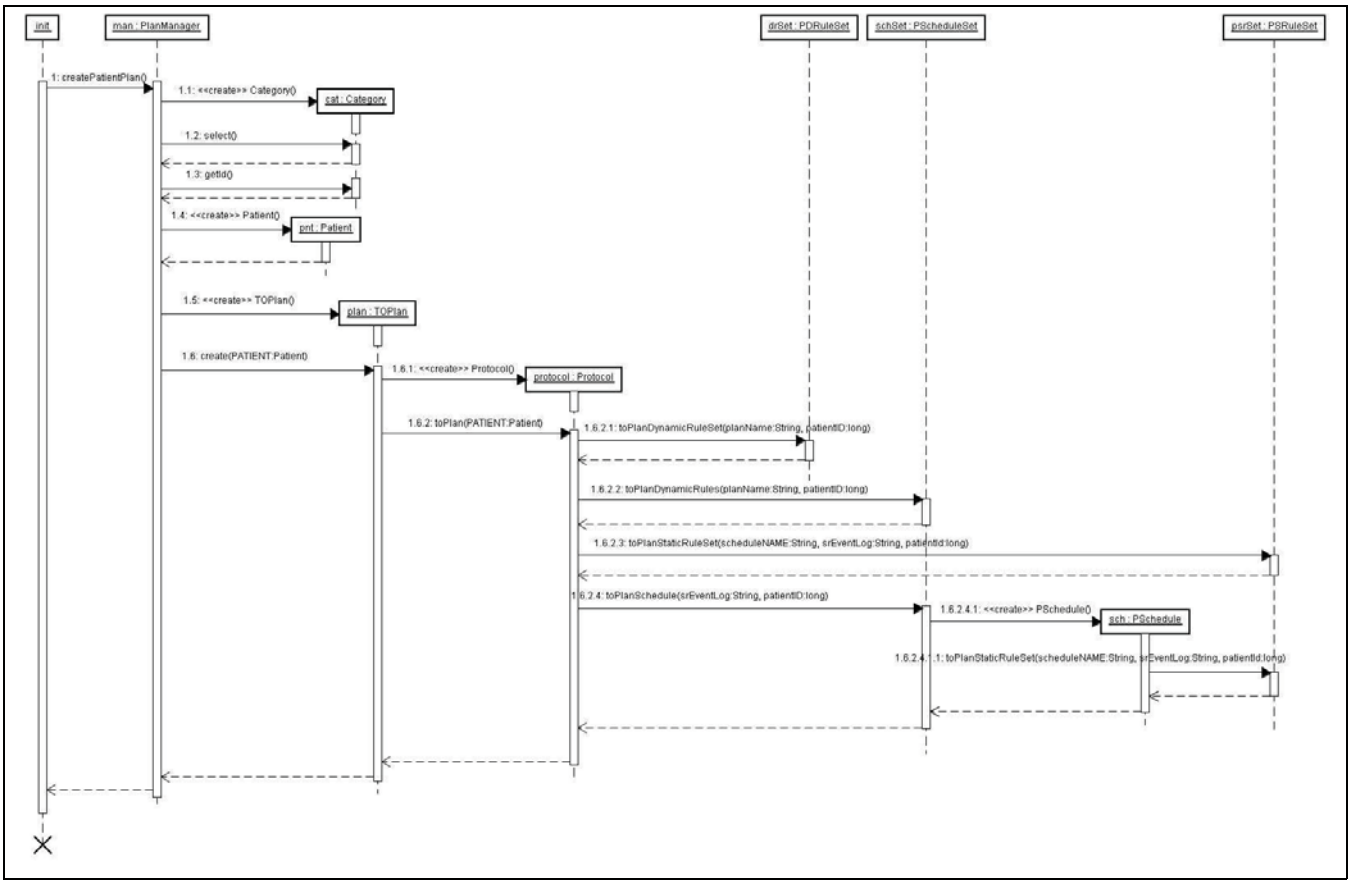


Figure 56 Sequence diagram for creating a patient plan in TOPS

Once the dynamic and static rule sets are created, the patient plan is assembled and becomes ready for installation, activation and execution within the DBMS.

Querying in TOPS

The information and knowledge relating to protocol specification and to an executing patient plan is in TOPS can be queried. The process of querying this information and knowledge in TOPS is illustrated in the sequence diagram of Figure 57. And the instance, cmd, of the TOPS command line facility, TOPSCmd, is initiated by sending the start() message, which allows it to display the command line prompt. At this prompt the user types a query using the manipulation/query language, TOPSQL. To handle the query, the TOPS command line instance creates an instance of the manipulation language processor, TOPSQL class, and passes on the query statement as the argument. The query processor first parses the query statement and then instantiates the query handler, <<create>>SELECTCmd(), which analyses the query condition before it executes the query by invoking a more specialised query handler such as the plan query, PLANQuery class, which handles all queries relating to a TOPS plan.

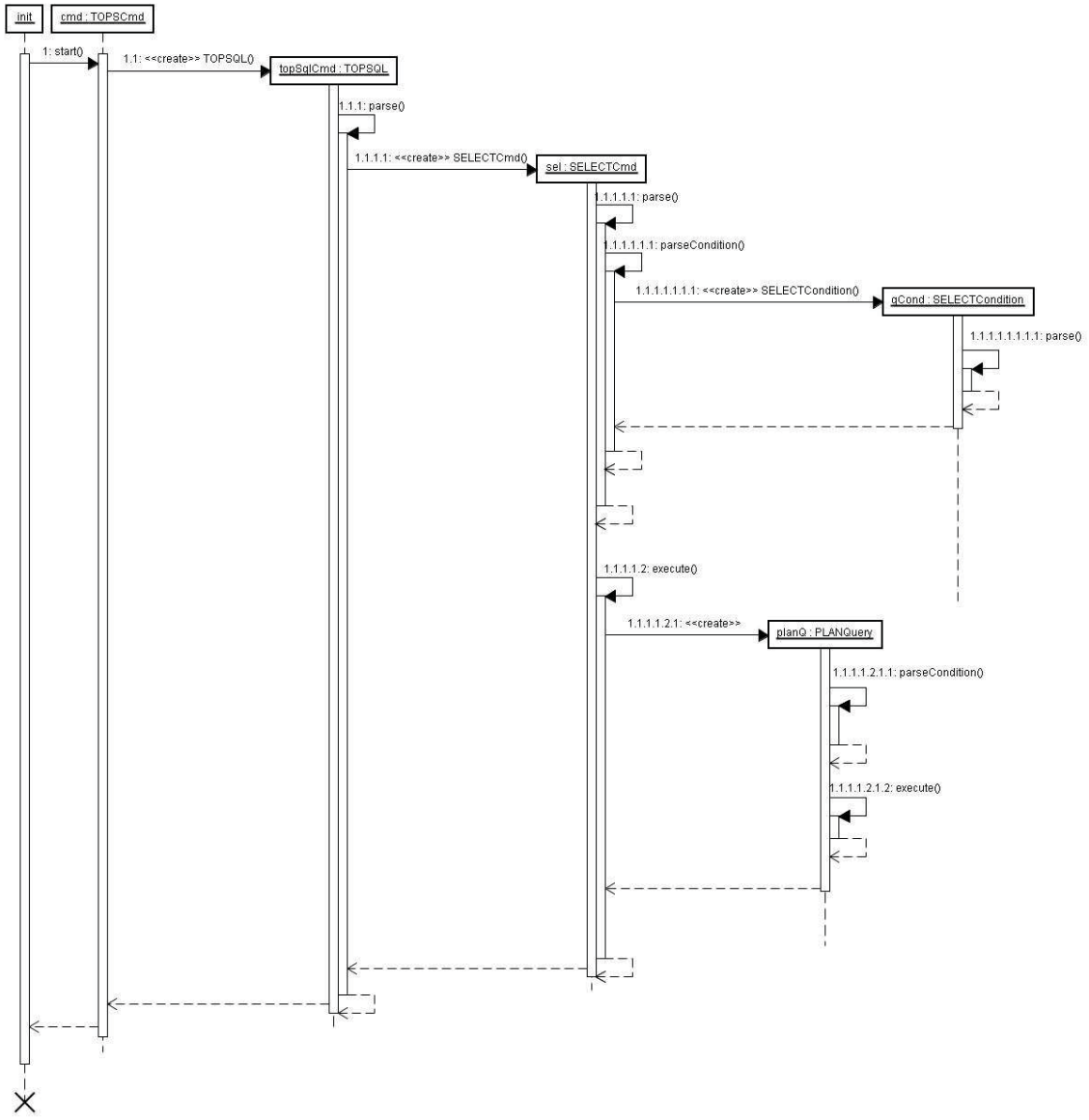


Figure 57 A sequence diagram for issuing a query in TOPS

Other specialised query handlers include the PATIENTQuery class and the PROTOCOLQuery class. Each specialised query handler uses one or more SQL queries to get information from the TOPS database to answer the original TOPSQL query.

Performing a Manipulation Operation

Figure 58 illustrates a sequence diagram for the process of performing an operation on a TOPS plan. Operations to manipulate protocol specifications, plans and patient information can be specified by the user through the TOPS command line facility, which is an instance of the TOPSCmd class. The process of performing a manipulation operation proceeds in a similar manner to that of performing a query

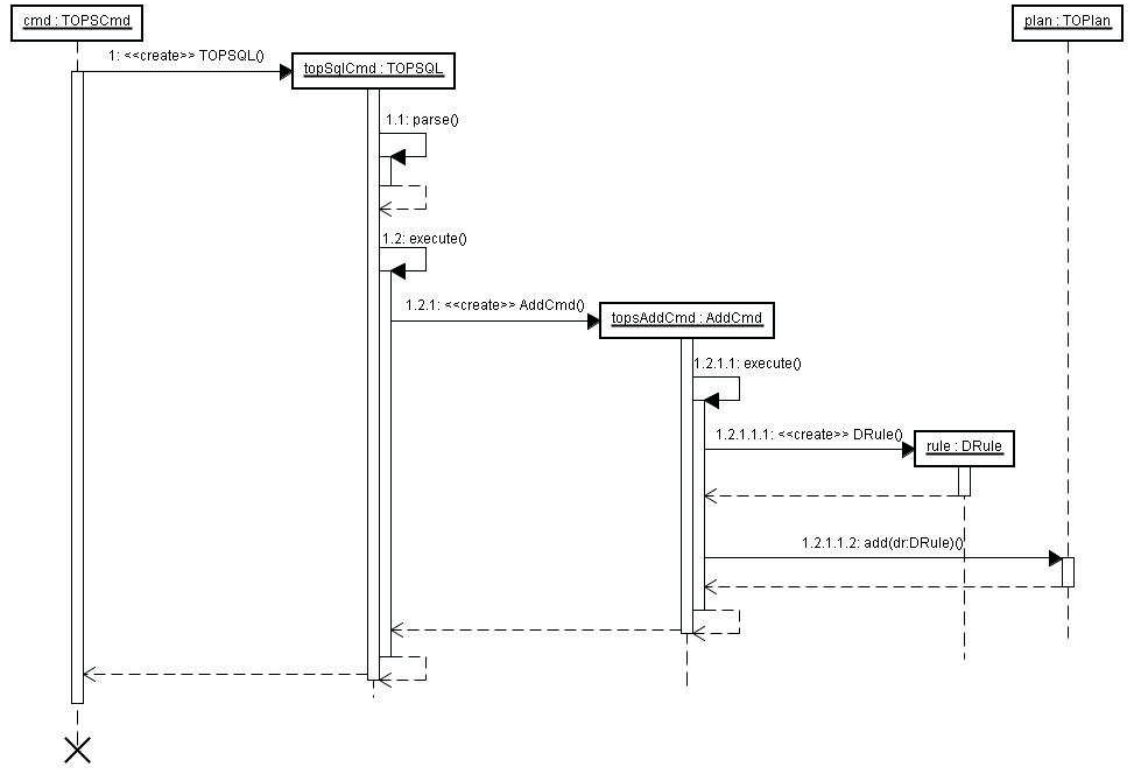


Figure 58 A sequence diagram for performing an operation on TOPS patient plan

in TOPS. On receiving the statement for the operation, the command line facility instantiates the manipulation language processor, the TOPSQL class, which parses the statement and the executes it by invoking a specialised statement handler such as the AddCmd class, which performs the ADD operation on the relevant TOPS object, such as adding a new rule to an existing plan.

9.4.4. The TOPS Database

The database system plays a central role in TOPS: first, it serves the purpose of storing the protocol specification; second, it holds the local patient record; third, it holds clinical information that is not patient-specific, such as orderable tests; and fourth it serves as the protocol execution engine through the ECA rule mechanism in the DBMS. This Section describes the design of the relational database used by TOPS.

The Protocol Specification Database

A protocol specification is initially created as a plain text file from an ordinary text editor. After being parsed using the mechanism described in Section 9.4 the specification is saved into the TOPS database, which is a relational database. This Section describes the protocol specification portion of the TOPS database. The extended entity-relationship diagram

illustrated in Figure 59 forms the basis for the relational schema for the protocol specification database. Boxes in Figure 59 represent *entities* while ellipses represent entity *attributes* with underlining of the attribute implying key attributes. For instance, PR_PROTOCOL is an entity whose attributes are id, name, date-created and date-authorised. The key attribute, id, is underlined. *Cardinality constraints* are represented using line sources and ends with multiple line sources and ends implying cardinality of greater than one while single line sources and ends imply unity cardinality. For example, an instance of the entity PR_PROTOCOL is associated with more than one instance of the entity PR_RULE, while each instance of the PR_RULE entity is associated with only one instance of the PR_PROTOCOL entity. The *is-part-of* relationship is presented by a line terminating with a diamond shape. For instance, each instance of the entity PR_ACTION is part of one or more instances of the PR_RULE entity. The *is-a* or *specialisation-generalisation hierarchy* is represented by a line with an arrow at the generalisation entity. For instance, each instance of the PR_CRITERIA entity is a specialisation of an instance of the PR_CONDITION entity.

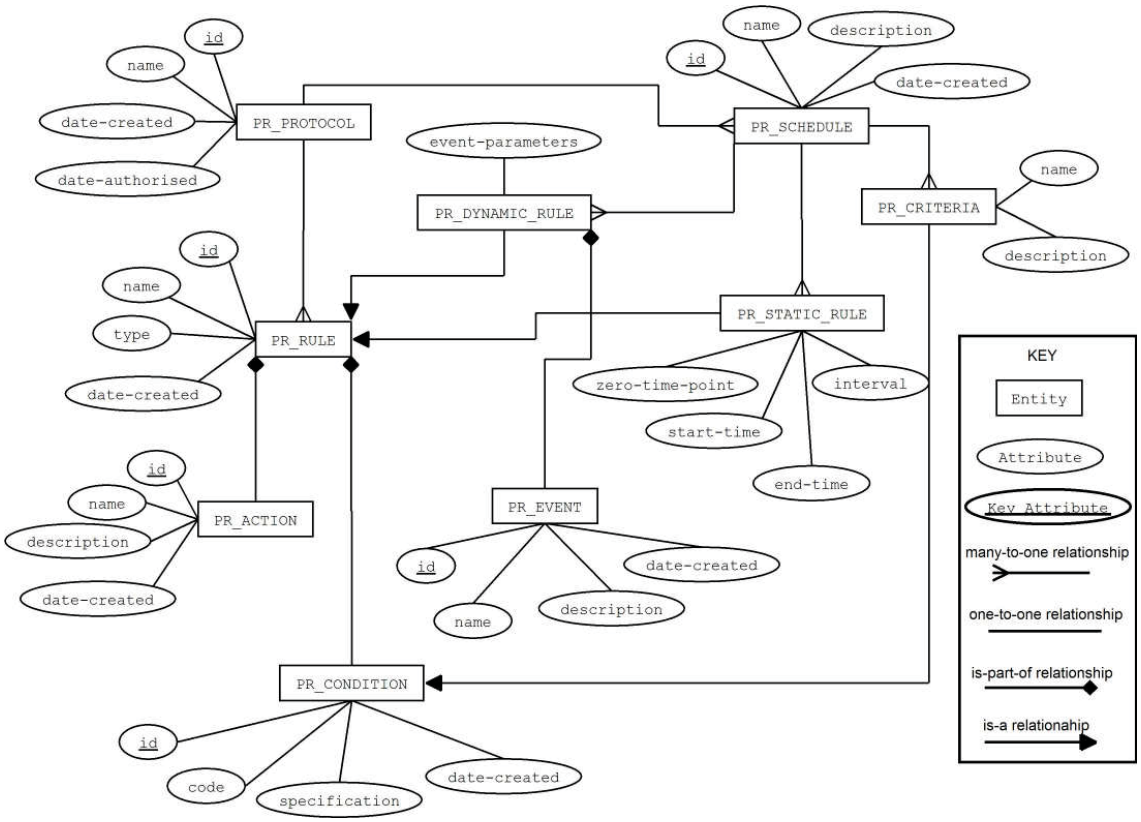


Figure 59 Entity-relationship diagram for the protocol specification in TOPS

The semantic model of Figure 59 is mapped into a normalised relational schema by using a mapping described by Ullman et al (Ullman and Widom 2001). The relational schema in presented in Appendix C.

The Patient Plan Database

This Section presents the database schema for the TOPS plan specification and execution database. Figure 60 illustrates the extended entity-relationship diagram for the TOPS plan specification. The notation used in the diagram is the same as that used in Figure 59. It can be seen that the EER diagram for a plan has less entities than that for the protocol. Firstly, the TOPS protocol has a set of schedules from which one or more are selected and combined into one for inclusion in creating a TOPS plan. The TOPS plan contains only one schedule holding static rules only. Secondly, a TOPS plan consists of rules that are implemented by using database triggers and external time triggers such that part of the TOPS plan's specification is contained in the database system's catalogue. The TOPS plan specification database serves the purpose of augmenting the database system's catalogue. The plan specification and execution database for a patient plan in TOPS consists of the TOPS plan database, the DBMS catalogue and the TOPS execution logs. The semantic model of the entity-relationship model of Figure 60 is mapped into a normalised relational database schema, which is presented in Appendix C. It should be pointed out that a TOPS plan belongs to a patient and is derived from a protocol created for the category to which a patient has been assigned.

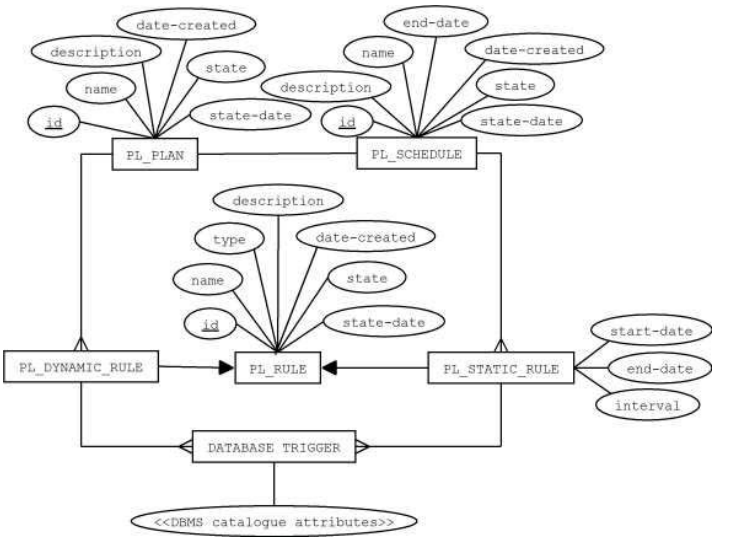


Figure 60 Entity-relationship diagram for the plan specification in TOPS

Although the entities for the TOPS patient and the TOPS protocol specification are not illustrated in Figure 60, there is a relationship between the TOPS patient and plan entities as well as between the TOPS plan and protocol entities. This explains the presence of the PATIENT_ID and the PROTOCOL_ID attributes in the TOPS plan relational table. The DATABASE_TRIGGER entity in Figure 60 is mapped to the USER_TRIGGERS table which is part of the database system. The relationships between the two types of rule entities, PL_DYNAMIC_RULE and PL_STATIC_RULE, on one hand, with the DATABASE_TRIGGER, on the other hand, are captured through the relational table. A single plan rule instance can be implemented by one or more triggers but, each trigger instance is part of the implementation of only one plan rule. This constraint is attained by having the attribute TRIGGER_NAME to constitute the primary key, thus, requiring the TRIGGER_NAME attribute to be unique.

The Execution Log Database

The execution of a TOPS plan proceeds through the execution of the rules that make up the plan. In order for TOPS to be able to allow the monitoring and manipulation of executing plans, there is a need for TOPS to maintain a number of execution logs.

Plan execution logs: At the *plan level*, they are two things that need to be monitored: the activity of the plan and the change of state of the plan over time. The overall plan activity in TOPS is maintained the system activity log, which uses the table PL_ACTIVITY_LOG. The TOPS plan activity is entered in the plan activity log, PL_PLAN_ACTIVITY_LOG. The change in the state of the plan over time is maintained in the plan state log, PL_PLAN_STATE_LOG.

Schedule execution logs: At the *schedule level,* only the schedule state is maintained. A schedule, in a TOPS plan, consists of a set of static rules, which monitor occurrences of time points and intervals. A schedule is active when any of its rules are active and finished when all rules have finished executing. The table PL_SCHEDULE_STATE_LOG is used to maintain the changes in the states of a TOPS schedule.

Rule execution logs: At the rule level, there a need to maintain changes in rule state and rule activity over time. The rule activity in the plan is maintained in the rule activity log, named PL_RULE_ACTIVITY_LOG. The change in the state of a rule is maintained in the rule state log, named PL_RULE_STATE_LOG. Since static rules monitor time events, a time event log, named PL_TIME_EVENT_LOG, is maintained.

The Patient Record

The patient record is a complex, distributed and heterogeneous medical information that spans the entire life time of a patient (Grimson, W, Berry et al. 1998). A single application captures only a portion of the entire patient record. TOPS maintains only a small part of the patient record and uses it to perform its functions. The patient record in TOPS consists of three parts: patient and clinician demographics, clinical laboratory investigations and advice (e.g., relating to diagnosis and medication). Figure 61 illustrates an entity-relationship diagram of the local patient record that it is used in TOPS. The attributes of entities are not presented in Figure 61 to avoid cluttering the diagram.

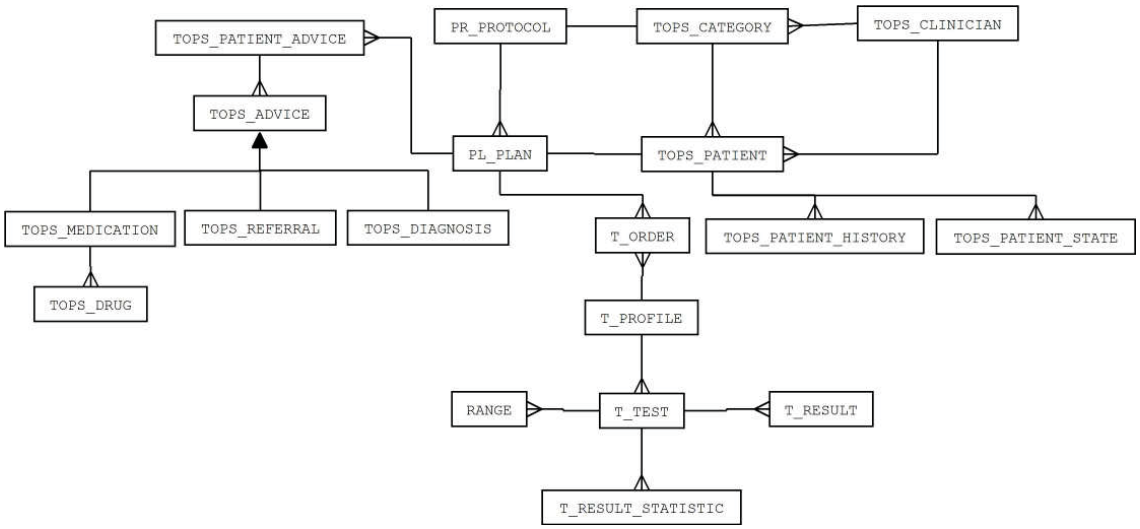


Figure 61 Entity-relationship diagram for the TOPS patient record

In TOPS, a clinician instance is associated with several clinical category instances and may take care of several patient instances. Each patient instance has at any one time only one instance of a TOPS plan. A TOPS patient instance may have several history and state instances. Both the TOPS plan and patient instances may be associated with several test order instances, each of which may specify one or more test profile instances. A test profile instance is a set of test instances, each of which must have ranges of values that represent a

normal patient condition. The normal range of values for a test may differ with patient age or sex – one test may have more that one normal range depending on the patient's attribute (sex or age). A test instance may have several result instances and several statistics may be monitored for it. Part of the TOPS plan actions may involve giving suggestions and advice relating to the patient associated with the plan. Three types of advice may be given and these are: medication (drug dosage), specialist referral and diagnosis-related advice.

The semantic model of the entity-relationship is mapped onto the normalised relational database schema, which is presented in Appendix B The entities PR_PROTOCOL and PL_PLAN have already been presented in Figure 59 and Figure 60 respectively. The resulting database schema includes database storage objects for clinical categories, patient demographics and clinicians as well as tables for clinical laboratory investigations and the advice available and given to a patient.

Views for the TOPS Database

To support a variety of queries that are expressed in the high-level language, TOPSQL, a number of SQL views a provided. The aim of the views is to enable the easy implementation and execution of queries expressed in TOPSQL. Appendix B presents a list of the SQL views that are defined in the TOPS database. The TOPS views are defined over the database schemes that have presented in previous sections.

9.5. TOPS's Support for the SpEM Framework and the MonCoos Approach

This section presents the design of the TOPS components for supporting each of the three planes within the SpEM framework. The architecture for the TOPS protocol specification mechanism is presented. This section also describes the protocol execution mechanism, which allows patient plans to be executed. Finally, the mechanism for manipulating the information and knowledge for supporting computerised protocols is presented.

9.5.1. The TOPS Specification Mechanism

The architecture for creating a TOPS protocol specification is illustrated in Figure 62. The process of creating the protocol specification involves editing, parsing and storing the specification in the database. The important components to support this task are the editor,

the parser and the specification database. The editing process creates the text file-based protocol specification in PLAN language.

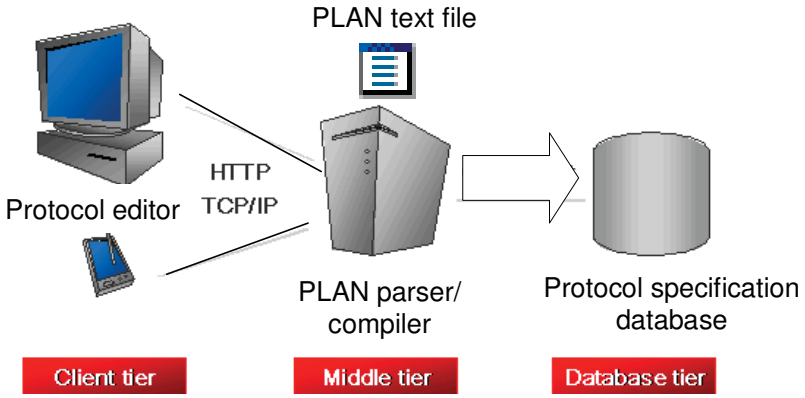


Figure 62 Creating the protocol specification in TOPS

The parser parses the PLAN specification and instantiates the protocol specification class. The resulting protocol specification object insert the protocol specification attributes into the corresponding relations of the specification database. The abstract form of the process for creating a protocol specification can be visualised as illustrated in Figure 63. A protocol specification is expressed in PLAN and takes the format of a plain text file. The protocol specification is translated into an object-oriented instance of the protocol specification, the protocol specification object. This protocol specification object maps the protocol specification object into tables in the relational database.

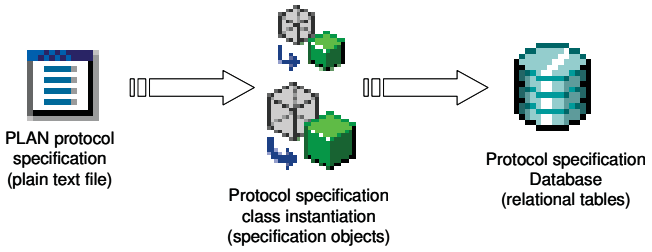


Figure 63 The abstract process for creating the protocol specification in TOPS

In TOPS, the PLAN protocol specification is initially created and stored as a text file. The protocol specification parser scans the PLAN specification text file and extracts the attributes of a protocol specification, which it uses to create objects for ECA components, rules and

schedules. These objects are used to instantiate the protocol specification class. In other words, the parser output is an object instance of the Protocol class. Figure 63 illustrates the abstract process for creating a protocol specification in TOPS. The parser for PLAN language protocol specifications has been developed and implemented. The object-oriented model of the TOPS specification parser is illustrated in Figure 64.

The Protocol class provides the mechanism for manipulating the protocol specification including adding the protocol specification to the database.

In Figure 64, classes whose names are in upper-case are parsers for the protocol component bearing the same name. For example, the PROTOCOL class is the parser for the protocol specification and creates a new instance of the Protocol class, while the SCHEDULE class is a parser for the protocol schedule and creates instances of the PSRuleSet and PDRuleSet classes, which are then used to create an instance of the PSchedule class. All the parsers in Figure 64 are specialisations of the Parser class. All the parsers follow the recursive descent parsing strategy (Aho and Ullman 1973).

The protocol specification object is used to view a text or graphical version of the specification, to instantiate a patient plan, and to store the protocol specification in the database. When the protocol specification is retrieved from the database, it is also held and manipulated in the form of the specification object.

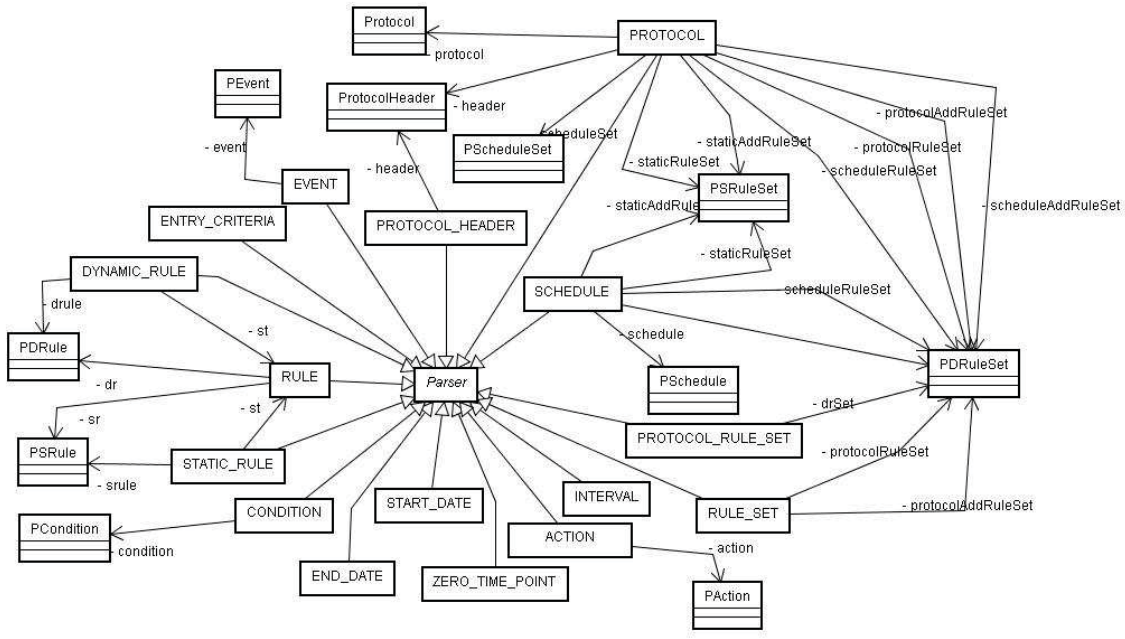


Figure 64 Class diagram for the PLAN language parser

9.5.2. The TOPS Execution Mechanism

This section describes the design and implementation of the ECA rule execution mechanism for TOPS. The section presents the design of the execution mechanism for time-driven static rules and the mechanism for executing the plan schedule in TOPS. The section also presents the design of ECA rule mechanism which services as the execution mechanism for TOPS. The aim of the design of the TOPS execution mechanism is to be generic enough to be applicable to any application scenario that could benefit from the ECA rule paradigm and the underlying database system.

A TOPS plan is composed of a time-driven schedule containing static rules and a set of dynamic rules. The execution mechanism of a TOPS plan is therefore made up of the execution mechanisms of the static and dynamic rule sets. This section presents the plan execution architecture and then describes the design of the execution mechanisms for the static and dynamic rules in TOPS.

The Plan Execution and Management Mechanism

Figure 65 illustrates the TOPS plan execution and management mechanism. The TOPS Plan Manager sets up, activates and permits a TOPS plan to be managed during its execution. The TOPS generic ECA Rule Mechanism extends the database trigger mechanism with time-driven rules and dynamic management functionality that is not supported by the database system database triggers. The TOPS Dynamic SQL Module dynamically builds the required SQL statements, submits the SQL statements the database system via JDBC, and receives results of queries from the database for onward transmission to the other components. The TOPS database contains specifications, execution state data and test orders and results part of the patient record. Test results are pushed to TOPS by the clinical laboratory or a laboratory simulator designed for the purpose of testing TOPS.

The resulting storage of the laboratory test result is eventually detected as an event of interest that triggers some patient plan rules. Certain high-level events can originate externally, for instance, from the clinician during an encounter with a patient. The next section describes the implementation of the TOPS database and its access component.

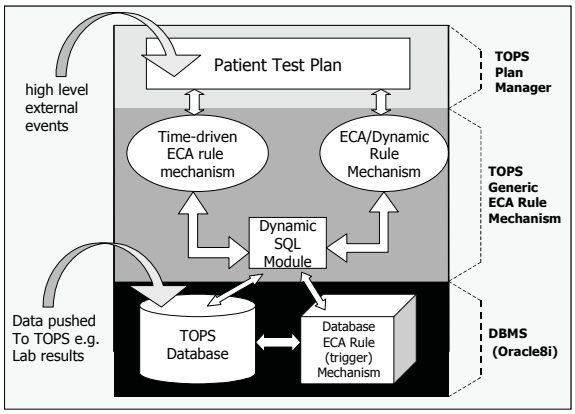


Figure 65 The TOPS plan execution and management mechanism

The core component of TOPS is the generic execution mechanism, which consists of the generic ECA rule mechanism and the database access component that handles connections and access to the database. The generic ECA rule mechanism accesses the database via the TOPS Database Access component.

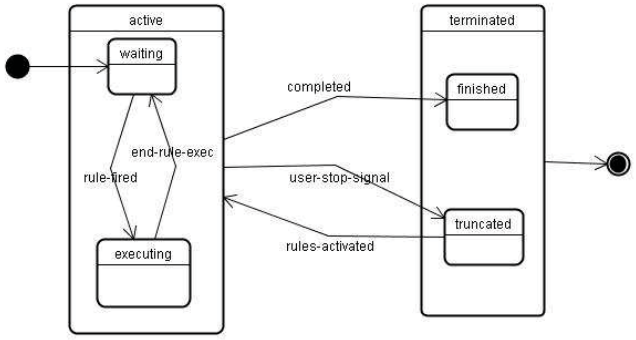


Figure 66 A state diagram for the patient plan

Plan Execution States

A TOPS plan goes through state transitions during its execution. These states and transitions of a TOPS plan are predefined and context-independent, that is, are independent of the plan's logic, content or protocol from which the plan is derived. Figure 66 presents the state chart diagram for a TOPS plan.

When a TOPS plan is created, it is automatically installed and activated. Its state changes from the initial state to the *waiting* state, which is a sub-state of the *active* state. In the *waiting* state, all rules in a plan are active and can react to any event that is of interest to

the plan. When a new event of interest is detected by any rule in the plan, the plan changes state from *waiting* to *executing* , a sub-state of the *active* state, and the rule is executed. When rule execution completes, the plan returns to the *waiting* state. When all rules have completed executing or their expiry period has passed, the plan changes state to the *finished*, a sub-state of the *terminated* state. This can happen at any point when the plan is in the *active* state. When a user stops an active plan, the plan changes state to the *truncated* state, which is also a sub-state of the *terminated* state. A plan that it is in the *truncated* state can be re-activated.

Patient Execution States

Figure 67 presents a context-independent state chart for a TOPS patient. A TOPS patient who is subject to a TOPS plan experiences state transitions that are of two types: context-independent predefined state transitions; and context-dependent and protocol-specific state transitions, which all occur as sub-state transitions of the *on-protocol* state. The *on-protocol* state is one of the context-independent predefined states in Figure 67.

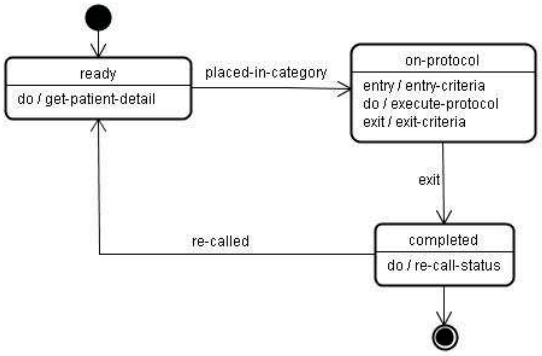


Figure 67 A high-level state diagram for a TOPS patient execution states

A protocol may define states and transitions as part of clinical logic. Such states as these are incorporated into the context-independent state chart of Figure 67 as sub states of the *on-protocol* state. A TOPS patient initially starts in the *ready* state. On being categorised, the patient changes state to the *on-protocol* state in which a plan is created from the relevant protocol and then executed. When in the *on-protocol* state, the patient may be subject to states and transitions that are specific to the plan or protocol until plan execution completes. On completion of the plan's execution, the patient state changes to the *completed* state, in

which the patient may be re-called into the *ready* state if there is a need to put the patient on another protocol.

The General Architecture for Rule Implementation and Execution Flow in TOPS

A TOPS plan consists of two sets of rules: the set of dynamic rules which are typical ECA rules; and the set of static rules, which are a special type of ECA rule that automate a timetable of clinical tasks that must be performed with respect to a given patient. These two types of rules in the TOPS plan are translated into one or more triggers. Figure 68 illustrates the implementation architecture and execution flow for the static and dynamic rules in TOPS. A dynamic rule is automatically translated into one or more triggers that are entirely in SQL and execute within the standard database management system (DBMS) trigger mechanism.

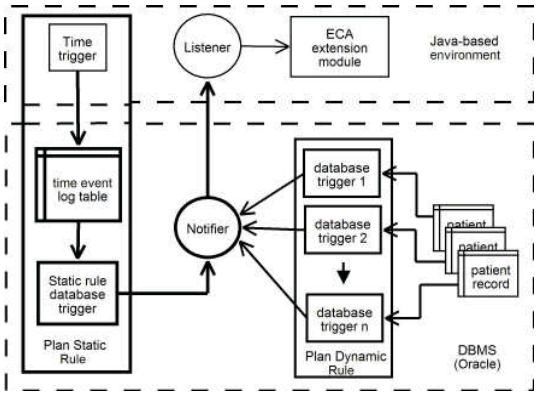


Figure 68 Rule implementation architecture and execution flow in TOPS

Dynamic rule database triggers monitor the local patient record. A static rule is automatically translated into a time trigger that is implemented outside the database system in a Java-based trigger mechanism, which monitors time events; and a database trigger implemented in SQL to realise the action part of the static rule. The time trigger signals time events through the time event log (a database table), which is being monitored by the static rule database trigger.

As illustrated in Figure 68, static and dynamic rule database triggers, when they execute, send messages containing execution information to the *notifier*, a Java-based module that is stored inside the DBMS. The *notifier* connects to and forwards the message to the *listener*, which is also a Java-based module residing outside the DBMS. The notifier invokes

the ECA rule extension module, which executes the rest of the rule's logic within the Java-based environment outside the DBMS.

Once an ECA rule is mapped or translated into database triggers, it is *added* to the database schema through the Dynamic SQL Module, which automatically builds the CREATE TRIGGER SQL statement and submits it to the database system for execution. Figure 69 illustrates the implementation of the ECA rule execution and manipulation mechanism in TOPS.

When an event of interest occurs, the database triggers representing the ECA rules are each fired, and executed in accordance with the rule execution model of the underlying database system. For instance, Oracle uses the execution model presented in Chapter 4 to maintain the proper firing sequence of multiple triggers and constraints checking. Examples of events in TOPS are data storage events associated with the creation of a new test order, the arrival of a new test result or the admission of a new patient.

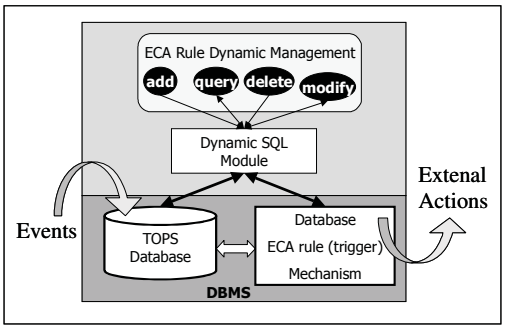


Figure 69 The rule execution and manipulation mechanism in TOPS

The actions of a TOPS rule currently include database events, external actions such as sending an alert or an e-mail message or displaying a message on the screen. As illustrated in Figure 69, the ECA rule dynamic manager provides operations that are to be performed on ECA rules in a dynamic fashion. These operations include add, query, delete and modify a rule. These four operations can be performed at any time in a dynamic fashion on any rule without affecting other rules.

Rule Execution States

Triggers in most database management systems are in one of two states, which are the *disabled* and the *enabled* states. Disabled triggers exist within the system but are prevented from monitoring and reacting to occurrences of events of interest.

TOPS plan rules are at a higher level than database triggers. States and transitions for TOPS plan rules need to be comprehensive enough to make it easier to provide information about the execution process and to perform manipulation operations without disrupting the plan's execution process.

Figure 70 presents the state chart for a rule in a plan in TOPS. A rule first goes into the *ready* state from the initial state. On the first occurrences of its events of interest, the rule is fired and enters the *executing* state in which its action is executed if its condition is satisfied. When execution completes rules state changes to be *waiting* state, where it stays until the next event is detected. From this point onwards, the rule's state changes to and fro between the *execution* and *waiting* state until it is retired, disabled or removed. The *ready, waiting* and *executing* states are sub-states of the *active* state.

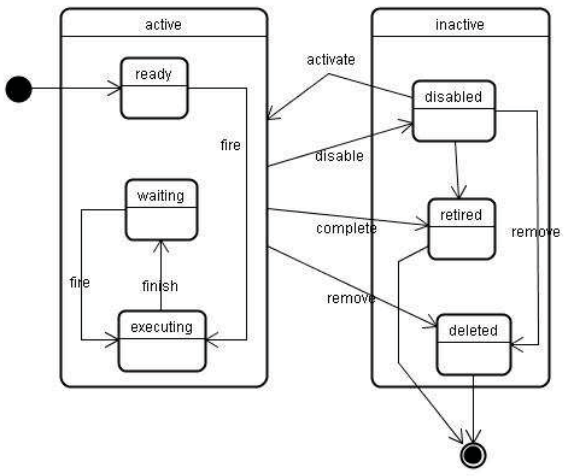


Figure 70 A state diagram for rule in a TOPS patient plan

When a rule in the *active* state is removed, the state changes to the *deleted* state. If it is disabled, the state changes to the *disabled* state. If the rule's active period expires, then its state changes to the *retired* state. If a disabled or retired rule is removed, its state changes to the *deleted* state, which marks its death. The *disabled, retired* and *deleted* states are sub-states of the *inactive* state.

Static Rule Execution Mechanism

A static rule executes a specified set of actions after every fixed interval of time, starting from a given time point and ending at a specified time point. Figure 71 illustrates the general design for executing a time-driven static rule.

In Figure 71, a static rule has a start date, which we denote d_s, and an end/expiry date, which we denote d_e. The static rule also has the time event interval such that the action is executed at time points e_1, e_2, … along the time axis, where $e_i - e_j = I$, for every $i = j+1$ and $e_i < d_e$. Thus the rule action is executed as long as the time point after the interval falls before the rule's expiry date, d_e. The rule's period of activity, p_a, is given by $p_a = d_e - d_s$. The number of times any given static rule will execute before its expiry date is give by p_a/(time event interval).

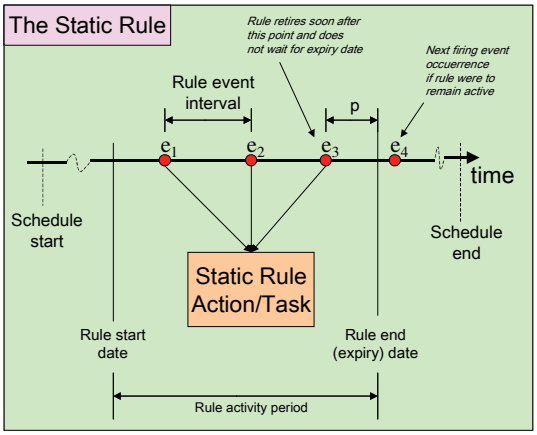


Figure 71 The execution mechanism for a time-driven static rule in TOPS

The period between the last time event, denoted e_l (in Figure 71, $e_l = e_3$), and d_e is denoted by $p = d_e - e_l$. The length of the time interval, p, depends on the size of the rule's time event interval and can be of arbitrary time unit including seconds, months or even years. We also note that $p \leq 0$ always. If $p = 0$, then rule's last event of interest coincides with its expiry date and so the rule should terminate immediately after executing its action. If $p < 0$, then the rule's last event of interest occurs before the expiry date (d_e) of the rule and the rule executes its action for the last time and then waits for period p, for the expiry date, d_e.

Since p can be of any size, allowing a rule to wait for its expiry date can lead to a situation where a number of rules are waiting for a long period (months or years) for nothing besides the occurrence of their expiry date. This unnecessarily prolongs the life of a rule. Since every rule knows its own expiry date and can calculate its next execution date, the rule

can determine its last time event and can, therefore, decide to retire immediately after its last execution rather than waiting for period p to expiry.

The TOPS design of the execution mechanism of a static rule is based on a timer that evokes the rule's action repeatedly after a fixed interval of time. The rule has a start time and an end/expiry time. The rule's first time event may or may not coincide with the start time of the rule. If the two do not coincide, then a delay period must be specified. The default is that the rule's start time coincides with its first time event. On the occurrence of the time event of interest based on the rule's interval and last execution time point, the rule's action performs the relevant task and then determines whether or not the time event that invoked it is the last event before the rule's expiry date. If the time event is the last one, the action detaches itself from the timer and deactivates the rule instead of waiting for the expiry date. If the event is not the last event of interest before the rule expires, the rule "sleeps" only to "re-awaken" on the occurrence of the next time event of interest. A single rule shares the timer with other rules. A rule cannot terminate the timer to avoid one rule to forcibly deactivate other rules.

In a TOPS plan, a schedule is a set of time-driven ECA rules, which are grouped together into a single collection. All rules in a schedule contribute to one overall objective. Figure 72 illustrates the execution mechanism designed for the TOPS schedule. The schedule consists of a single timer with a start and an end/expiry time stamps, a schedule monitor in form of a single ECA rule, $R_{monitor}$, and a set of time-driven static rules. In Figure 72, R_i where i = 1, 2, …, (n-1), n, is a static rule, and e_j, where j = 1,2, …, is a time event of interest to one or more rules in a schedule. $R_{monitor}$ is a schedule monitor, which is an ECA rule that monitors rules in the schedule and the end/expiry date of the schedule. The monitor is defined in the

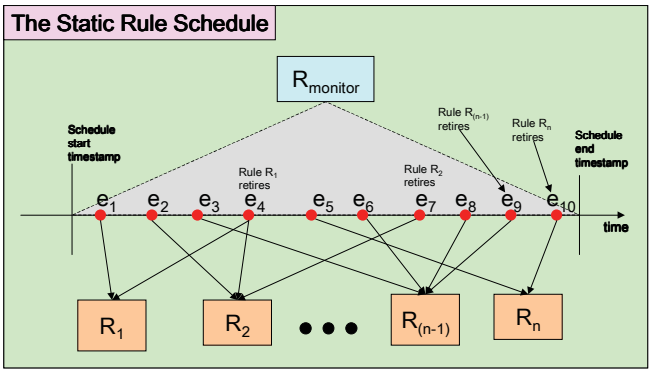


Figure 72 The execution mechanism for a schedule in TOPS

same way as static rules with an interval that is less or equal to the lowest interval in the rule set. The schedule's start time is the time stamp at which it is invoked. When a schedule starts, it first determines its own expiry date by examining the expiry dates of all the rules it holds. The schedule takes the expiry date of the rule with the latest expiry date. The schedule then activates all its static rules. Each rule attaches itself to the schedule's timer and uses it as an event source. The schedule monitor, $R_{monitor}$, also attaches itself to the schedule's timer. Each rule then proceeds independently as described in the first part of this sub-section.

The schedule monitor ($R_{monitor}$) executes to scan all rules in the schedule checking their execution status. When the schedule monitor discovers that all rules in a schedule have finished executing and are inactive or terminated, it then terminates the timer and deactivates the schedule. When all static rules are terminated, the timer continues to execute but does not generate any time event since all rules are inactive and there is no time event that is of interest to any rule. All rules in a schedule can terminate before the schedule's end date. In this case, the schedule monitor terminates the schedule and does not wait for the schedule's expiry date.

Rules in an executing schedule can be dynamically manipulated. The rules can be added, deleted, or modified dynamically without affecting the execution of other rules in the schedule.

TOPS's Handling of the Challenges from the Lack of Comprehensive DBMS Support for ECA rules

TOPS' implementation of the SpEM framework and MonCooS approach with the specification model, PLAN, and its language, PLAN, that uses a modern DBMS poses a number of challenges due to the lack of comprehensive and flexible support for the ECA rule paradigm. The following summarises how TOPS handles the limitations of ECA rule support in the underlying DBMS.

The mutating table problem: To protect a trigger from seeing an inconsistent data set, Oracle prevents a trigger from accessing the table that is being altered by the triggering transaction. Although Russell (Russell 2002) provides a solution for by-passing this problem using a temporary table and two triggers, TOPS solves this problem by separating the *ECA rule action* (in the protocol) from the *trigger action* and, therefore, also from the triggering transaction. The trigger action simply passes a message to an external action processor so that when the ECA rule action eventually executes the triggering transaction will have committed

and the table being altered is no longer mutating and will, therefore, be accessible. In other words, TOPS uses the deferred coupling mode for action execution which does not experience Oracle's mutating table problem.

Trigger restriction to monitoring one table: To monitor more that one table, TOPS uses a combination of the deferred execution mode with an event queue so that each trigger monitors one table and sends event messages to the event queue, which will be monitored for events on several tables.

Support for a domain expert (clinician) to make a decision before a rule's action is executed: TOPS avoids the immediate coupling mode in preference to the detached coupling mode since, in addition to avoiding the mutating table problem, it also allows a clinician to make a decision before taking any action. This is achieved by making trigger actions execute immediately while restricting their actions to passing a message to a detached action execution mechanism, which can prompt a clinician, possibly in asynchronous mode.

Fixed trigger execution order under lower priority with respect to integrity constraints: Two problems that may arise due to this limitation are: a) protocol execution may be interfered with if an integrity constraint is violated and triggering transaction associated with a protocol rule is rolled back; and b) if a trigger associated with a protocol rule fails to execute, this may cause the rollback of a legitimate and important transaction, e.g., a vital update to a patient record. TOPS cannot avoid experiencing the first problem since it has no control over integrity constraints. However, TOPS avoids the second problem during CGP execution by restricting the effects of trigger actions to *message passing* involving an external rule listener. This means that triggers that implement CGP rules are guaranteed to execute successfully all the time since: 1) they cannot violate any integrity constraint because they do not affect database state; and 2) in TOPS, the trigger action that performs the message passing is guaranteed to succeed all the time, even if the receiver of the message is unavailable.

Lack of trigger communication functionality with external environment: In TOPS, triggers that implement protocol rules communicate with an external Java environment by using HTTP sockets (see Appendix J). This communication is currently unidirectional from the trigger to the external rule listener. As a result, there is no way a trigger can gain control of external actions or receive feedback from the execution of external actions.

Summary

This section has presented the design of the TOPS rule execution mechanism. ECA rule or protocol rule execution in TOPS is mainly based on the underlying database trigger execution mechanism. TOPS' contribution is in the following aspects:

1) The provision of the functionality to allow the dynamic management of the rules;
2) The mapping of high-level logical ECA rules to database triggers; and
3) The provision of a high-level event service to extend the limited set of possible events provided by current DBMS and generally extending the database trigger mechanism to support those aspects of ECA rules that are not adequately supported.

9.5.3. The TOPS Manipulation Mechanism

This Section presents the design of the manipulation mechanism in TOPS. The manipulation mechanism in TOPS allows specifications to be maintained and patient plans to be managed by using the **TOPS Q**uery **L**anguage, TOPSQL, while they are in the process of execution.

General Strategy for the Implementation of TOPSQL

In supporting the management of clinical protocols, use is made of the relational database model and its mechanism for supporting ECA rules as the core operating environment. The implementation strategy for the manipulation language, TOPSQL, is to define the language to be at a level higher than the SQL such that it can be implemented using the SQL at a lower level. Figure 73 presents the implementation strategy adopted for TOPSQL.

The strategy is to implement TOPSQL through an object-oriented environment that maps easily or can easily access, through the use of SQL, the relational model-based protocol system database.

TOPSQL queries can be supported by a set of TOPSQL object classes that access a rich set of the protocol system's logs, views and protocol and plan specification tables using the SQL. Queries that involve the replay of plans are implemented through re-play simulator classes which have access to execution logs and views by using the SQL. The implementation strategy for TOPSQL illustrated in Figure 73 has the advantage that it guarantees portability and ease-of-maintenance through the use of the object-oriented paradigm and SQL.

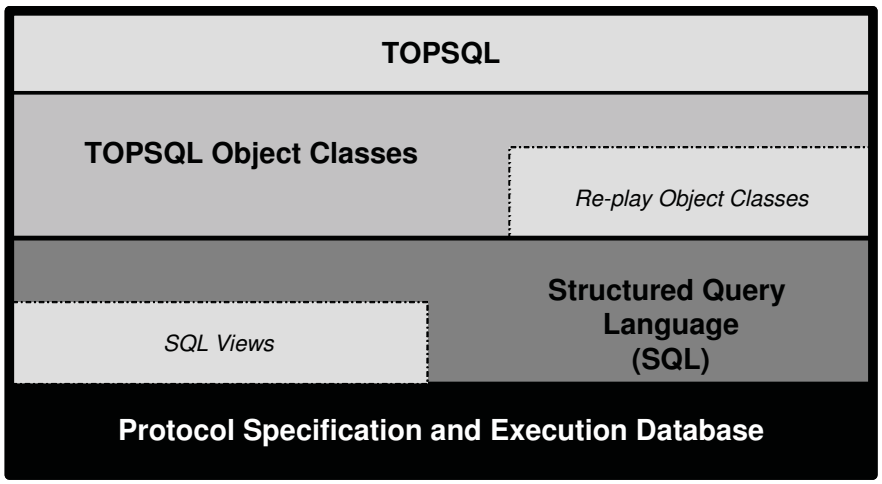


Figure 73 TOPSQL implementation strategy

The TOPS Architecture for the Implementation of TOPSQL

The conceptual architecture for the TOPS manipulation mechanism, which implements TOPSQL, is illustrated in Figure 74. The *TOPS client* provides the interface for the user to specify either the query or the operation he/she desire to be performed on either specifications or patient plans.

The *manipulation manager* interfaces with the TOPS clients and determines whether the user's request is for a query or an operation on TOPS objects and invokes the appropriate handler. The *query parser* parses the query statement and

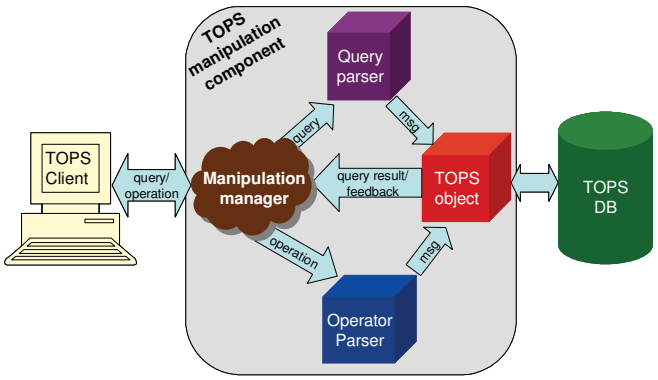


Figure 74 The TOPS manipulation mechanism

analyses the requirements of the query. The results of the analysis are passed as parameters for the message sent to the TOPS object, which should be the subject of the query.

The *operator parser* parses and analyses the statement representing the operation required to be performed. The results of analysis are passed on to the appropriate TOPS object. The query and operator parsers together implement TOPSQL, the manipulation language described in Chapter 8. The subject of a query or an operation requested by the user is one of the objects in TOPS, the *TOPS object*.

The TOPS object can be a protocol, plan, patient, category, or rule. The TOPS database holds information about protocol specifications and the patient plan execution process. All the information that is the subject of a query or an operation is held in the TOPS database. Each TOPS object performs the user query or operation by accessing the TOPS database.

The object model of the TOPS manipulation mechanism

The manipulation of protocols, plans and patients, which involves issuing queries and performing operations on the objects, has been implemented in the TOPS manipulation mechanism whose high-level object model is illustrated in Figure 75.

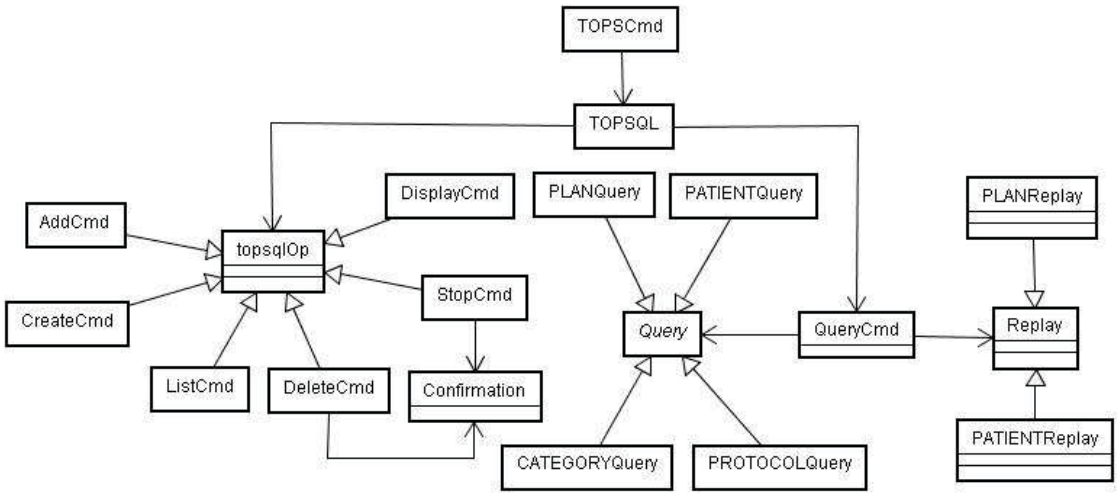


Figure 75 The class diagram for the TOPS manipulation mechanism

The TOPSCmd class provides a command line interface to the user and accepts commands in the form of TOPSQL statements and passes them on to the TOPSQL class. The TOPSQL class is a parser for TOPSQL statements, which invokes either the TopsqlOp class, for manipulation operations or the QueryCmd, for queries, for further processing. The

specialisations for the manipulation operations class include the AddCmd class - for adding objects, the CreateCmd – creating objects, ListCmd class - for listing names of objects, e.g., listing protocol names, DisplayCmd class for displaying detailed specification for an object, DeleteCmd class – for deleting objects and StopCmd class - for stopping the execution of a plan.

QueryCmd class has to distinguish between a query and a request to replay events in the system. The specialisation for the Query class includes the PLANQuery class – for handling queries relating to a plan, the PATIENTQuery class – for handling queries relating to patients, the CATEGORYQuery class - for handling queries relating categories, and the PROTOCOLQuery class – for handling queries relating to a protocol. The specialisations for the Replay class are the PLANReplay class – for allowing the plan's execution to be replayed, and the PATIENTReplay class – for allowing the events happening to a patient to be replayed.

9.6. The Architecture of TOPS

This section presents the architecture for TOPS. As illustrated Figure 76, the architecture has three layers. *External* to the system are users and external systems. The *top* layer is the clinical protocol management functionality that allows users to specify, store, execute manipulate and query clinical protocols and external systems to supply and receive information from the system. The *middle* layer provides services that 1) extend the ECA rule execution mechanism of the underlying database system and 2) handle connections to the database. The *bottom* layer is the ECA rule execution mechanism, which is the *ECA rule mechanism* in a modern database system.

TOPS Clients and External Systems: *Users* of the system, who may be either clinicians or patients, use the TOPS clients. Typically, users should be subject to security checks and authorization. Currently, basic security is provided through user-names and passwords. Besides users, the system can interacts with *external systems* such as the clinical laboratory information system for test order submission and result receipts. Other systems may want to access information relating to protocol specifications and execution process of TOPS plans.

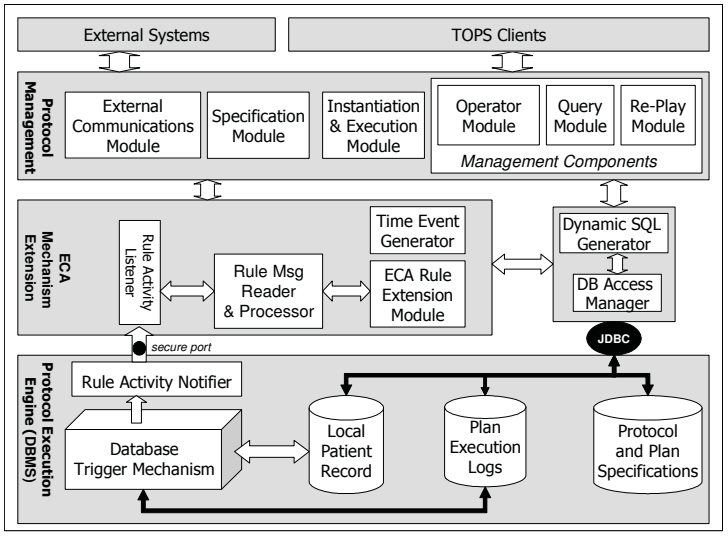


Figure 76 The Architecture of TOPS

The Protocol Management Layer: This generally provides users with the functionality for managing patients and patient categories, creation of protocol specifications for patient categories, creating, executing and manipulating patient plans, and querying the system's static and dynamic information.

The ECA Rule Mechanism Extension: The ECA Rule Mechanism Extension's main purpose is to provide the functionality that is not adequately supported by the ECA rule execution mechanism in the underlying database system and to perform actions that need to be performed outside the database system. The *Time Events Generator* extends the database trigger mechanism by providing a time event detector. It generates time events of interest to specific rules within each patient plan.

The mechanism for supporting time triggers is illustrated in Figure 71. The Java-based time trigger mechanism is used to give signals for the occurrence of only those time events that are of interest to rules in the patient plans. The Time Events Generator was necessitated by the absence of the support for temporal triggers in modern database systems, including the Oracle DBMS used in TOPS.

The *Rule Activity Listener* listens and receives messages from rules executing within the database system. Modern database systems do not provide support for rules to communicate externally with applications outside the database. For instance, current *database connectivity through* the Java Database Connectivity (JDBC) and the Open Database Connectivity (ODBC) do not support active behaviour or *push* functionality (only *pull* functionality is supported). Hence, there is a need for a separate mechanism to allow

rules to communicate externally. With the Rule Activity Listener, rules inside the database can communicate with other modules of the system that are outside the database.

Lastly, the *Dynamic SQL Statement Generator* and the *Database Access Manager* are the two components that are dedicated to handling standard communication through database connectivity between the database system and external components. The Dynamic SQL Statement Generator generates the required SQL statements to allow dynamic manipulation of both rules and data in the database.

The trigger mechanism of a database system is used as the engine for executing the ECA rule-based clinical protocols. One ECA rule in a patient plan is mapped to one or more database triggers. The mapping is predefined for each of the two main types of rules, i.e., the static rules and dynamic rules. A static (time-driven) rule is mapped to one Java-based *time trigger* that signals the occurrence of a time event and one *database trigger* that reacts to this signal. Dynamic rules are mapped to only one database trigger. Each ECA rule in a plan monitors either the patient's record or the plan's execution logs such as the time event log.

9.7. Discussion and Related Work

A Relational Database Model-Based Knowledge Representation Scheme for CGPs: Lobach *et. al.* (1997) represented guideline content and logic using a hybrid of structured and procedural knowledge representation formalism. Advantages of the relational database format for storing CGP knowledge has been identified to include: compatibility with Internet applications and technologies for information exchange such as XML; popular model that has evolved into an industry standard; supported by many DBMS tools; sharability through applications using SQL; easily explained using a tabular representation (Lobach, Gadd et al. 1997). An advantage the approach taken by TOPS is the portability afforded by both the object-oriented environment and the relational model and SQL. Another advantage is that both the object-based mapping and the relational database are compatible with XML, which gives TOPS the advantage of future adaptability into a distributed client-server framework. It has already been demonstrated that TOPS can act as a clinical protocol management server for distributed clients (Jones, Dube et al. 2003).

Concept or Phenomenon Equivalent to Patient Plan Rule Set in TOPS: In TOPS, each patient has his or her own rule set making the patient's plan. The same or similar idea is found in Appelrath et al's active repository that uses an active DB for implementing the

persistent and reactive parts of a process- centred software engineering environment (Appelrath, H-J, Behrends et al. 1995). There, they identified the need for rule sets on a project basis, requiring extensions to their toolbox regarding multi-user and meta-programming capabilities. The software project is equivalent to the patient object in TOPS. However, their system could not support this feature for customised rule sets as it lacked multi-user support, which was needed for supporting several user groups each having its own specific set of rules.

Rule Modification And Evolution: Geppert et al (1995) describe the implementation of rule-base evolution with respect to event type modification only. The detection of composite events was based on Petri Nets called the SAMOS Petri Net (S-PN). The S-PN also maintains the event history. They gave the algorithm for the event type modification based on the manipulation and reconstruction of the S-PN structure. The modification of a rule means changing the rule's event, condition or action. In modern commercial database systems such as the Oracle DBMS, deleting and then replacing the rule by a new rule can achieve rule modification. The rule's ECA components are not accessible as separate objects. The rule-base in these commercial systems cannot be queried at rule component level. In TOPS, rule evolution is a major aspect that needs to be supported to allow flexibility in changing the specification of a test-ordering plan. The rule-base in TOPS consists of test-ordering plans. Each test plan should be considered as a "stand-alone" rule-base that should be considered in isolation from other plans. The rules of a test plan must not interact with rules of another plan unless that plan belongs to the same patient. At the test plan level, there is still need for plan-level operations and queries.

Rule Monitoring Intervals:: Geppert et al (1995) also introduced the concept of a "monitoring interval" in the SAMOS active database prototype (Geppert, A. and Dittrich 1993). The monitoring interval is a time interval that can be specified (in terms of a start time and end time) to determine when an event has occurred in order to be considered as relevant. In TOPS, monitoring intervals could be applied to dynamic rules to prevent rules whose plans have expired from executing or to give the dynamic rules an expiry period. The dynamic rules would automatically deactivate or retire once the current date is beyond the end time and should not execute in reaction to events occurring before the start time. Thus, each dynamic rule would execute only during the specified "monitoring interval".

Correctness of TOPS Plans: In TOPS there is need to perform rule analysis (Bailey, JA 1997; Baralis, E., Ceri et al. 1998) in order to verify the correctness of a protocol and

patient plans. The generation of triggers need to be formalised in order to guarantee the correctness of the resulting patient plan. The current implementation of TOPS relies on the domain expert's analysis of the clinical protocol and the resulting sets of ECA rules in guaranteeing the correctness of protocol rules. Development of a formal method for analysing protocol rules and verifying their correctness has been left as part of future work.

TOPS Plan Manageability: To make clinical test-ordering plan manageable in TOPS, there is need to introduce rule stratification and modularisation (Baralis, Elena , Ceri et al. 1996) in a test plan. The division of a plan into a schedule, which is a set of static rules, and a set of dynamic rules forms the basis of rule stratification in a test plan. There is need for an explicit formal specification of the stratification criteria. Global rules could form a stratum that exists external to all test-ordering plans.

Message Transmission by an Trigger-Based TOPS Rule to One or More Applications External to DBMS: In TOPS, most of the time, rules do not automatically perform actions on behalf of the clinicians. Instead, the rules either prompt, recommend or alert. Hence all rules in TOPS need to transmit messages to one or more TOPS modules that are external to the DBMS. Hanson et al (1998) proposed an integrated, flexible framework for interaction between an active DBMS and applications. Possible problems that can occur when a rule signals events or sends messages to applications that were dealt with by Hanson et al (1998) are: 1) Lost-dependency operation problem (LDO):- signalled events or messages sent may be lost or not acted on by the receiving application (the client); and 2) Dirty dependency operation problem (DDO) when an application or a client is allowed to process an uncommitted event signal or message. In TOPS, message transmission from database triggers that implement protocol ECA rules to modules external to the DBMS or other systems is achieved via HTTP connections between trigger actions (the HTTP clients) and an external HTTP server process, which in turn links with the external applications. These HTTP connections are subject to security authentication. However this method of communication between triggers and external application still suffer from the LDO problem. To address the LDO problem, a feedback tracking system may need to be implemented in TOPS. The DDO problem my not be an issue in TOPS since the agent that acts on event signals or messages from trigger is responsible for committing the event signals or messages. The agent is the clinician who is allowed to choose not to act on TOPS messages.

Creating Specification Using a High Level Description Language: Eder et al use a graphical description language to specify business processes or flow (Eder, Groiss et al.

1994). They translate the resulting specifications into triggers of an active database. In TOPS the same approach is adopted . The only difference being that the language used in TOPS, PLAN (Wu, B. and Dube 2001), is not graphical. PLAN is a specification language that is higher than database triggers and has advantages of being independent from a specific product or trigger language. Specifications based on triggers are at a low level making such specifications more difficult to read and debug.

Li and Chakravarthy's ECA Agent: Li *et al* (1999) used a mediator to provide ECA functionality to Sybase, a relational DBMS. Their ECA agent is similar to TOPS and can also be considered as a wrapper to the underlying DBMS. Several aspects and features of Li and Chakravarthy's ECA agent (1999) bear some similarities to TOPS. The first aspect of similarity is the use of ECA rule parser to scan and parse ECA rules for syntax errors. The parser will create events and rules and generate the required SQL. The event and rule specifications are stored in relational tables. Events and rules are created from the specifications stored in these tables. A second similarity is that, in TOPS, a rule execution "notifier" sends a message to a rule activity "Listener" each time a rule's action is executed. The notifier is a java stored procedure that executes inside the DBS while the listener is an external java stored procedure that executes inside the DBS while the listener is external java routine. Furthermore the listener is an HTTP server while the notifier is an HTTP client. Li and Chakravarthy provide an "event notifier" which sends notifications of primitive event occurrences to a "local event detector" after receiving a signal from an executing DB trigger. A third similarity is that after the occurrence of an event, TOPS involves an "external action" processor which then launches the appropriate actions. Li and Chakravarthy use an "action handler" which calls the actions defined as an event that has occurred.

There are a number of differences between TOPS and Li and Chakravarthy's ECA agent. In TOPS, a comprehensive treatment of composite events has been left to become part of future work although they are of fundamental significance to the problem being handled by TOPS. Li et. al. use the SNOOP event specification language originally designed for Sentinel. They went even further to enhance the SQL trigger definition by incorporating the SNOOP event definition. Li and Chakravarthy (1999) deal mainly with ECA rules that are submitted individually to the system and they provide no mechanism to group rules together. In TOPS rules are grouped into sets that form complex objects – the protocol or plan. Furthermore, it is important to query both the rule specifications and their activity history. As a result it is necessary that TOPS provides a ECA rule query language for this purpose. Li

and Chakravarthy (1999) do not discuss the issue of querying the rule-base and rule activity history.

The Paradigms in TOPS: TOPS employs object-orientation and the ECA rule paradigm within the context of the relational database model. One of the important issues within these paradigms in TOPS is the synchronisation of objects across the paradigms. Porto et al (1999) have investigated persistent object synchronisation in *active relational databases.* They propose an architecture that is based on a replication strategy, which maintains server tuples and client-cached objects synchronised with respect to state. A combination of the object-oriented paradigm with the active relational model offers the problem of "impedance mismatch" between the object-oriented model and the relational model. This challenge points to the need to deal with structural and behavioural model clashes, which include: object attributes that are stored in different relational tables; object relationships, e.g., inheritance, that have no equivalent in the relational model; and state change in application objects are reflected in persistent versions of these objects and vice-versa. The main issues to be dealt with include: representing the object life cycle inside an active relational database system; and implementing object behaviour via database triggers and stored procedures. This book has not addressed these issues and problems associated with the use of different paradigms that need to interact across their boundaries

9.8. Summary

This chapter has described the design and implementation TOPS, the prototype system for managing clinical protocols for the domain of clinical test ordering by clinicians. The chapter proceeded to attain its aim and objectives by first describing the general and specific problem to which TOPS serves the purpose of a solution. The requirements, from the domain and technical perspectives, have been presented. The Chapter then presented the design of TOPS in terms of the functional, object and dynamic models before giving more detailed descriptions of the design of important aspects and components of the system which include: the protocol specification database; the three mechanisms for the specification, execution and manipulation of clinical protocols; the architecture of TOPS; and the implementation of TOPS. The design of TOPS described in this chapter addresses the requirements of the protocol management framework that has been introduced in Chapter 3 and 5; and implements the approach and method that has been described in Chapter 5 and explained in

detail in chapters 6-8. This Chapter also presented a review of the related work and discussed the implications to the design of TOPS. The next chapter demonstrates that TOPS, as described in this chapter, achieves its aims and objectives by presenting a demonstration and an evaluation of its functionality.

Chapter 10. Case Study: Supporting the Management of the Microalbuminuria Protocol for Patients with Diabetes Mellitus

10.1.Introduction

This chapter presents a case study that uses TOPS to manage a clinical protocol for the diagnosis and treatment of microalbuminuria (MA) in diabetes patients. The case study applies the SpEM framework and MonCooS approach in supporting the management of the *microalbuminuria protocol* (MAP). The MAP is modelled and specified in the specification language, PLAN, parsed and stored in the TOPS database, executed by the TOPS execution mechanism and both the MAP specification and executing instances are manipulated by using the language TOPSQL. The medical aspects of the work presented in this Chapter relied on the assistance of medical domain experts within the MediLink Programme (MediLink 2003), a multi-institutional inter-disciplinary research programme spanning the Dublin Institute of Technology, Trinity College and St. James's Hospital.

The rest of this chapter is organised as follows: Section 10.2 presents a brief clinical background to the microalbuminuria protocol and its significance; Section 10.3 presents a description of the microalbuminuria protocol; Section 10.4 demonstrates the method of capturing knowledge and specifying the MAP; Section 10.5 briefly describes the creation of the MAP specification database in TOPS; Section 10.6 discusses the execution of the MAP using TOPS; Section 10.7 discusses the manipulation aspect of the management of the MAP in TOPS; Section 10.8 presents a discussion that focuses on the strength and limitations of the protocol management framework presented in this Book; and, lastly, Section 10.9 summarises of this Chapter.

10.2.Clinical Background: Diabetes and Microalbuminuria

Diabetes is a chronic disease defined as "inappropriate glucose metabolism leading to impaired removal of glucose from the circulation" (Ristow 2004). The main characteristic of diabetes is a sustained elevated blood glucose level resulting from *insulin deficiency* or from *insulin resistance*. Insulin deficiency results from an insufficient secretion of insulin by pancreatic beta cells. With insulin deficiency, the body does not have enough insulin to metabolise blood glucose and reduce it to appropriate levels. Insulin resistance is the body's inability to properly use the insulin that it produces to reduce blood glucose level in the body to appropriate levels. Both insulin deficiency and insulin resistance lead to *hyperglycaemia* or high blood glucose levels. Diabetes is one of the major chronic diseases in developed countries where it is increasing and directly or indirectly through the effects of its many complications, accounts for approximately 10% of healthcare expenditure (Andreassen, Gomez et al. 2002). The disease is also on the increase in developing countries.

The clinical management of diabetes is of huge importance in minimising the incidence and effects of the disease's long-term complications (Andreassen, Gomez et al. 2002). The long-term complications of diabetes are mainly based on the disturbances of carbohydrate, protein and fat metabolism (American Diabetes Association 2002). Diabetic renal disease is one class of diabetic complications that result from the disturbance in protein metabolism in diabetes patients. It has been found out that one in three patients with diabetes will be affected by diabetic renal disease (Harvey, Rizvi et al. 2001). Microalbuminuria is a renal disease associated with kidney abnormalities and other organs of the body. The presence of microalbuminuria or proteinuria (nephropathy) increases the risk of large blood vessel disease and premature death. Early intervention can preserve renal function, preventing progression to end stage renal disease (Mogensen 2003). Late intervention may slow the rate of renal decline to dialysis. Interventions can also reduce other vascular morbidity and mortality (Harvey, Rizvi et al. 2001). The clinical aim of the microalbuminuria protocol (MAP) for diabetes mellitus patients is to minimise rapid progression into end-stage renal failure in diabetes patients through early intervention and management (American Diabetes Association 2002; Mogensen 2003). In this case study TOPS, aims at assisting in

achieving this aim by serving as a tool for scheduling, monitoring and coordinating clinical intervention using clinical laboratory tests.

10.3.Description of the Microalbuminuria Protocol (MAP)

Every year at annual review of diabetes patients, the patient's urine is screened for *protein loss*. The screening looks for microalbuminuria, proteinuria and raised serum creatinine. In those with renal changes or renal impairment, *urine albumin excretion* (UAE) should also be monitored every 6 months. The following interventions are necessary for people with renal changes:

Diabetes renal screening: Every patient will be provided with a universal specimen pot and asked to bring an early morning urine specimen (mid stream) to their annual review appointment. The urine is dipped for albumin in the dipstick urine test (DUT). If there is either no albumin or a trace of albumin on dip testing, the sample is sent to the biochemistry laboratory for an albumin-creatinine ratio (ACR) test to be performed. Table 10.1 presents the guideline's clinical interpretations of the results for the albumin-creatinine ratio (ACR).

Table 10.1 Interpretation of the albumin-creatinine ratio (ACR)

DIAGNOSIS	DESCRIPTION	RESULT RANGE
Normal	Negative or trace of albumin on dip testing and an albumin	ACR <3.0 (20 mg/l)
Significant	Negative or trace of albumin on dip testing and an albumin	ACR >3.0 (200 mg/l)
Microalbuminuria	Should only be diagnosed if there have been 2 positive results. Dip positive for protein in the absence of a urinary infection, confirmed by a 24 hour protein loss of > 200mg/l. If this is the first result, please repeat screening.	ACR >3.0 (200 mg/l) (within a 6 month period)

If the DUT results in positive proteinuria being identified, the specimen is sent to the microbiology laboratory for culture and sensitivity to exclude infection. If there is no infection, and this is the first time that proteinuria has been identified, a 24-hour urine collection is sent to the biochemistry laboratory to assess creatinine clearance and 24-hour protein loss.

Optimum glycaemic control: HbAlc < 7%. Hb stands for *haemoglobin,* the compound in the red blood cells that transports oxygen. When glucose in the blood sticks to haemoglobin A, gylcosylated haemoglobin or HbA1c or haemoglobin A1c is created. Haemoglobin occurs in several variants; the main variant is known as haemoglobin A. Thus,

A1c is a specific subtype of haemoglobin A. The *1* is a subscript to the A, and the *c* is a subscript to the *1*. Diabetes patients have a high amount of HbA1c because they have a higher level of blood glucose than non-diabetics.

Blood pressure control is undertaken aiming at attaining the targets for diabetes patients presented in Table 10.2. Angiotensin Converting Enzyme (ACE) inhibitors for blood pressure control is prescribed to maximum tolerated dose and the Serum-Creatinine Ratio (SCR) and Potassium are monitored only if the patient is not pregnant.

Table 10.2 Blood pressure targets for diabetes patients

PATIENT CATEGORY	BLOOD PRESSURE (mmHg)	
	Asystolic	*Diasystolic*
Everyone with diabetes	140	80
Diabetes, aged > 40 with renal changes	130	75
Diabetes, aged <40 with renal changes	120	70

10.4.Creating a PLAN Specification of the MAP

The method used in this Section models clinical protocols using the UML state chart as a tool to capture and enhance the domain knowledge in terms of the ECA rule paradigm. It has been noted, in the literature, that events, event parameters, conditions, actions and activities are already supported in UML state charts, where it is possible to support variants of ECA rules (Berndtsson, Mikael and Calestam 2001).

10.4.1. Modelling Knowledge in the MAP

The method of CGP knowledge modelling presented in Chapter 5 will be demonstrated using the protocol for the treatment and management of microalbuminuria (MA) in diabetes mellitus. Figure 77 illustrates the state chart for the microalbuminuria protocol (MAP).

The process of renal screening illustrated in Figure 77 starts with the annual screening of blood and leucocytes in urine using the dipstick urine test (DUT) as described in Section 10.3. If the DUT is positive, i.e., blood and leucocytes are present in urine, then screening for other infections is done before a patient can be referred to a nephrologist. If the DUT is negative, i.e., blood and leucocytes are absent from urine, then the patient is screened for microalbuminuria, which involves three measurements of urine albumin using the albumin-creatinine ratio (ACR) test over a period of six months. If ACR is less than 20 mg/l at any point, then the patient is cleared of microalbuminuria and becomes subject to the annual DUT. If ACR is greater than 200 mg/l, then the patient is referred to the nephrologists.

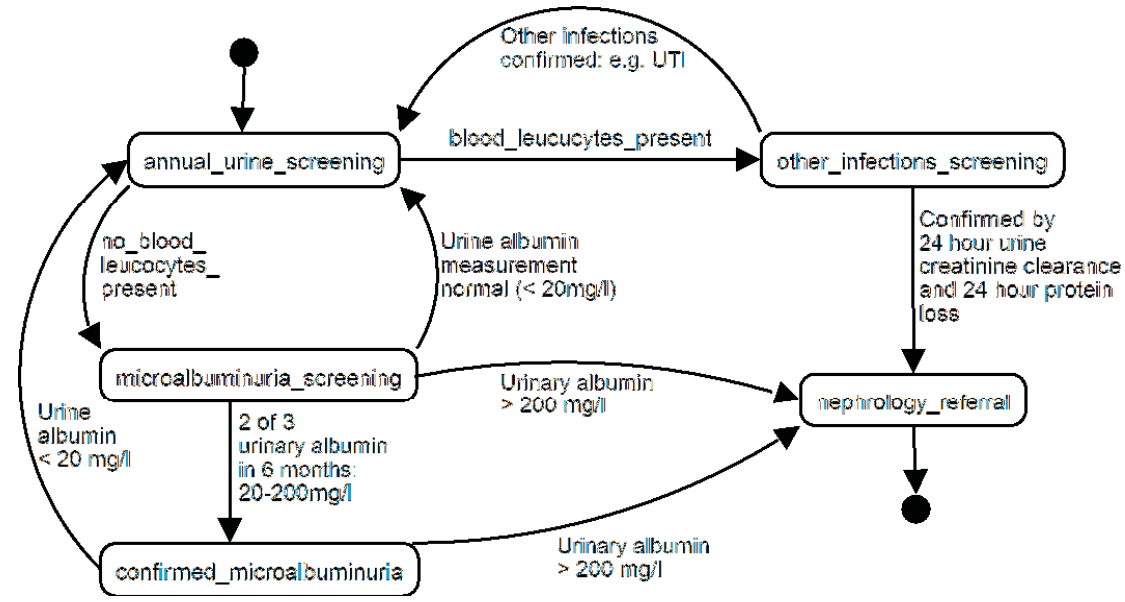


Figure 77 State chart for the microalbuminuria protocol

If ACR is in the range 20-200 mg/l in 2 of the 3 measurements taken over 6 months, then the patient is diagnosed with microalbuminuria. This diagnosis is confirming with the 24 hour creatinine clearance and protein loss measurements. If microalbuminuria is confirmed, then treatment and monitoring of microalbuminuria commences. At any point during the treatment of microalbuminuria, the patient is referred to the nephrologist if ACR is greater than 200 mg/l. The patient is also placed on annual screening if ACR drops to less than 20 mg/l.

The state chart of Figure 77 is used to generate event-condition-action (ECA) rules that implement the logic of the protocol. For each state and its associated transitions, rules are designed to handle the following:

a) Perform what must be done when the patient enters the state;
b) Perform what must be done during the patient's stay in the state;
c) Perform what must be done when a patient exits from the state;
d) Monitor the conditions that cause the patient to be moved from one state to another, i.e., conditions for state transitions.

Section 10.4. presents a demonstration of the creation of the MAP specification from the state chart presented in this Section.

10.4.2. Creating the MAP Specification

This Section presents the sets of Event-condition-action (ECA) rules that express the logic of the MAP. The ECA rules are derived with the aid of the state chart of Figure 77.

Each patient is initially placed on annual screening for microalbuminuria. This places the patient in the *annual-urine-screening(AUS)* state of the protocol as illustrated in the state chart of Figure 77. Table 10.3 presents three ECA rules, in both structured English and in the specification language PLAN, to implement the logic of the AUS state. The three ECA rules handle the following:

a) Schedule a dip-stick urine (DSU) test annually;
b) Place the patient on screening for other infections, i.e., change patient state to *other-infections-screening* state, when the DSU test turns out to be positive; and
c) Place the patient on microalbuminuria screening, i.e., change patient state to the *microalbuminuria-screening* state, when the DSU test turns out to be negative.

Specifying Rules for the MAP

Rules for Annual Urine Screening (AUS)

Table 10.3 Rules for the annual_urine_screening (AUS) state

	Rule Code (type)	Rule Description	Rule in PLAN
Annual_urine_screening (AUS)	AUS1 (static)	**ON** end of year **DO** perform dip-stick urine (DSU) test	STATIC_RULE AUS1, DESCRIPTION: dip-stick urine test at the end of every year for screening renal complications in diabetes patients, FROM: user_defined_date, STARTING: 0 year, ENDING: 1 year, ON EVERY: 1 year, DO: order_test('DSU');
	AUS2 (dynamic)	**ON** result of DSU test **IF** DSU test is positive (blood and leucocytes present in urine) **DO** put patient on screening for other infections	RULE AUS2, DESCRIPTION: if dipstick urine test shows presence of blood and leucocytes check presence or absence of other infections e.g. urinary tract infections, ON: result_arrival('DSU'), IF: DSU%result%database%t_results = positive%string, DO: patient_state ('other_infections_screening');
	AUS3 (dynamic)	**ON** result of DSU test **IF** DSU test negative (no blood and leucocytes in urine) **DO** micro-albuminuria Screening	RULE AUS3, DESCRIPTION: if dipstick urine test is negative then screen for microalbuminuria, ON: result_arrival('DSU'), IF: DSU%RESULT% DATABASE%T_RESULT = NEGATIVE%STRING, DO: PATIENT_STATE('microalbuminuria_screening');

Rules for Screening of Other Infections

A patient enters the *other-infections-screening (OIS)* state from the *annual-urine-screening* state when the dip-stick urine test is found to be positive.

Table 10.4 Rules for other_infections_screening (OIS)

	Rule Code (type)	Rule Specification in General ECA Rule Format	Rule Specification in PLAN
Other_infections_screening (OIS)	OIS1 (static)	**ON** entry into this state (OIS) **DO** check patient for urinary tract infection (UTI)	STATIC_RULE OIS1, DESCRIPTION: on entry to the OIS schedule the patient is tested for other urinary tract inections (UTI), FROM: start-of-schedule, STARTING: 0 munites, ENDING: 1 minute, ON EVERY: 1 minute, DO: order_test ('UTI');
	OIS2 (dynamic)	**ON** obtaining result for UTI examination **IF** UTI is *not* present **DO** 24 hour creatinine clearance and protein loss tests (24CRCL_PL)	RULE OIS2, DESCRIPTION: if UTI is not present then perform 24 hour creatinine and 24 hour protein loss tests, ON: result_arrival('UTI'), IF: UTI%result% database%t_result = negative%string, DO: order_test('24CRCL_PL');
	OIS3 (dynamic)	**ON** obtaining result for UTI examination **IF** UTI is present **DO** put patient back on annual screening for renal complications and treat the UTI	RULE OIS3, [DESCRIPTION: if UTI is present then place back on annual screening, ON: result_arrival("UTI"), IF: UTI%result% database%t_result = positive%string, DO: patient_state('annual_urine_screening');
	OIS4 (dynamic)	**ON** obtaining results for 24CRCL_PL **IF** 24CRCL_PL result is positive **DO** nephrology referral	RULE OIS4, DESCRIPTION: if 24 hour creatine clearance and 24 hour protein loss tests are positive then proteinuria is confirmed and refer patient to nephrologist, ON: result_arriavle('24CRCL_PL'), IF: 24CRCL_PL%RESULT%DATABASE%T_TEST = POSITIVE%STRING, DO: patient_state ('nephrology_referral');
	OIS5 (dynamic)	**ON** obtaining results for 24CRCL_PL **IF** 24CRCL_PL result is negative **DO** put patient back on annual screening for renal complications	RULE 0IS5, DESCRIPTION: if 24 hour creatine clearance and 24 hour protein loss is negative then return patient to annual screening, ON: result_arriavle('24CRCL_PL'), IF: 24CRCL_PL%RESULT%DATABASE%T_TEST = NEGATIVE%STRING, DO: patient_state ('annual_urine_screening');

Table 10.4 presents the rules that capture the knowledge on screening other infections within the context of the MAP. These rules, i.e., the *other-infections-screening* state rules, handle the following aspect of the MAP:

- On entry to this state, the patient is checked for urinary tract infection (UTI);
- The patient is placed back on annual urine screening with UTI treatment if UTI is found to be present;
- 24 hour creatinine clearance and protein loss (24CRCL-PL) are measured in the event that UTI is confirmed to be absent;
- Patient is referred to nephrologist if the 24CRCL-PL test is positive; and
- The patient is put back on annual urine screening (AUS) if the 24CRCL-PL test is negative.

Table 10.5 Rules for microalbuminuria_screening (MAS)

	Rule Code (type)	Rule Specification in General ECA Rule Format	Rule Specification in PLAN
microalbuminuria_screening (MAS)	MAS1 (static)	**ON** entry into this state **DO** order the tests ACR and SCR	STATIC_RULE MAS1, DESCRIPTION: at the start of this schedule MAS order the two ACR and SCR tests, FROM: start_of_schedule, STARTING: 0 minutes, ENDING: 1 minute, ON EVERY: 1 minute, DO: order_test('ACR, SCR');

	MAS2 (dynamic)	**ON** obtaining result for ACR
		IF ACR > 20
		DO order ACR twice at an interval of 2-3 months
			RULE MAS2,
			DESCRIPTION: if the first ACR result is > 20 mg/l order two more tests within the next six months,
			ON: result_arrival("ACR"),
			IF: ACR%RESULT%DATABASE%T_RESULTS
			> 20%DOUBLE,
			DO: ADD_RULE
			{
			STATIC_RULE ma2sr2
			DESCRIPTION rule orders ACR test during the next 6 month period
			*FROM time_rule_added
			*STARTING 0 months
			*ENDING 6 months
			*ON_EVERY 3 months
			*DO order_test ('ACR')
			};

	MAS3 (dynamic)	**ON** obtaining result for ACR **IF** ACR < 20 **DO** place patient on annual screening (AUS)	RULE MAS3, DESCRIPTION: if ACR < 20 mg/l then place patient on annual screening, ON: result_arrival('ACR'), IF: ACR%RESULT% DATABASE%T_RESULTS > 20%DOUBLE, DO: PATIENT_STATE('annual_urine_screening');
	MAS4 (dynamic)	**ON** result for ACR **IF** 2 of 3 ACR result is in range 20-200mg/l in 6 months **DO** place patient in state confirmed_microalbuminuria	**Alternative 1:** RULE MAS4a, DESCRIPTION: rule to analyse the 3 ACR measurements taken over 6 months FROM time_rule_added STARTING 0 months ENDING 6 months ON_EVERY 6 months DO: 2_of_3_ACR_check (); **Alternative 2:** RULE MAS4a, DESCRIPTION: rule to analyse the 3 ACR measurements taken over 6 months ON: result_arrival('ACR') DO: 2_of_3_ACR_check (); RULE MAS4b, DESCRIPTION: if 2 of 3 ACR in 20-200 mg/l within 6 months then microalbuminuria is confirmed, ON: 2_of_3_ACR_check(), IF: result = positive DO: PATIENT_STATE('confirmed_microalbuminuria');
	MAS5 (dynamic)	**ON** result for ACR **IF** ACR > 200mg/l **DO** put patient on nephrology_referral	RULE MAS5, DESCRIPTION: if ACR > 200 mg/l then refer patient to nephrologist for possible proteinuria, ON: RESULT_ARRIVAL('ACR'), IF: ACR%RESULT%DATABASE%T_TEST > 200%DOUBLE, DO: PATIENT_STATE('nephrology_referral');

Rules for Microalbuminuria screening (MAS)

A patient on annual urine screening (AUS) is put on microalbuminuria screening (MAS) when the DSU test is negative. Table 10.5 presents rules for handling MAS. The rules in Table 10.5 capture the following knowledge aspects of the MAP:

- Albumin creatine ratio (ACR) and serum creatinine ratio (SCR) tests are performed when the patient enters the AUS state;
- On-going ACR tests are scheduled at 2-3 months intervals if ACR > 20 mg/l. This gives rise to a rule that waits for the first ACR value and checks the condition, ACR>20. If the condition is satisfied, the rule adds a new static rule that suggest or prompts for an ACR measurement after every 2-3 months;
- The patient is placed on AUS if the condition, ACR<20, if satisfied;
- During a 6-month period, 3 ACR measurements must have been taken. If 2 of the three ACR values fall in the range 20-200 mg/l, then microalbuminuria is confirmed and the patient is moved to the confirmed microalbuminuria state (CMAS); and

- The patient is referred to a nephrologist if ACR exceeds 200 mg/l.

Rules for Confirmed Microalbuminuria

When a patient who is on microalbuminuria screening (MAS) experiences 2 ACR measurements in the range 20-200 mg/l out of 3 taken at an interval of 2-3 months for 6 months, the patient is scheduled for microalbuminuria treatment. The patient's state is changed to the *confirmed-microalbuminuria* state. Table 10.6 presents rules that capture the knowledge required to manage patients on microalbuminuria management. These rules take care of the following aspects of the protocol, MAP:

- On the patient's entry into the *confirmed-microlbuminuria* state, the following is done:
    1. The optimisation of glycaemic control is suggested;
    2. BP is measured;
    3. ACE inhibitor is administered if patient falls into the type 1 diabetes category; and
    4. Further ACR measurements are scheduled on a monthly basis;
- The patient is placed back on annual urine screening if it occurs that ACR < 20 mg/l at any time;
- The patient is placed on nephrology referral if it occurs that ACR > 200 mg/l at any time.

Table 10.6 Rules for confirmed_microalbuminuria (CMA)

	Rule Code (type)	Rule Specification in General ECA Rule Format	Rule Specification in PLAN
Confirmed_microalbuminuria(CMA)	CMA1 (static)	**ON** entry into this patient state **DO** optimise glycaemic control	STATIC_RULE CMA1, DESCRIPTION: at the start of this schedule suggest optimisation of glycaemic control, FROM: on_start_of_schedule, STARTING: 0 minutes, ENDING: 1 minute, ON EVERY: 1minute, DO: suggest ('glycaemic_control_optimisation');
	CMA2 (static)	**ON** entry into this patient state **DO** check BP	STATIC_RULE CMA2, DESCRIPTION: at the start of this schedule suggest BP measurement, FROM: start_of_schedule, STARTING: 0 minutes, ENDING: 1 minute, ON EVERY: 1 minute, DO: ORDER_TEST ('BP');
	CMA3 (static)	**ON** entry into this patient state **IF** diabetes_type = 1 **DO** prescribe ACE Inhibitor	STATIC_RULE CMA3, DESCRIPTION: If patient suffers from diabetes type 1 then prescribe ACE inhibitor, FROM: start_of_schedule, STARTING: 1 minute, ENDING: 1 minute, ON EVERY: 1 minute, DO: suggest_prescription ('ACE_inhibitor');
	CMA4 (static)	**ON** every 1 month **DO** order test ACR and SCR	STATIC_RULE CMA4, DESCRIPTION: ACR and SCR tests are performed every month for all microalbuminuria patients, FROM: start_of_schedule, STARTING: 0 months, ENDING: indefinite, ON EVERY: 1 month, DO: order_test ('ACR, SCR');
	CMA5 (dynamic)	**ON** obtaining result of ACR **IF** ACR < 20mg/l **DO** put patient on annual urine screening (AUS)	RULE CMA5, DESCRIPTION: if becomes normal (ACR < 20 mg/l) at any time then the patient is placed on annual screening, ON: result_arrival('ACR'), IF: ACR%RESULT% DATABASE%T_RESULT < 20%DOUBLE, DO: PATIENT_STATE('annual_urine_screening');
	CMA6 (dynamic)	**ON** obtaining result for ACR **IF** ACR > 200mg/l **DO** put patient on nephrology_referral (NPH)	RULE CMA6, DESCRIPTION: if becomes abnormal (ACR > 200 mg/l) at any time then the patient is placed on nephrology referral, ON: result_arrival('ACR'), IF: ACR%RESULT% DATABASE%T_RESULT > 200%DOUBLE, DO: PATIENT_STATE('nephrology_referral');

Rules for Nephrology Referral (NPH)

Table 10.7 presents the two rules that handle the preparation and sending of the patient's referral note.

Table 10.7 Rules for nephrology_referral (NPH)

	Rule Code (type)	Rule Specification in General ECA Rule Format	Rule Specification in PLAN
Nephrology_referral (NPH)	NPH1 (static)	**ON** entry to state **DO** create patient referral note	STATIC_RULE NPH1, DESCRIPTION: when a patient is referred to a specialist a patient referral note is created, FROM: start-of-schedule, STARTING: 0 minute, ENDING: 1 minute, ON EVERY: 1 minute, DO: referral_note ('nephrologist');
	NPH2 (dynamic)	**ON** creation of patient referral note **DO** e-mail to nephrologists or print patient referral note	RULE NPH2, DESCRIPTION: when a referral note is created it must immediately be sent to the specialist either by post or e-mail, ON: new_referral_note(), IF: true DO: send_referral_note();

When a patient is scheduled for referral to the nephrologist, a referral note should be prepared. The referral note also needs to be sent to the nephrologist either by post (printout) or by e-mail.

Specifying the MAP in PLAN

Table 10.8 presents the outline structure for the specifications of the schedules and the resulting outline structure for the protocol specification for the microalbuminuria protocol (MAP). The schedule and protocol rule sets are designed by following a few simple guidelines that will allow the resulting specification to conform to the guidelines presented in Section 6.6 of Chapter 6. These guidelines are summarised as follows:

- Each schedule corresponds to a state in the state chart of Figure 77. The schedule associated with the start state in the state chart is labelled START_STATE is the only schedule that will be active at the beginning of execution;
- For patient plans that are derived from protocols that involve patient state such as the MAP, schedule activation is effected through invoking the action PATIENT_STATE('patient-state-name') from a rule;
- Suppose it occurs that a rule, R1, in a schedule, S1, potentially triggers another rule, R2, in a second schedule, S2. In such a case, either R1 or R2 is moved from the schedule and placed into the protocol rule set;
- In general, rules that monitor changes in the state of a patient are good candidates for belonging to the protocol rule set; and

Table 10.8 Specification of the Microalbuminuria Protocol (MAP)

<table>
<tr>
<td rowspan="2">Structure of the MAP Schedules in PLAN</td>
<td>
Annual Urine Screening (AUS)

^SCHEDULE^ AUS,
DESCRIPTION: This is a microalbuminuria protocol schedule called AUS for Annual dipstick Urine Screening;
START-STATE;
RULE AUS2,<body of rule AUS2>;
RULE AUS3,<body of rule AUS3>;
^END SCHEDULE^

Othe Infections Screening (OIS)

^SCHEDULE^ OIS,
DESCRIPTION: This is a microalbuminuria protocol schedule called OIS for SCREENING OTHER INFECTIONS in the diagnosis of microalbuminuria and proteinuria;
RULE OIS2,<body of rule OIS2>;
RULE OIS3,<body of rule OIS3>;
RULE OIS4,<body of rule OIS4>;
RULE OIS5,<body of rule OIS5>;
^END SCHEDULE^

Nephropathy Referral (NPH)
^SCHEDULE^ NPH,
DESCRIPTION: This is a microalbuminuria protocol schedule named NPH for nephrology referral – handles preparation and transmission of the necessary documentation for the referral;
RULE NPH2,<body of rule NPH2>;
^END SCHEDULE^
</td>
<td>
Microalbuminuria Screening (MAS)

^SCHEDULE^ MAS,
DESCRIPTION: This is a microalbuminuria protocol schedule called MAS for the screening of microalbuminuria;
RULE MAS2,<body of rule MAS2>;
RULE MAS3,<body of rule MAS3>;
RULE MAS4,<body of rule MAS4>;
RULE MAS5,<body of rule MAS5>;
^END SCHEDULE^

Confirmed Microalbuminuria (CMA)

^SCHEDULE^ CMA,
DESCRIPTION: This is a microalbuminuria protocol schedule named CMA for confirmed microalbuminuria – handles treatment and control of microalbuminuria;
RULE CMA5,<body of rule CMA5>;
RULE CMA6,<body of rule CMA5>;
^END SCHEDULE^
</td>
</tr>
<tr>
<td>Structure of the MAP Specification in PLAN</td>
<td colspan="2">
MAP Specification

@PROTOCOL@ MAP;
DESCRIPTION: This is a protocol for the diagnosis and management of microalbuminuria in diabetes patients;
CREATOR: DR JOHN NOLAN;
CATEGORY: DIABETIC_NEPHROPATHY;
#SCHEDULE_SET#
^SCHEDULE^ AUS, <AUS_rules> ^END SCHEDULE^
^SCHEDULE^ OIS, <OIS_rules> ^END SCHEDULE^
^SCHEDULE^ MAS, <MAS_rules> ^END SCHEDULE^
^SCHEDULE^ CMA, <CMA_rules> ^END SCHEDULE^
^SCHEDULE^ NPH, <NPH_rules> ^END SCHEDULE^
#END SCHEDULE_SET#
~RULE_SET~
STATIC_RULE AUS1,<body of rule AUS1>;
RULE OIS1,<body of rule OIS1>;
RULE MAS1,<body of rule MAS1>;
RULE CMA1,<body of rule CMA1>;
RULE CMA2,<body of rule CMA2>;
RULE CMA3,<body of rule CMA3>;
RULE CMA4,<body of rule CMA4>;
RULE NPH1, <body of rule NPH1>;
~END RULE_SET~
@END PROTOCOL@
</td>
</tr>
</table>

- All rule activation cycles should be identified and approved by a domain expert. Rules activation cycles that are not permitted from the domain perspective should be eliminated by revising the rule design.

By applying these guidelines to the rules obtained with the aid of the state chart for the MAP, the specification for the MAP with the outline structure and content presented in Table 10.8 is obtained. The complete PLAN specification for the MAP is presented in the Appendix C.

10.5.The TOPS Database for the MAP Specification

The MAP specification, which is expressed in the protocol specification language PLAN, was parsed by the TOPS plan parser described in Section 9.4). The MAP specification was stored

in the TOPS database where it can be managed. In the TOPS database, the MAP specification attributes are stored in a set of relational tables, which are illustrated in Appendix E. Once stored in the relational database, the MAP specification can be queried, manipulated and converted to XML for sharing. For a full listing of the parser output the reader is referred to Appendices D. Appendix I.1 illustrates the TOPSQL command for displaying the MAP protocol specification after retrieving it from the TOPS relational database.

10.6.Executing the MAP in TOPS

10.6.1. Creation of Plans from the MAP

To execute the MAP with respect to a given patient, an instance of the MAP that is specific to the individual patient is created, the *patient plan.* To create the patient plan, TOPS first retrieves the protocol specification and then uses it to create a patient plan by customising the MAP rules so that they apply specifically to the individual patient. The customisation process involves:

a) Specifying absolute time points for static rules for the patient. This may require prompting for further information from the domain expert;
b) Assigning absolute values specific to the patient to domain-dependent terms within the protocol, e.g., a term like *date-of-conception* may be replaced by the value *15 January 2004.* This may also require interaction with the domain expert or the electronic medical record; and
c) Making each rule focus its monitoring and its action on this particular patient. In other words, the rule is made to react only to events happening to this individual patient only.

For a full listing of the execution log for a TOPS session for creating a MAP plan, the reader is referred to Appendix F.

10.6.2. The MAP Plan Installation and Activation

Once the patient plan is created in TOPS, it is installed and activated in order to start its execution process. The installation of the patient plan in TOPS involves the following:

a) Creating the patient plan specification database. This database allows the patient-specific instance of the MAP to be managed effectively;

b) Generating the SQL code for and creating the Oracle database triggers that implement each patient plan rule. TOPS performs this task automatically without user intervention;
c) Creating the Java object-based time triggers for the static rules in the patient plan;
d) Ensuring that any special requirements such as system monitors are up and running.

Once the installation process is completed, the patient plan is activated and ready to start execution. For a full listing of a TOPS session for the MAP patient plan installation, activation and execution the reader is referred to Appendix G.

10.6.3. The MAP Execution Process

TOPS' execution of a patient plan is essentially event-driven and follows the ECA rule execution pattern. In addition, a TOPS protocol and hence all plans derived from it, may or may not, involve patient states. TOPS distinguishes between the two type of protocols by inspecting the set of rule actions for the patient state action and the starting schedule. The MAP involves patient states that are implemented through a rule action that changes the state of a patient. The state of a patient is protocol-dependent. In a TOPS protocol specification, patient states are represented by schedules in the protocol specification and the schedule should have the same name as the state that it represents. To know which rules belong to which state, TOPS simply queries the protocol specification database for rules that belong to the state's corresponding schedule. The schedule associated with the start state is, by default, active on installation of the patient plan. Each change in patient state requires that only rules associated with that state be active. Other rules remain deactivated until the patient state changes to that associated with the rules. In addition to the deactivation of rules belonging to the previous patient state, the rule action that changes the state of a patient also activates rules belonging to the new state. Patient state changes are effected by invoking the command PATIENT_STATE('*new_state'*) in the action part of the ECA rule.

The MAP execution is initially triggered by the *annual_urine_screening (AUS)* rule, AUS1, which suggests, on an annual basis, that the dip-stick urine (DSU) test be performed on the patient. The result of the DSU test triggers either of the rules: AUS2 and AUS3. The patient is subject to rule AUS1 for as long as he keeps being referred back from to the AUS state either the *other_infections_screening (OIS)* state or the *microalbuminuria_screening (MAS)* state because either the patient has no other urinary tract infections (UTI) or has no microalbuminuria (MA). If the patient is found to have UTI while in the OIS state, then the rule OIS4 moves the patient into the *nephrology_referral* (NPH))state, which effectively

terminates the execution of the MAP. If the patient is found to have microalbuminuria while in the MAS state, then the rule MAS4 moves the patient into the *confirmed_microalbuminuria (CMA)* state for the treatment and management of this clinical condition. If this management succeeds, rule CMA5 returns the patient to the AUS state and if the patient's condition becomes worse, rule CMA6 moves the patient to the NPH state, which also effectively terminates TOPS execution of the MAP. The full listing of TOPS' session for the execution of the MAP is presented in Appendix G.

10.7. Managing the MAP in TOPS

The management of the MAP includes functionality offered by each of the three management planes that have been introduced in Chapter 3 and further explained in Chapter 5 of this Book. The specification and execution of the MAP in TOPS have already been described in the previous sections. The management of the MAP also includes the provision of the ability to manipulate the MAP using the high-level declarative manipulation language, TOPSQL. The manipulation of the MAP consists of performing operations on the MAP specification and the patient plans derived from it as well as issuing queries against both of these aspects of managing the MAP. The next subsections discuss the manipulation of the MAP in TOPS.

10.7.1. Operations Performed on the MAP Specification and Patient Plans

TOPS supports mainly three operations that are to be performed on objects within the system. The three operations are supported according to the manipulation approach presented in Section 8.3 of Chapter 8 of this Book. These operations are:

a) *Addition:* rules can be added to the MAP protocol specification and also to the patient-specific plans that represent instances of the MAP protocol. For instance, it can be noted that rule CMA1 suggest that the *serum creatinine ratio (SCR)* be measured and rule CMA3 suggest that the patient's *blood pressure (BP)* be measured but there is no other rule that makes a follow-up on the results of these two measurements. To provide for this follow-up, new rules will need to be added to the MAP specification and also to the patient plans. Since the MAP specification and the MAP patient plans are stored in the database, adding new rules can be performed effectively within the framework and context of security, concurrency and integrity constraints provided by the DBMS.

b) *Modification:* Provision is made in TOPS for rules of the MAP to be modified and updated. For example, suppose a new test has been developed for aiding the diagnosis of microalbuminuria. Suppose further that a healthcare organisation that uses the MAP has decided to use this test in place of the albumin creatinine ratio measurement. In this case, there is a need for the rules MAS1-6 and CMA4-6 will need to be edited and updated to accommodate the healthcare organisation's new preference.

c) *Deletion:* Rules in the MAP specification or patient plans can be deleted on the fly. For example, suppose in one healthcare organisation, healthcare experts are convinced that glycaemic control is already optimised for all their diabetic patient. In such a case, the healthcare experts may decide to delete rule CMA1, which suggests the optimisation of glycaemic control.

These operations can be performed on the fly with no recourse to parsing the protocol again or re-installation of the individual patient plans. This is possible due to the characteristic modularity of the ECA rule paradigm. However, this strength of the ECA rule paradigm is also its weekness in supporting the management of ECA rules. For instance, suppose one had added a rule, named CMA3a, to follow up on a patient's BP to the patient's plan derived from the microalbuminuria protocol and then, at a later time, one deleted the rule CMA3, which suggest the measurement of BP. In such a case, rule CMA remains waiting in the active state with no potential of ever being fired or executed. This problem can only be handled if a mechanism exists to analyse and maintain dependencies among rules in a single patient plan. In other words, some form of *rule dependency constraints* for patient plans need to be introduced and a *rule dependency constraint enforcement mechanism* for these constraints needs to be developed. This book has not addressed this problem, leaving it to future work. Currently, TOPS does not have such a constraint enforcement mechanism so will not be able to handle this scenario properly.

10.7.2. Querying the MAP Specifications and Patient Plans

An important aspect of protocol knowledge and information management is the ability to query the protocol specifications and their individual patient-specific instances. The framework presented in Chapters 5 and 8 of this book describes the functionality to query the specifications of the MAP as well as its execution process. In other words, both the static and the dynamic aspects of the protocol can be queried in TOPS.

The dynamic aspect of a protocol refers to the execution process whose evolution can be queried the along the temporal dimension. At any one moment during the plan's execution the patient plan's rule composition may be different from that at a later or an earlier instant. In the case for the MAP, as a patient is moved from one state to another, schedules and rules are deactivated while others are activated. Further to this, some new rules may be introduced. As result, this gives rise to an interesting type of query that requests for the executing patient plan at a given point in time or during a given time period in the past..

Another useful type of query is the request for a replay of the execution of the MAP patient plan that occurred during a given time interval in the past. The output of such a query is effectively a simulation of all rule executions that occurred during the time interval in question. This feature is currently not fully implemented in the current version although the design of TOPS takes it into consideration.

The reader is referred to the Appendix I for a sample of queries and results of these queries in the TOPS context of the manipulation of the MAP.

10.8. Case Study Findings and Discussion

The findings of this case study can be summarised as follows:

Method of capturing and specifying guideline/protocol: The use of the highly intuitive state chart makes it easy to communicate with domain experts during guideline/protocol information/knowledge elicitation, capture and specification. The use of the UML state chart also makes the subsequent extraction of the relevant ECA rules easier since the state chart naturally supports the ECA rule paradigm (Calestam 1999; Berndtsson, Mikael and Calestam 2001) and is easily understood by domain experts.

Creating the computerised protocol specification: The TOPS protocol specification parser, which uses an object-based mapping between the PLAN specification and the underlying relational database for storing protocol specifications, proved to be efficient and effective as a simple tool for creating the ECA rule-based protocol specifications in the database.

The database of protocol specifications: The microalbuminuria protocol specification was stored in the Oracle relational database. A single protocol specification in the database consisted of components that were spread over several relations/tables. This offered a simple way to visualise specification information using the familiar tabular format.

Easy manipulation of information: The relational database model was found to offer a uniform and flexible way to access, manipulate and query all information from specification, to executing process state, to data in the patient record. Flexibility was guaranteed by the SQL, which allows queries that combine data on attributes from several entities subject to constraints within the database.

Dynamic generation of SQL code for triggers that implement ECA rules in protocols: The generation of SQL trigger code that implement the ECA rules of the MAP was automatically supported by TOPS and required no user intervention. This makes it easy for application domain experts to use TOPS with no knowledge of the SQL trigger specification language. However, domain experts still needed to be familiar with the protocol specification language, PLAN, which should ideally be closer to their domain language than the SQL.

Protocol action support: The execution of the action specified in any protocol rule is subject to the availability of the appropriate software module that implements the action. In other words, rule actions in the microalbuminuria protocol needed to be predefined and any new action required by the protocol requires that the module to implement such an action be developed. However, once the action software modules were developed, they were generic and re-usable by other protocols.

Challenges from the manipulation of complex information: High-level operations or queries on protocols and/or their components were implemented using a number of operations or queries on several relational tables. This may be a significant overhead in terms of performance. An important limitation to this case study is the lack of performance benchmark measures on the DBMS query processing and the execution of protocols by means of database triggers. In order to facilitate the performance of useful operation and queries on complex objects such as protocols and patient plans, there was a need for the development of generic and specialised software modules that provide support for TOPSQL at a level that is higher than the SQL to avoid repeated typing of several queries to perform one conceptually higher level operation.

Challenges expected from the integration of TOPS into the clinical environment: TOPS's protocol execution relies on monitoring events occurring within the patient's medical record. A number of factors prevented the TOPS implementation of the MAP to be deployed within a real clinical environment. These factors included the fact that existing systems used in the hospital had proprietary interfaces and database schemes whose specification could not be not be obtained due to licencing, security and confidentiality issues. For example, the

diabetes patient record in St James's Hospital is implemented in a system called Diamond, which is based on MS Access and MS SQL and whose schema and interfaces were inaccessible due to the nature of its licence as well as concerns about patient confidentiality and security of information. Furthermore, a large part of the diabetes patient record was still paper-based. TOPS itself had the limitation that it lacked an appropriate user interface suitable for clinicians to specify protocols in PLAN. Although the clinician being consulted found it easy to understand protocols written in PLAN, we do not expect clinicians to work directly with PLAN when using TOPS. The development of a user-friendly interface required effort and time, which was not available and so has been left as part of future work.

10.9. Chapter Summary

This Chapter has demonstrated the applicability and effectiveness of the framework, approach and method presented in Chapters 5-8 of this Book by using the proof-of-concepts system, TOPS, to support the management of the microalbuminuria protocol (MAP) for diabetes patients. It was shown that the microalbuminuria protocol knowledge can be modelled and specified by using the ECA rule paradigm guided by the state chart. The functionality provided by the three management planes presented in Chapter 3 and further explained in Chapters 5-8 are then made available through TOPS for application to the protocol. The specification language, PLAN, was used to specify the resulting protocol specification. Once a PLAN specification is obtained, TOPS is used to store the specification in the database for effective management, thus, making it possible to execute, perform operations and query both various aspects of the MAP using the manipulation language, TOPSQL. An important limitation of this case study is that the implementation of the MAP protocol in TOPS was not evaluated in a real clinical environment. Due to this limitation, an evaluation of TOPS in real practice and, hence, of the framework and approach that it embodies, cannot be made at this time.

PART IV: CONCLUSION

Chapter 11 Conclusion

11.1. Introduction

The research challenge in this study was to investigate into the management of information and knowledge for supporting the complex domain of computerised clinical guidelines and protocols (CGPs) with the aims of coming up with a generic and unified management framework and approach for supporting computerised CGPs by using the event-condition-action (ECA) rule paradigm as currently supported in modern advanced database systems. This research problem can be broken down into two specific challenges as follows:

The challenge from the clinical guideline and protocol domain: The demand for the incorporation of clinical guidelines/protocols for patient care into the clinician's daily routine as a way to reduce clinical practice variation, improve quality, contain costs and optimise resource utilisation has led to calls for the computerisation of clinical guidelines/protocols as one method of contributing to the promotion of clinicians' acceptance and compliance.

The challenge for the computing domain: The management of computerised CGPs poses a major challenge to the information management domain. Since the ECA rule paradigm has proved to be promising in specifying medical knowledge through the Arden Syntax for Medical Logic Modules (MLMs) (ASTM 1992; HL7 1999), it is worthy investigating further its application in the computerisation of clinical guidelines or protocols. Using the ECA rule paradigm to manage computerised CGPs also offers the challenge to demonstrate a practical requirement for further improvements to the ECA rule paradigm support in modern database systems. Furthermore, in literature, computer-based CGPs have been supported mainly with respect to their specificatioin and execution. The challenge is to develop a CGP management framework that also incoparates the dimension of the manipulation of knowledge and information. This should involve performing operations and issuing queries.

This chapter is a conclues this book by presenting a review of this book in Section 11.2; a summary of the main contributions in Section 11.3; an outline of the benefits arising from the outcomes of this research in Section 11.4; identification of the limitations as the basis for pointers to future directions in Section 11.5; and, finally, an objective evaluation of the study presented in this book in Section 11.6.

11.2. Review

Supporting the management of CGPs is seen in the literature to be involving mainly the provision of expressive specification languages and flexible execution mechanisms for the CGPs. Thus, once a CGP is specified and in execution, it is not easy to manipulate the information and knowledge that is incorporated in the CGP systems. This book has provided for the manipulation of CGP information and knowledge within the framework for the management of CGPs. The SpEM framework has been developed to provide CGP management in terms of the three planes for the specification, execution and manipulation of CGPs. This framework has been supported in the MonCooS approach and method, which provides CGP management functionality for allowing protocols to be specified in the specification language, PLAN, and stored in a database; executed by using ECA rule-based mechanism whose implementation is based on database triggers; and manipulated using the manipulation language, TOPSQL. The prototype system, TOPS, was developed to implement the MonCooS approach for the management of CGPs for clinical test-ordering by clinicians. In TOPS, use of the ECA rule support and data management functionality in a modern DBMS has been made in order to support the management of CGPs. TOPS uses the database trigger mechanism of the Oracle9i DBMS as the CGP execution engine with both the CGP specification database, the patient record and CGP execution state data held within the Oracle9i DBMS. The case study on the management of the microalbuminuria protocol uses TOPS to show that a real protocol can be specified, executed and manipulated according to the SpEM framework and the MonCooS approach.

11.3. Contributions

The contributions of this Book can be summarised as follows:

- A characterisation of the problem of managing CGP information as consisting of the three generic planes of specification, enforcement/execution and manipulation, with each plane having its own levels of abstraction and interacting, in a dynamic fashion, with the other two planes.
- A unified framework, SpEM, together with a comprehensive approach, MonCooS, for supporting the management of clinical guidelines and protocols (CGPs). The SpEM framework incorporates the manipulation of CGP knowledge and information as an

additional dimension to the dimensions of specification and execution, which are commonly supported in the literature.

- An approach that uses the ECA rule paradigm for both modelling and implementation of CGPs within the context of a unified framework; and a demonstration that the ECA rule paradigm is a viable technology for real applications (such as the management of CGPs) and needs further comprehensive support in modern database management systems;
- An advanced mechanism and general platform for manipulation of complex information and its implementation in a tool, called TOPS, for CGP management as a demonstration that the framework and approach developed in this study can be applied in practice; and
- A case study that applies the prototype system, TOPS, to the case for managing a microalbuminuria protocol for diabetic patients. The microalbuminuria protocol was drawn with the help from domain experts from a local Dublin hospital.

11.4. Benefits of Research Outcomes

The SpEM framework, and the MonCooS approach together with the prototype system, TOPS, can be beneficially applied in other applications. Applications that could benefit are those that require assistance with the monitoring of situations, timely interventions and response and coordination tasks in which dynamic manipulation of domain information is important. Another general characteristic of applications that could benefit are those that make use of domain information and knowledge that is specified and used to establish and enact interventions (actions, tasks and activities) that need to be performed within the context of a specific application domain problem. For instance, the support for business/clinical workflow, which could be specified, executed and manipulated according to the SpEM framework and the MonCooS approach. In insurance and credit policy management, generic policies could be specified using PLAN-based language and the appropriate customisations could be applied to them to create specific insurance or credit policies that suits the circumstances of each individual customer or group of customers. These policies could then be enforced and managed from year to year until their maturity period expires or until they are terminated accordingly.

11.5. Limitations and Future Directions

The successful support for the management of CGPs depends on the easy, accurate capture and specification of CGP information. Chapter 6 described a methodology for capturing domain knowledge. Further investigations are needed to enhance and validate the methodology with the aim of making it easy to use by clinician. It is also necessary to investigate into a practical and formal method to augment domain expertise in analysing and verifying the correctness of protocol specifications and patient plans by using techniques for active rule analysis (Bailey, JA 1997; Baralis, E., Ceri et al. 1998; Bailey, J, Poulovassilis et al. 2000).

The specification language, PLAN, needs further enhancements in a number of aspects. First, CGPs are also considered to be clinical algorithms that can be expressed by means of flowcharts, which incorporates constructs for sequences, repetition and parallelisation of patient care actions, tasks and activities. PLAN needs enhancements to provide for the three constructs: sequencing, repetition or iteration and concurrency. An investigation needs to be carried out to determine how these constructs can be supported in cooperation with the ECA rule paradigm. Second, further enhancements are required in PLAN to exploit the research results from the Active Databases by incorporating a more expressive event language and algebra for specifying composite and temporal events and conditions. Third, there is a need to move towards introducing sharability and portability through the standardisation of PLAN by making it an XML-based rule language. Use could be made of concepts from XRML (Lee and Sohn 2003) and RuleML (Boley, Tabet et al. 2001) as well as XML-based storage formats and XML query languages for the manipulation of knowledge and information for CGPs. Alternatively, the only existing HL7 standard (HL7 1999) for specifying medical knowledge modules, the Arden Syntax (Hripscak, Luderman et al. 1994), could be investigated in order to find a way for using it as a sub-language for specifying ECA rules in PLAN. Fifth, it is necessary to develop a method and tools for the creation of protocol specifications in an intuitive way, e.g., enabling domain experts (clinicians) to use a GUI method of creating and viewing PLAN specifications.

This book has proposed the use of an ECA rule mechanism of a modern DBMS as the core engine for protocol execution. The extremely limited support for ECA rules within modern database systems makes the task of supporting the SpEM framework and the MonCooS approach difficult.

First, the real world events in PLAN specifications need to be mapped to event model of the DBMS trigger mechanism. Currently, TOPS implements a basic mechanism to map events in PLAN specifications to database trigger events, there is a need to develop a formal model and a software mechanism for this mapping.

Second, temporal or time events are not supported in the database trigger mechanism. Currently, TOPS implements a basic time trigger mechanism which does not support temporal events. *Third,* temporal conditions are limited to the temporal features allowed in SQL conditions. Further, trigger conditions suffer from SQL statement restrictions, for instance, in the Oracle9i DBMS, trigger conditions may not contain the SELECT statement and cannot make calls to stored procedures and functions. Future work would investigate the development and implementation of a comprehensive event and condition specification models that can be implemented to work with a modern DBMS such as the Oracle database system.

This book has not addressed the issue of how to define and maintain *inter-dependencies between rules* in a protocols specified in PLAN. Hence, the manipulation operations of addition, modification and deletion of rules in a protocol or an executing patient plan in TOPS is currently not subject to any form of constraints as pointed out in Section 10.7.1. Further work is required to investigate the specification and enforcement of what may be called *rule dependency constraints* for PLAN-based protocols and the patient plans derived from them.

While the use of a modern DBMS as the protocol execution provides the opportunity to make use of security functionality existing within the DBMS, this book has not addressed the issues of security and confidentiality, which are of fundamental significance within a patient care setup. Future work would investigate the incorporation of a security and confidentiality model that is suitable for the clinical environment and also incorporates the underlying DBMS security mechanisms.

In practice, patients may suffer from co-morbidities, i.e., more than one medical problem at one time. For instance diabetes patients may also have vascular, eye and renal complications. This demands that a provision be made for patients to be placed into more than one category and to have more than one patient plan at a time. The handling of co-morbidities has not been dealt with in this book and, consequently, the prototype system TOPS does not make a provision for managing co-morbidities. Future work would investigate how to handle co-morbidities in a safe way.

The prototype system, TOPS, and the protocol, MAP, developed in the case study have not yet been put to actual use in a real clinical setting although the protocol was developed with the help of a practicing medical expert. This represents a limitation in the form of the lack the clinical validation of the work presented in this book.

This book has presented a new framework, approach and method for supporting the management of CGP information and knowledge. The book has also argued that using the ECA rule paradigm and active database systems to support this framework, approach and method would result in effective support for the management of CGP information and knowledge while focusing mainly on monitoring and coordination, and deliberately leaving the reasoning task to the domain expert. This is essentially a qualitative argument. The only proof of whether or not the resulting software environment is of better quality than other existing software for CGP management support may be obtained by applying the developments of this book to real-world circumstances for CGP management.

REFERENCES

Aho, A V and Ullman, J D (1973). *Theory of Parsing, Translation and Compiling*, Prentice Hall Professional Technical Reference.

American Diabetes Association (2002). *Position Statement: Diabetic Nephropathy*. Diabetes Care **25**(Supplement 1): *s85-s89*.

Andreassen, S, Gomez, E J and Carson, E R (2002). *Introduction: Computers in Diabetes 2000*. Computer Methods and Programs in Biomedicine **69**: *93-95*.

Appelrath, H J, Behrends, H, Jasper, H and Kamp, V (1994). *Active database technology supports cancer clustering*. Applications of Databases: 1st International Conference (ADB-94), Vadstena, Sweden, Springer. **819:** *351-364*.

Appelrath, H-J, Behrends, H, Jasper, H and Zukunft, O (1995). *Case studies on active database applications: Lessons learned. Technical Report*. Oldenburg, University of Oldenburg.

ASTM (1992). *Standard Specification for Defining and Sharing Modulor Health Knowledge Bases (Arden Syntax for Medical Logic Modules)*. Annual book of ASTM Standards. Philadelphia, American Society of Testing and Materials. **14.01:** *539-587*.

Audet, A, Greenfield, S and Field, M (1990). *Medical practice guidelines: current activities and future directions*. Ann Intern Med **118**(9): *709 -714*.

Bailey, J, Poulovassilis, A and Newson, P (2000). *A dynamic approach to termination analysis for active database rules*. 1st International Conference on Computational Logic (DOOD'2000 Stream), London. *1106 -1120*.

Bailey, J A (1997). *On the Foundation of Termination Analysis of Active Database Rules*. Computer Science. Melbourne, University of Melbourne**:** *126*.

Baralis, E, Ceri, S and Paraboschi, S (1996). *Modularization techniques for active rules design*. ACM Transactions on Database Systems (TODS), ACM Press , New York, NY, USA **21**(1): *1 - 29*.

Baralis, E, Ceri, S and Paraboschi, S (1998). *Compile-time and runtime analysis of active behaviors*. IEEE Transactions on Knowledge and Data Engineering, **10**(1041-4347): *353-370*.

Bates, W, Kuperman, G J, Teich, J M, Tanasijevic, M J, Ma'Luf, N, Rittenberg, E, Jha, A, Fiskio, J and Winkelman, J (1999). *A Randomised Control Trial Of Computer-Based Intervention to Reduce Utilisation Of Redundant Liboratory Tests.* JAMIA **106**(2): *144-50*.

Benjamins, R, Fensel, D and Gomez, P A (1998). *Knowledge Management through Ontologies.* The Second International Conference on Practical Aspects of Knowledge Management (PAKM'98), Basel, Switzerland. *5.1-5.12*.

Berndtsson, M (1994). *Reactive object-oriented databases and CIM*. 5th Intl Conf on Database and Expert System Applications (DEXA'94), Athens. *769-778*.

Berndtsson, M and Calestam, B (2001). *A Uniform Approach for Supporting Active Database Features in UML and OMT*. Skovde, Computer Science Department, University of Skovde.

Berndtsson, M, Chakravarthy, S and Lings, B (1996). *Coordination Among Agents Using Reactive Rules*. Skovde, University of Skovde**:** *23*.

Berry, D, Wu, B, Pardon, S, Duignan, F, Grimson, W, Gaffney, P, Clarke, F and Feely , J (1999). *A Test Request Protocol System*. IFCC WorldLab Conference, Bologna, Italy.

Bindels, R, de Clercq, P A, Winkens, R A G and Hasman, A (2000). *A test ordering system with automated reminders for primary care based on practice guidelines*. Int J Med Inform **58-59**: *219-233*.

Boley, H, Tabet, S and Wagner, G (2001). *Design Rationale of RuleML: A Markup Language for Semantic Web Rules*. SWWS'01, Stanford.

Booch, G (1993). *Object-Oriented Analysis and Design with Applications*, Addison-Wesley Pub Co. *608*.

Boran, G, O'Moore, R, Grimson, W, Peters, M, Hasman, A, Groth, T and van Merode, F (1996). *A new clinical laboratory information system architecture from the OpenLabs Project offering advanced services for laboratory staff and users*. Clinical Chemistry Acta **248**: *19-30*.

Borghoff, U M and Pareschi, R (1997). *Information Technology for Knowledge Management*. Journal of Universal Computer Science **3**(8): *835-842*.

Bowie, J and Barnett, G O (1976). *MUMPS - An Economical and Efficient Time-Sharing Language for Information Management*. Computer Methods and Programs in Biomedicine **6**: *11-21*.

Buckingham Shum, S (1998). *Negotiating the Construction of Organisational Memories*. Information Technology for Knowledge Management. Borghoff, UM and Pareschi, R. Berlin, Heidelberg, New York, Springer-Verlag**:** *55-78*.

Caironi, P V C, Portoni, L, Combi, C, Pinciroli, F and Ceri, S (1997). *HyperCare: a Prototype of an Active Database for Compliance with Essential Hypertension Therapy Guidelines*. AMIA Ann Fall Symposium, Philadelphia, PA, Hanley and Belfus. *288-292*.

Calestam, B (1999). *OMT-A: An Extension of OMT to Model Active Rules*. Computer Science. Skovde, University of Skovde**:** *133*.

Ceri, S, Cochrane, R J and Widom, J (2000). *Practical Applications of Triggers and Constraints: Successes and Lingering Issues*. 26th International Conference on Very Large Databases (VLDB), Cairo, Egypt. *254-262*.

Ceri, S, Grefen, P and S'anchez, G (1997). *WIDE - A Distributed Architecture for Workflow Management*. International Workshop on Research Issues in Data Engineering (RIDE), Birmingham, UK, IEEE Computer Society Press. *76--79*.

Ceri , S and Ramakrishnan, R (1996). *Rules in database systems*. ACM Computing Surveys (CSUR) **28**(1): *109-111*.

Chaudhry, N, Moyne, J and Rundensteiner, E A (1998). *Active Controller: utilizing active databases for implementing multistep control of semiconductor manufacturing*. IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part C, **21**: *217-224*.

Chen, P (1976). *The Entity-Relationship Model: Toward a Unified View of Data*. ACM Transactions on Database Systems **1**(1): *9 - 36*.

Clayton, P D, Pryor, T A, Wgertz, O B and Hripcsak, G (1989). *Issues and Structures for Sharing Medical Knowledge Among Decision-making Systems: The 1989 Arden Homestead Retreat.* Annu Symp Comput Appl Med Care. *116-121*.

Collet, C, Habraken, P, Coupaye, T and Adiba, M (1994). *Active rules for the Software engineering platform GOODSTEP*. 2nd International Workshop on Database and Software Engineering, 16th International Conference on Software Engineering, Sorrento, Italy.

Cyran, M (2002). *Oracle9i Database Concepts, Release 2 (9.2)*. Redwood Shores, CA 94065, USA, Oracle Corporation.

Dayal, U, Blaustein, B, Buchmann, A, Chakravarthy, U, Hsu, M, Ledin, R, McCarthy, D, Rosenthal, A and Sarin, S (1988). *The HIPAC Project: Combining Active Databases and Timing Constraints*. SIGMOD Record **17**(1): *51-69*.

Dazzi, L, Fassino, C, Saracco, R, Quaglini, S and Stefanelli, M (1997). *A Patient Workflow Management System Built on Guidelines*. AMIA Annual Fall Symposium.

de Clercq, P A, Blom, J A, Hasman, A and Korsten, H H M (1999). *GuiDE: an architecture for the acquisition and execution of clinical guideline-application tasks*. Belgium-Netherlands Conf on Artificial Intelligence, Maastricht. *171-172*.

de Clercq, P A, Blom, J A, Hasman, A and Korsten, H H M (2000). *An ontology-driven approach for the acquisition and execution of clinical guidelines*. Medical Informatics Europe, Hannover, IOS Press. *714-719*.

de Clercq, P A, Hasman, A, Blom, J A and Korsten, H H M (2001). *Design and implementation of a framework to support the development of clinical guidelines*. Int J Med Inf. **64**(2-3): *285-318*.

Diaz, O, Jaime, A, Paton, N W and al-Qaimari, G (1994). *Supporting dynamic displays using active rules*. SIGMOD Record, ACM Special Interest Group on Management of Data **23**: *21-26.*

Dittrich, K R, Gatziu, S and Geppert, A (1995). *The Active Database Management System Manifesto: A Rulebase of ADBMS Features*. 2nd Workshop on Rules in Databases (RIDS), Athens, Greece, Springer.

Dube, K (2000a). *BNF Syntax of PLAN*. Dublin, Computer Science Department, School of Computing, Dublin Institute of Technology**:** *3*.

Dube, K (2000b). *PLAN Language for Specifying Clinical Test-Ordering Protocol Using the ECA Paradigm*. Dublin, Computer Science Department, School of Computing, Dublin Institute of Technology**:** *23*.

East, T D, Morris, A H, Clemmer, T, Orme, J F, Wallace, C J, Henderson, S, Sittig, D F and Gardner, R M (1990). *Development of computerised critical care protocols - a strategy that really works!* SCAMC: *564-568*.

Eder, J, Groiss, H and Nekvasil, H (1994). *A Workflow System Based on Active Databases*. Connectivity: *249-265*.

Eisenberg, J M (1985). *Physician utilisation: the state of research about physicians' practice patterns*. **23**: *461-483*.

Elmasri, R and Navathe, S B (2000). *Fundamentals of Database Systems.*, Addison-Wesley. *289-295*.

Elson, R B and Connelly, D P (1995a). *Computerised Decision Support Systems in Primary Care.* Prim Care Clin Off Pract **22**: *365-84*.

Elson, R B and Connelly, D P (1995b). *Computerised Patient Records in Primary Care: Their Role in Mediating Guideline-Driven Physician Behaviour Change*. Arch Fam Med **4**: *698-705*.

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

Fox, J and Das, S (2000). *Safe and sound*. Safe and sound, AAAI Press.

Fox, J, Johns, N and Rahmanzadeh, A (1998). *Disseminating medical knowledge: the PROforma approach.* Artificial Intelligence in Medicine **14**(No1): *157-82*.

Fox, J, Johns, N, Rahmanzadeh, A and Thomson, R (1996). *PROforma: a method and language for specifying clinical guidelines and protocols*. Medical Informatics Europe, Amsterdam.

Gatziu, E, Geppert, A and Dittrich, K R (1991). *Integrating active concepts into an object-oriented database system*. 3rd International Workshop on Database Programming Languages, Naphlion, Greece.

Geppert, A and Dittrich, K R (1993). *SAMOS: An active, object-oriented database system*. 1st Intl. Workshop on Rules in Database Systems, Edinburg,UK.

Geppert, A, Kradolfer, M and Tombros, D (1995). *Realization of Cooperative Agents using an Active Object-Oriented Database System*. Rules in Database Systems (RIDS '95), Second International Workshop, Glyfada, Athens, Greece, Springer. *327-341*.

Gietz, W and Dupree, C (2002). *Oracle9i Application Developer's Guide - Object-Relational Features, Release 2 (9.2)*. Redwood Shores, CA 94065, USA, Oracle Corporation.

Gordon, C, Herbert, I and Johnson, P (1996). *Knowledge Representation and Clinical Practice Guidelines: the DILEMMA and PRESTIGE projects.* Medical Informatics Europe, Copenhagen, IOS Press. *511-515*.

Gordon, 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.

Gordon, C, Jackson-Smale, A and Thomson, R (1994). *DILEMMA: Logic engineering in primary care, shared care and oncology (AIM Project A2005)*. Computer Methods and Programs in Biomedicine **45**: *37-39*.

Gordon, C and Veloso, M (1996). *The PRESTIGE Project: Implementing Guidelines in Healthcare*. Medical Informatics Europe 96.

Graeser, K (1994). *Active databases: Tools of the software AG on the way towards active databases*. Datenbak Rundbrief (in German) **14**: *15-16*.

Greenes, R A, Peleg, M, Boxwala, A A, Tu, S W, Patel, V L and Shortliffe, E H (2001). *Sharable Computer-Based Clinical Practice Guidelines: Rationale, Obstacles, Approaches, and Prospects.* Medinfo 2001, London, UK. *201-5*.

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

Grimson, J, Stephens, G, Jung, B, Grimson, W, Berry, D and Pardon, S (2001). *Sharing Health-Care Records over the Internet*. IEEE Internet Computing: *49-58*.

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 of Medical Informatics **52**: *3-27*.

Grossman, R M (1983). *A review of cost containment strategies for laboratory testing*. Med Care **21**: *783-802*.

Grosso, W E, Eriksson, H, Fergerson, R W, Gennari, J H, Tu, S W and Musen, M A (1999). *Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000)*, Stanford Medical Informatics, Stanford University**:** *36*.

Gryz, J (1998a). *An algorithm for query folding with functional dependencies*. 7th International Symposium on Intelligent Information Systems. *7-16*.

Gryz, J (1998b). *Query folding with inclusion dependencies*. 14th IEEE Int. Conf. on Data Engineering (ICDE'98), Orlando, Florida. *126-133*.

Guarnero, A, Marzuoli, M, Molino, G, Terenziani, P, M., T and Vanni, K (1998). *Contextual and temporal clinical guidelines*. AMIA Symp. *683-7*.

Gutleber, J, Schimak, G and Humer, H (1997). *Using Active Behaviour in Environmental Monitoring Systems*. Environmental Software Systems (IFIP TC5 WG5.11 International Symposium on Environmental Software Systems), Chapman & Hall.

Hanson, E N, Chen, I C, Dastur, R, Engel, K, Ramaswamy, V, Tan, W and Xu, C (1998). *A Flexible and Recoverable Client/Server Database Event Notification System*. VLDB Journal **7**: *12-24*.

Harvey, J N, Rizvi, K, Craney, L, Messenger, J, Shah, R and Meadows, P A (2001). *Population-based survey and analysis of trends in the prevalence of diabetic nephropathy in Type 1 diabetes*. Diabet Med **18**(12): *998-1002*.

HL7 (1999). *Arden Syntax for Medical Logic Modules.* Standards of the Health Level 7. USA.

Hripcsak, G, Clayton, P, Jenders, R, Cimino, J and Johnson, S (1996). *Design of a clinical event monitor.* Comput Biomed Res **29**: *194-221*.

Hripscak, G, Luderman, P, Pryor, T A, Wigertz, O B and Clayton, P D (1994). *Rationale for the Arden Syntax.* Computers and Biomedical Research **27**: *291-324*.

Institute of Medicine (IOM) (1992). *Guidelines for clinical practice: from development to use.* Washington, D.C., National Academy Press.

Jasper, H (1994). *Active databases for active repositories*. 10th International Conference on Data Engineering, Houston, Texas, U.S.A. *375-384*.

Jenders, R A, Hripcsak, G, Sideli, R V, DuMouchel, W, Zhang, H, Cimino, J J, Johnson, S B, Sherman, E H and Clayton, P D (1995). *Medical decision support: experience with implementing the Arden Syntax*. Annu. Symb. Comput. Appl. Med. Care: *169-173*.

Jenders, R A, Huang, H, Hripcsak, G and Clayton, P D (1998). *Evolution of a knowledge base for a clinical decision support system encoded in the Arden Syntax*. AMIA Symp. *558-62*.

Johnson, P D, Tu, S W, Booth, N, Sugden, B and Purves, I N (1999). *A guideline model for chronic disease management*. Stanford, Stanford Medical Informatics.

Johnson, P D, Tu, S W, Booth, N, Sugden, B and Purves, I N (2000). *Using scenarios in chronic disease management guidelines for primary care*. AMIA Annual Symposium, Philadelphia., Hanley and Belfus.

Jones, R, Dube, K and Wu, B (2003). *TOPME: An XML-Based Client-Server Front-End for the Distributed Management of Clinical Protocol for TOPS*. Dublin, Dublin Institute of Technology**:** *5*.

Kanouse, D E and Jacoby, I (1988). *When does Information Change Practitioners' Behaviour?* Int J Technol Health Care **4**: *27-33*.

Kawano, H, Nishio, J H and Hasegawa, T (1994). *How does knowledge discovery cooperate with active database techniques in controlling dynamic environments?* 5th Intl. Conf. on Database and Expert System Applications (DEXA'94), Athen. *370-379*.

Kotz-Dittrich, A and Simon, E (1999). *Active database systems: Expectations, Commercial Experience, and Beyond*. Active Rules in Database Systems. Paton, NW. New York, Springer-Verlag**:** *367-404*.

Kuperman, G J, Teich, J M, Tanasijevic, M J, Ma'Luf, N, Rittenberg, E, Jha, A, Fiskio, J, Winkelman, J and Bates, W (1999). *Improving response to critical laboratory results with automation: results of a randomised control trial.* JAMIA **6 No.6**: *512-22*.

Lanarkshire Diabetes Group (1999). *Local Protocol for the Implementation of the St. Vincent Declaration (SIGN Guidelines 4,9,10,12,19,25),* *http://www.nhslanarkshire.co.uk/hq/disease/pdf_files/st_vincent.pdf*. Motherwell, UK, NHS Lanarkshire**:** *11*.

Lee, J K and Sohn, M M (2003). *The eXtensible Rule Markup Language*. Communications of the ACM **46**(5): *59-64*.

Li, L and Chakravarthy, S (1999). *An agent-based approach to extending the native active capability of relational database systems*. 15th International Conference on Data Engineering. *384-391*.

Lobach, D F, Gadd, C S and Hales, J W (1997). *Structuring clinical practice guidelines in a relational database model for decision support on the Internet*. AMIA Annual Fall Symposium. *158-162*.

Lobach, D F and Hammond, W E (1994). *Development and evaluation of a computer-assisted management protocol (CAMP): improved compliance with care guidelines for diabetes mellitus*. Annual Symposium on Computer Applications in Medical Care. *787-791*.

Lopes, C V and Hursch, W L (1995). *Separation of Concerns*. Boston, M, Northeaster University, A**:** *20*.

Matimer, D, McCauley, B, Nightingale, P, Ryan, M, Peters, M and Neuberger, J (1992). *Computerised protocols for laboratory investigation and their effect on use of medical time and resouces*. Journal of Clinical Pathology **45**: *572-574*.

McDonalds, C J, Wilson, G A and McCabe, G (1980). *Physician Response to Computer Reminders*. JAMIA **244**(14): *1579-1581*.

MediLink (2003). *The MediLink Programme, http://www.cs.tcd.ie/medilink/*, Healthcare Informatics Centre, Computer Science Department, Trinity College, Dublin. **2003**.

Melton, J (2003). *Information Technology - Database Languages - SQL: Part 2, Foundation (SQL/Foundation)*, ISO/IEC & ANSI**:** *1332: 125-128*.

Miksch, S (1999). *Plan Management in the Medical Domain*. AI Communications **12**(4): *209-235*.

Mogensen, C E (2003). *Microalbuminuria and hypertension with focus on type 1 and type 2 diabetes*. J Intern Med **254**(1): *45-66*.

Motta, E (1999). *Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design*. Amsterdam, IOS Press.

Musen, M A, Gennari, J H, Eriksson, H, Tu, S W and Puerta, A R (1995). *PROTEGE II: Computer Support For Development Of Intelligent Systems From Libraries Of Components*. MEDINFO: The Eighth World Congress on Medical Informatics, Vancouver, B.C., Canada. *766-770*.

Musen, M A, Tu, S W, Das, A K and Shahar, Y (1996). *EON: A component-based approach to automation of protocol-directed therapy*. JAMIA **3**(6): *367-88*.

Musen, M A, Tu, S W and Shahar, Y (1992). *A problem-solving model for protocol-based care: from e-ONCOCIN to EON.* MEDINFO, IMIA: *519-525*.

Newell, A and Simon, H A (1972). *Human problem solving*. Englewood Cliffs, NJ, Prentice Hall.

Nykanen, P (2000). *Decision Support Systems in Healthcare*. Computer Science and Information Sciences, University of Tampere.

Ogunyemi, O, Zeng, Q and Boxwala, A A (2002). *Object-oriented guideline expression language (GELLO) specification*, Decision Systems Group, Brigham and Women's hospital, Harvard Medical School.

Ohno-Machado, L, Gennari, J H, Murphy, S, Jain, N H, Tu, S W, Oliver, D E, Pattison-Gordon, E, Greenes, R A, Shortliffe, E H and Barnett, G O (1998). *The GuideLine Interchange Format: A Model for Representing Guidelines.* JAMIA **5**(4): *357-372*.

OMG (2001). *OMG Unified Modeling Language Specification*, Object Management Group (OMG).

O'Moore, R, Groth, T, Grimson, W and Boran, G (1996). *Advanced Informatics and Telematics for Optimization of Clinical Laboratory Services*. Computer Methods and Programs in Biomedicine **50**(2): *85-206*.

OpenClinical (2001). *The Medical Knowledge Crisis and its solution Through Knowledge Management*, OpenClinical.org.

OpenClinical (2003). *Knowledge management for medical care: GUIDE, http://www.openclinical.org/gmm_guide.html*, OpenClinical,. **2003**.

Overhage, J M, Tierney, W M, Zhou, X H and MacDonald, C J (1997). *A randomised trial of 'corollary orders' to prevent errors of omission.* JAMIA **4**: *364-375*.

Owens, K T (1994). *Using Oracle7 Triggers to Implement Business Rules*, RevealNet Inc.**:** *14*.

Parnas, D L (1972). *On the Criteria to be used in Decomposing Systems into Modules*. Communications of the ACM **15**(12): *1053-1058*.

Paton, N W, Ed. (1999). *Active Rules in Database Systems*. Monographs in Computer Science. New York, Springer-Verlag. *439*.

Paton, N W and Diaz, O (1999). *Active Database Systems*. ACM Computing Surveys **31**(1): *.63-103*.

Pattison-Gordon, E, Cimino, J J, Hripcsak, G, Tu, S W, Gennari, J H, Jain, N L and Greenes, R A (1996). *Requirements of a sharable guideline representation for computer applications*. Stanford, California, USA, Stanford University.

Peleg, M, Boxwala, A A, Omolala, O, Zeng, Q, Tu, S W, Lucson, R, Bernstam, E, Ash, N, Mork, P, Ohno-Machado, L, Shortliffe, E H and Greenes, R A (2000). *GLIF3: The evolution of a guideline representation format*. AMIA Annual Symposium , LA, CA, Philadelphia, Hanley and Belfus.

Peleg, M, Tu, S M, Bury, J, Ciccarese, P, Fox, J, Greenes, R A, Hall, R, Johnson, P D, Jones, N, Kumar, A, Miksch, S, Quaglini, S, Seyfang, A, Shortliffe, E H and Stefanelli, M (2002). *Comparing Computer-Interpretable Guideline Models: A Case-Study Approach*. JAMIA **10**(1): *52-68*.

Peters, M and Broughton, P M G (1993). *The role of expert systems in improving test request patterns of clinicians*. Ann Clinical Biochem **30**: *.52-59*.

Peters, M, Broughton, P M G and Nightingale, P G (1991). *Use of Information Technology for Auditing Effective Use of Laboratory Services*. Clinical Pathology **44**: *539-542*.

Peters, M, Clarke, I R and Parekh, J e a (1991). *Automatic application of rule-based decision-support - a specialist unit investigation manager*. Current Perspectives in Healthcare Computing. Richard, B**:** *129-136.*

Porto, F A M, Carvalho, S R, Vianna e Silva, M J and Melo, R N (1999). *Persistent object synchronization with active relational databases*. Technology of Object-Oriented Languages and Systems (TOOLS 30). *53-62*.

Protocol Steering Committee (1998). *Protocol for Viral Hepatitis Testing, (http://www.healthservices.gov.bc.ca/msp/protoguides/gps/vihep.html)*. British Columbia, Canada, British Columbia Medical Association. **2004:** *6*.

Qian, X (1996). *Query folding*. 12th International Conference on Data Engineering, New Orleans, LA. *48-55*.

Quaglini, S, Stefanelli, M, Caporusso, V and Panzarasa, S (2000a). *Managing Non-Compliance in Guideline-based Careflow Systems (Poster)*. AMIA Annual Symposium.

Quaglini, S, Stefanelli, M, Cavallini, A, Micieli, G, Fassino, C and Mossa, C (2000b). *Guideline-based careflow systems*. Artif Intell Med **20**(1): *5-22*.

Quaglini, S, Stefanelli, M, Lanzola, G, Caporusso, V and Panzarasa, S (2001). *Flexible guideline-based patient careflow systems*. Artif Intell Med **22**(1): *65-80*.

Ristow, M (2004). *Neurodegenerative disorders associated with diabetes mellitus*. Journal of Molecular Medicine.

Rumbaugh, J, Jacobson, I and Grady, B (1998). *The Unified Modeling Language Reference Manual*, Addison-Wesley Pub Co. *256*.

Rumbaugh, J R, Blaha, M R, Lorensen, W, Eddy, F and Premerlani, W (1990). *Object-Oriented Modeling and Design*, Prentice-Hall. *500*.

Russell, J (2002). *Oracle9i Application Developer's Guide - Fundamentals, Release 2 (9.2)*. Redwood Shores, CA 94065, USA, Oracle Corporation.

Sailors, R M, Bradshaw, R L and East, T D (1998). *Moving the Arden Syntax Outside of the (Alert) Box: A Paradigm for Supporting Multi-Step Clinical protocols*. JAMIA, **5**(Symposium Supplement): *1071*.

Schadow, G, Russler, D C, Mead, C N and MacDonald, C J (2000). *Integrating medical information and knowledge in the HL7 RIM*. AMIA Ann Fall Symp. *764-768*.

Scherpbier, H (1995). *CT Study With Contrast in Patients With Renal Failure (a sample Medical Logic Module (MLM))*, MLM Library, Columbia-Presbyterian Medical Center, New York City. **2004**.

Schreiber, G, Akkermans, H, Anjewierden, A, de Hoog, R, Shadbolt, N, Van de Velde, W and Wielinga, B (1999). *Knowledge Engineering and Management: The CommonKADS Methodology*, The MIT Press. *471*.

Schwiderski, S (1996). *Monitoring the Behavior of Distributed Systems*. PhD Book, Computer Science. Cambridge, Selwyn College, Computer Lab, University of Cambridge.

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

Sherman, E H, Hripcsak, G, Starren, J, Jenders, R A and Clayton, P (1995). *Using intermediate states to improve the ability of the Arden Syntax to implement care plans and reuse knowledge*. Symp Comput Appl Med Care. *238-42*.

Shiffman, R N (1997). *Representation of Clinical Practice Guidelines in Conventional and Augmented Decision Tables*. JAMIA **4**(5): *382-393*.

Shortliffe, E H, Axline, S G, Buchanan, B G, Merigan, T C and Cohen, S N (1973). *An Artificial Intelligence Program to Advise Physicians Regarding Antimicrobial Therapy*. Computers and Biomedical Research **6**: *544-560*.

Shortliffe, E H, Scott, C A and Bischoff, M B (1981). *ONCOCIN: An expert system for oncology protocol management*. 7th International Joint Conference on Artificial Intelligence. *876--881*.

Simon, E and Kotz-Dittrich, A (1995). *Promises and realities of active database systems*. 21st VLDB Conference, Zurich, Switzerland. *642-653*.

Smith, B J and McNeely, M D D (1999). *The influence of an expert system for test ordering and interpretation on laboratory testing*. Clinical Chemistry **45**(8): *1168-75*.

Starren, J and Xie, G (1994). *Comparison of Three Knowledge Representation Formalisms for Encoding the NCEP Cholesterol Guidelines*. Eighteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC, Hanley & Belfus. *792-796*.

Stonebraker, M, Hanson, E N and Potamianos, S (1988). *The POSTGRES Rule Manager*. IEEE Transactions on Software Engineering **14**(7): *897 - 907*.

Tagg, R and Lelatanavit, W (1998). *Using an active DBMS to implement a workflow engine*. International Database Engineering and Applications Symposium (IDEAS). *286-295*.

Terenziani, P, Mastromonaco, F, Molino, G and Torchio, M (2000). *Executing clinical guidelines: temporal issues.* AMIA Symp. *848-52*.

Terenziani, P, Molino, G and Torchio, M (2001). *A modular approach for representing and executing clinical guidelines*. Artif Intell Med. **23**(3): *249-76*.

Thomson, R (1995). *DILEMMA: Decision Support in Primary Care, Oncology and Shared Care*, IOS Press.

Tu, S W, Johnson, P D and Musen, M A (2001). *A Typology for Modeling Processes in Clinical Guidelines and Protocols*. AMIA Annual Symposium, San Antonio.

Tu, S W and Musen, M A (1999). *A Flexible Approach to Guideline Modeling*. AMIA Annual Symposium, Washington, D.C. *420-424*.

Tu, S W and Musen, M A (2000). *From Guideline Modeling to Guideline Execution: Defining Guideline-Based Decision-Support Services*. AMIA Annual Symposium, Los Angeles, CA, Hanley & Belfus Inc. *863-867*.

Tu , S W and Musen, M A (2001). *Modeling Data and Knowledge in the EON Guideline Architecture*. MedInfo, London, UK.

Tu, S W and Musen, M A (2001). *Representation Formalisms and Computational Methods for Modeling Guideline-Based Patient Care*. Computer-Based Support for Clinical Guidelines and Protocols-Proceedings of EWGLP 2000. Heller, B, Loffler, M, Musen, M and Stefanelli, M. Amsterdam, IOS Press. **83:** *115-132*.

Ullman, J D and Widom , J (2001). *A First Course in Database Systems*, Prentice Hall. *528*.

van Walraven, C and Naylor, C D (1998). *Do we know what inappropriate laboratory utilisation is? A systematic review of laboratory clinical audits*. JAMIA **280**: *550-558*.

van Wijk, M A M, Bohnen, A M and van der Lei, J (1999). *Analysis of the practice guidelines of the Dutch College of General Practitioners with respect to the use of blood tests*. JAMIA **6**(4): *322-331*.

van Wijk, M A M, Mosseveld, M and van der Lei, J (1999). *Design of a decision support system for test ordering in General Practice: choices and decisions to make*. Methods of Information in Medicine **38**: *355-61*.

Wang, D, Peleg, M, Tu, S W, Boxwalla, A A, Greenes, R A, Patel, V L and Shortliffe, E H (2002). *Representation Primitives, Process Models and Patient Data in Computer-Interpretable Clinical Practice Guidelines: A Literature Review of Guideline Representation Models*. International Journal of Medical Informatics **68**(1-3): *59-70*.

Widom, J and Ceri, S, Eds. (1996). *Active Database Systems: Triggers and Rules for Advanced Database Processing*. San Francisco, USA, Morgan Kaufmann.

Wu, B (1996). *Specifying and enforcing integrity constraints in object oriented databases*. PhD Book, Computer Science. Manchester, Department of Computation, UMIST.

Wu, B (1998). *PLAN : the framework and its language for modelling clinical test request protocols*. Dublin, Department of Computer Science, DIT.

Wu, B and Dube, K (1998). *The BNF definition of PLAN : the framework and its language for modelling clinical test request protocols*. Dublin, Department of Computer Science, DIT.

Wu, B and Dube, K (2001). *PLAN: a Framework and Specification Language with an Event-Condition-Action (ECA) Mechanism for Clinical Test Request Protocols*. 34th Hawaii International Conference on System Sciences (HICSS-34):: the Mini-Track in Information Technology in Healthcare, Maui, Hawaii, IEEE Computer Society, Los Alamitos, California. *140*.

APPENDIX

A. The BNF Syntax of PLAN

<protocol> ::= @PROTOCOL@<protocol_body>@END PROTOCOL@
<protocol_body> ::= <protocol_header>#SCHEDULE_SET#<schedule_list>#END SCHEDULE_SET#<protocol_rule_set>
<protocol_header> ::= <protocol_name>;<description>;<creator>;<category>;
<protocol_name> ::= <identifier>
<description> ::= DESRIPTION: <descriptive_text>
<descriptive_text> ::= string
<category> ::= CATEGORY: <category-name>
<creator> ::= CREATOR: <creator-name>
<schedule-list> ::= <schedule> | <schedule>,<schedule-list>
<schedule> ::= ^SCHEDULE^<schedule_body>^END SCHEDULE^
<schedule_body> ::= <schedule_header>;<schedule_rule_list>;
<schedule_header> ::= <schedule_name>;[<initial_state>;]<entry_criteria>
<initial_state> ::= INITIAL_STATE: {ACTIVE | INACTIVE}
<schedule_name> ::= <identifier>
<entry-criteria> ::= ENTRY_CRITERIA,CONDITION: <condition_spec>[,<description>];
<condition_spec> ::= <comparison_attribute>%<attribute_entity>%<value_source>[%<source_name>]<comparison_operator><right_value><right_value_type>
<schedule_rule_list> ::= <schedule_rule>|<schedule_rule>;<schedule_rule_list>
<schedule_rule> ::= <static_rule>|<dynamic_rule>
<static-rule> ::= <rule_header>,<time_events>,<action_spec>
<rule_header> ::= {STATIC_RULE | RULE} <rule_name>,[<description>,][<initial_state>,]
<time_events> ::= <ref_point>,<start_point_spec>,<end_point_spec>,<interval_spec>
<ref_point> ::= FROM: <identifier> | <domain_term>
<start_point_spec>::= STARTING: <time_length> <time_unit>
<end_point_spec>::= ENDING: <time_length> <time_unit>
<interval_spec> ::= ON EVERY: <time_length> <time_unit>
<time_length> ::= integer
<time_unit> ::= YEAR|YEARS|MONTH|MONTHS|WEEK|WEEKS|DAY|DAYS|HOUR|HOURS|MINUTE|MINUTES|SECOND|SECONDS
<action_spec> ::= DO: <action> ([<parameter_list>])
<action> ::= ORDER|ISSUE_ALERT|SEND_MAIL|...
<parameter_list> ::= <parameter> | <parameter>,<parameter_list>
<parameter> ::= <string_parameter> | <number_parameter>
<string_parameter> ::= 'STRING'
<number_parameter> ::= DOUBLE | INTEGER
<dynamic-rule> ::= <rule-header>,[<description>,]<event_spec>,<condition_spec>,<action_spec>;
<event_spec> ::= On: <event> ([<parameter_list>])
<event> ::= RESULT_ARRIVAL | DISCHARGE | CHECK_IN | ...
<protocol_rule_set> ::= ~RULE_SET~ <protocol_rule_list> ~END RULE_SET~
<protocol_rule_list> ::= <dynamic_rule> | <dynamic_rule>;<protocol_rule_list>

B. The Relational Schema for the TOPS Database in Oracle SQL

B.1. The TOPS protocol specification database schema

```
CREATE TABLE PR_PROTOCOL
  (
   ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128)  NOT NULL UNIQUE,
   DESCRIPTION VARCHAR2(128)  NULL,
   DATE_CREATED DATE  NOT NULL,
   CREATOR_ID NUMBER(38) NOT NULL,
   DATE_AUTHORISED DATE NULL,
   AUTHORISER_ID NUMBER(38) NULL,
   CATEGORY_ID NUMBER(38)  NOT NULL UNIQUE,
   SCHEDULES NUMBER(38) NOT NULL,
   PROTOCOL_RULES NUMBER(38) NOT NULL,
   CONSTRAINT PK_PR_PROTOCOL PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_STATIC_RULE
  (
   ID NUMBER(38)  NOT NULL,
   ZERO_TIME_REF_TERM VARCHAR2(128)  NOT NULL,
   START_TIME NUMBER(38)  NOT NULL,
   END_TIME NUMBER(38)  NOT NULL,
   INTERVAL NUMBER(38)  NOT NULL
,  CONSTRAINT PK_PR_STATIC_RULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_PROTOCOL_RULE
  (
   ID NUMBER(38)  NOT NULL
,  CONSTRAINT PK_PR_PROTOCOL_RULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_SCHEDULE
  (
   ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128)  NOT NULL UNIQUE,
   DESCRIPTION VARCHAR2(228)  NULL,
   CREATOR_ID NUMBER(38) NOT NULL,
   DATE_CREATED DATE  NOT NULL
,  CONSTRAINT PK_PR_SCHEDULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_DYNAMIC_RULE
  (
   ID NUMBER(38)  NOT NULL,
   EVENT_ID NUMBER(38)  NOT NULL,
   EVENT_PARAMETERS VARCHAR2(128)  NULL,
   RULE_TYPE VARCHAR2(128)  NOT NULL,
   CONSTRAINT PK_PR_DYNAMIC_RULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_RULE
  (
   ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128)  NOT NULL UNIQUE,
   DESCRIPTION VARCHAR2(228)  NULL,
   RULE_TYPE VARCHAR2(128)  NOT NULL,
   CREATOR_ID NUMBER(38) NOT NULL,
   DATE_CREATED DATE  NOT NULL,
   CONSTRAINT PK_PR_RULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_SCHEDULE_RULE
  (
   ID NUMBER(38)  NOT NULL
,  CONSTRAINT PK_PR_SCHEDULE_RULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_CONDITION
  (
   ID NUMBER(38)  NOT NULL,
   CODE VARCHAR2(50) NOT NULL UNIQUE,
   ATTRIBUTE VARCHAR2(128) NOT NULL,
   ATTRIBUTE_ENTITY VARCHAR(128) NOT NULL,
   SOURCE_TYPE VARCHAR2(128) NOT NULL,
   SOURCE_NAME VARCHAR2(128) NULL,
   RIGHT_VALUE VARCHAR2(128) NULL,
   DATA_TYPE VARCHAR2(40),
   COMPARATOR VARCHAR2(20) NULL,
   DESCRIPTION VARCHAR2(228)  NULL,
   DATE_CREATED DATE  NOT NULL
,  CONSTRAINT PK_PR_CONDITION PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_ACTION
  (
   ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128)  NOT NULL UNIQUE,
   DESCRIPTION VARCHAR2(228)  NULL,
   DATE_CREATED DATE  NOT NULL
,  CONSTRAINT PK_PR_ACTION PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_EVENT
  (
   ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128)  NOT NULL UNIQUE,
   DESCRIPTION VARCHAR2(228)  NULL,
   DATE_CREATED DATE  NOT NULL,
   CONSTRAINT PK_PR_EVENT PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_PROTOCOL_SCHEDULE
  (
   PROTOCOL_ID NUMBER(38)  NOT NULL,
   SCHEDULE_ID NUMBER(38)  NOT NULL
,  CONSTRAINT PK_PR_PROTOCOL_SCHEDULE PRIMARY KEY (PROTOCOL_ID, SCHEDULE_ID)
  ) ;

CREATE TABLE PR_RULE_CONDITION
  (
   RULE_ID NUMBER(38)  NOT NULL,
   CONDITION_ID NUMBER(38)  NOT NULL
,      CONSTRAINT PK_PR_RULE_CONDITION PRIMARY KEY (RULE_ID,
CONDITION_ID)
  ) ;

CREATE TABLE PR_RULE_ACTION
  (
   RULE_ID NUMBER(38)  NOT NULL,
   ACTION_ID NUMBER(38)  NOT NULL,
   ACTION_PARAMETERS VARCHAR2(1000)  NULL,
  CONSTRAINT PK_PR_RULE_ACTION PRIMARY KEY (RULE_ID, ACTION_ID)
  ) ;

CREATE TABLE PR_SCHEDULE_SRULE
  (
   SCHEDULE_ID NUMBER(38)  NOT NULL,
   RULE_ID NUMBER(38)  NOT NULL
,    CONSTRAINT PK_PR_SCHEDULE_SRULE PRIMARY KEY (SCHEDULE_ID,
RULE_ID)
  ) ;

CREATE TABLE PR_PROTOCOL_PRULE
  (
   PROTOCOL_ID NUMBER(38)  NOT NULL,
   RULE_ID NUMBER(38)  NOT NULL
,    CONSTRAINT PK_PR_PROTOCOL_PRULE PRIMARY KEY (PROTOCOL_ID,
RULE_ID)
  ) ;

CREATE TABLE PR_PROTOCOL_SRULE
  (
   PROTOCOL_ID NUMBER(38)  NOT NULL,
   RULE_ID NUMBER(38)  NOT NULL,
   CONSTRAINT  PK_PR_PROTOCOL_SRULE  PRIMARY  KEY  (PROTOCOL_ID,
RULE_ID)
  ) ;


CREATE TABLE PR_CRITERIA
  (
   ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128) NOT NULL UNIQUE,
   CRITERIA_TYPE VARCHAR2(50) NOT NULL,
   DESCRIPTION VARCHAR2(400) NULL,
   CONSTRAINT PK_PR_CRITERIA PRIMARY KEY (ID)
  ) ;

CREATE TABLE PR_CRITERIA_CONDITION
   (
   CRITERIA_ID NUMBER(38) NOT NULL,
   CONDITION_ID NUMBER(38) NOT NULL,
   CONSTRAINT  PK_CRITERIA_CONDITION  PRIMARY  KEY
(CRITERIA_ID,CONDITION_ID)
   );

CREATE TABLE PR_SCHEDULE_CRITERIA
   (
   SCHEDULE_ID NUMBER(38)  NOT NULL,
   CRITERIA_ID NUMBER(38)  NOT NULL,
   CONSTRAINT  PK_PR_CASE_SWITCH  PRIMARY  KEY  (SCHEDULE_ID,
CRITERIA_ID)
   ) ;

CREATE TABLE PR_SCHEDULE_DRULE
(
SCHEDULE_ID NUMBER(38)  NOT NULL,
RULE_ID NUMBER(38)  NOT NULL,
CONSTRAINT  PK_PR_SCHEDULE_DRULE  PRIMARY  KEY  (SCHEDULE_ID,
RULE_ID)
   );

CREATE TABLE TOPS.PR_COMPOSITE_CONDITION(
ID NUMBER(38) NOT NULL,
COND1_ID NUMBER(38),
COND2_ID NUMBER(38),
COMP1_ID NUMBER(38),
COMP2_ID NUMBER(38),
COMPARATOR VARCHAR2(10),
CONSTRAINT PK_PR_COMPOSITE_CONDITION PRIMARY KEY(ID)
);

CREATE TABLE TOPS.PR_STATE_ACTION(
STATE_ID NUMBER(38) NOT NULL,
ACTION_ID NUMBER(38) NOT NULL,
ACTION_PARAMETERS VARCHAR2(300),
CONSTRAINT  PK_TOPS_STATE_ACTION  PRIMARY  KEY  (STATE_ID,
ACTION_ID)
);

ALTER TABLE PR_PROTOCOL_SRULE ADD (
 CONSTRAINT FK_PROTOCOL_SRULE_protocol
     FOREIGN KEY (PROTOCOL_ID)
         REFERENCES TOPS.PR_PROTOCOL (ID)
             ON DELETE CASCADE
);

ALTER TABLE PR_PROTOCOL_SRULE ADD (
 CONSTRAINT FK_PROTOCOL_SRULE_srule
     FOREIGN KEY (RULE_ID)
         REFERENCES TOPS.PR_STATIC_RULE (ID)
             ON DELETE CASCADE
```

```
);


ALTER TABLE PR_STATE_ACTION ADD (
 CONSTRAINT FK_TOPS_STATE_ACTION_STATE
    FOREIGN KEY (STATE_ID)
       REFERENCES TOPS_PATIENT_STATE (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_STATE_ACTION  ADD (
 CONSTRAINT FK_PR_TOPS_STATE_ACTION_ACTION
    FOREIGN KEY (ACTION_ID)
       REFERENCES PR_ACTION (ID)
          ON DELETE CASCADE
);

ALTER TABLE TOPS.PR_COMPOSITE_CONDITION ADD (
 CONSTRAINT FK_PR_COMPO_COND_COND_1
    FOREIGN KEY (COND1_ID)
       REFERENCES TOPS.PR_CONDITION (ID)
          ON DELETE CASCADE
);
ALTER TABLE TOPS.PR_COMPOSITE_CONDITION ADD (
 CONSTRAINT FK_PR_COMPO_COND_COND_2
    FOREIGN KEY (COND2_ID)
       REFERENCES TOPS.PR_CONDITION (ID)
          ON DELETE CASCADE
);

ALTER TABLE TOPS.PR_COMPOSITE_CONDITION ADD (
 CONSTRAINT FK_PR_COMPO_COND_SELF_1
    FOREIGN KEY (COMP1_ID)
       REFERENCES TOPS.PR_COMPOSITE_CONDITION (ID)
          ON DELETE CASCADE
);
ALTER TABLE TOPS.PR_COMPOSITE_CONDITION ADD (
 CONSTRAINT FK_PR_COMPO_COND_SELF_2
    FOREIGN KEY (COMP2_ID)
       REFERENCES TOPS.PR_COMPOSITE_CONDITION (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_PROTOCOL ADD (
 CONSTRAINT FK_PR_PROTOCOL_CATEGORY
    FOREIGN KEY (CATEGORY_ID)
       REFERENCES TOPS_CATEGORIES (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_PROTOCOL ADD (
 CONSTRAINT FK_PR_PROTOCOL_CREATOR
    FOREIGN KEY (CREATOR_ID)
       REFERENCES TOPS_CLINICIANS (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_PROTOCOL ADD (
 CONSTRAINT FK_PR_PROTOCOL_AUTHORISER
    FOREIGN KEY (AUTHORISER_ID)
       REFERENCES TOPS_CLINICIANS (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_STATIC_RULE ADD (
 CONSTRAINT FK_PR_STATIC_RULE_PR_RULE
    FOREIGN KEY (ID)
       REFERENCES PR_RULE (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_PROTOCOL_RULE ADD (
 CONSTRAINT FK_PR_PROTOCOL_RULE_DYNAM_RUL
    FOREIGN KEY (ID)
       REFERENCES PR_DYNAMIC_RULE (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_DYNAMIC_RULE ADD (
 CONSTRAINT FK_PR_DYNAMIC_RULE_EVENT
    FOREIGN KEY (EVENT_ID)
       REFERENCES PR_EVENT (ID));

ALTER TABLE PR_DYNAMIC_RULE ADD (
 CONSTRAINT FK_PR_DYNAMIC_RULE_PR_RULE
    FOREIGN KEY (ID)
       REFERENCES PR_RULE (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_SCHEDULE_RULE ADD (
 CONSTRAINT FK_PR_SCHEDULE_RULE_DYN_R
    FOREIGN KEY (ID)
       REFERENCES PR_DYNAMIC_RULE (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_PROTOCOL_SCHEDULE ADD (
 CONSTRAINT FK_PR_PROTOCOL_SCHEDULE_PR_PRO
    FOREIGN KEY (PROTOCOL_ID)
       REFERENCES PR_PROTOCOL (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_PROTOCOL_SCHEDULE ADD (
 CONSTRAINT FK_PR_PROTOCOL_SCHEDULE_PR_SCH
    FOREIGN KEY (SCHEDULE_ID)
       REFERENCES PR_SCHEDULE (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_RULE_CONDITION ADD (
 CONSTRAINT FK_PR_RULE_CONDITION_DYNA_RU
    FOREIGN KEY (RULE_ID)
       REFERENCES PR_DYNAMIC_RULE (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_RULE_CONDITION ADD (
 CONSTRAINT FK_PR_RULE_CONDITION_PR_CONDIT
    FOREIGN KEY (CONDITION_ID)
       REFERENCES PR_CONDITION (ID)
          ON DELETE CASCADE
) ;

ALTER TABLE PR_RULE_ACTION ADD (
 CONSTRAINT FK_PR_RULE_ACTION_RULE
    FOREIGN KEY (RULE_ID)
       REFERENCES PR_RULE (ID)
          ON DELETE CASCADE
) ;

ALTER TABLE PR_RULE_ACTION ADD (
 CONSTRAINT FK_PR_RULE_ACTION_PR_ACTION
    FOREIGN KEY (ACTION_ID)
       REFERENCES PR_ACTION (ID)
          ON DELETE CASCADE
) ;

ALTER TABLE PR_SCHEDULE_SRULE ADD (
 CONSTRAINT FK_PR_SCHEDULE_SRULE_PR_SCHEDU
    FOREIGN KEY (SCHEDULE_ID)
       REFERENCES PR_SCHEDULE (ID)
          ON DELETE CASCADE
) ;

ALTER TABLE PR_SCHEDULE_SRULE ADD (
   CONSTRAINT FK_PR_SCHEDULE_SRULE_PR_STATIC
      FOREIGN KEY (RULE_ID)
         REFERENCES PR_STATIC_RULE (ID)
            ON DELETE CASCADE
) ;

ALTER TABLE PR_PROTOCOL_PRULE ADD (
 CONSTRAINT FK_PR_PROTOCOL_PRULE_PR_PROTOC
    FOREIGN KEY (PROTOCOL_ID)
       REFERENCES PR_PROTOCOL (ID)
          ON DELETE CASCADE
) ;

ALTER TABLE PR_PROTOCOL_PRULE ADD (
 CONSTRAINT FK_PR_PROTOCOL_PRULE_PR_PROTO1
    FOREIGN KEY (RULE_ID)
       REFERENCES PR_PROTOCOL_RULE (ID)
          ON DELETE CASCADE
) ;

ALTER TABLE PR_SCHEDULE_DRULE ADD (
 CONSTRAINT FK_PR_SCHEDULE_DRULE_PR_SCHEDU
    FOREIGN KEY (SCHEDULE_ID)
       REFERENCES PR_SCHEDULE (ID)
          ON DELETE CASCADE
) ;

ALTER TABLE PR_SCHEDULE_DRULE ADD (
 CONSTRAINT FK_PR_SCHEDULE_DRULE_PR_DYNAMI
    FOREIGN KEY (RULE_ID)
       REFERENCES PR_DYNAMIC_RULE (ID)
          ON DELETE CASCADE
) ;
ALTER TABLE PR_SCHEDULE ADD (
 CONSTRAINT FK_PR_SCHEDULE_CREATOR
    FOREIGN KEY (CREATOR_ID)
       REFERENCES TOPS_CLINICIANS (ID)
          ON DELETE CASCADE
);
ALTER TABLE PR_RULE ADD (
 CONSTRAINT FK_PR_RULE_CREATOR
    FOREIGN KEY (CREATOR_ID)
       REFERENCES TOPS_CLINICIANS (ID)
          ON DELETE CASCADE
);


ALTER TABLE PR_CRITERIA_CONDITION ADD (
 CONSTRAINT FK_CRITERIA_CONDITION_COND
    FOREIGN KEY (CONDITION_ID)
       REFERENCES PR_CONDITION (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_CRITERIA_CONDITION ADD (
 CONSTRAINT FK_CRITERIA_CONDITION_CRI
    FOREIGN KEY (CRITERIA_ID)
       REFERENCES PR_CRITERIA (ID)
          ON DELETE CASCADE
);


ALTER TABLE PR_SCHEDULE_CRITERIA ADD (
 CONSTRAINT FK_PR_SCHEDULE_CRITERIA_1
    FOREIGN KEY (CRITERIA_ID)
       REFERENCES PR_CRITERIA (ID)
          ON DELETE CASCADE
);

ALTER TABLE PR_SCHEDULE_CRITERIA ADD (
 CONSTRAINT FK_PR_SCHEDULE_CRITERIA_2
    FOREIGN KEY (SCHEDULE_ID)
       REFERENCES PR_SCHEDULE (ID)
          ON DELETE CASCADE
);
```

B.2. The TOPS patient plan database schema

```
CREATE TABLE PL_PLAN
  (
   ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128)  NOT NULL UNIQUE,
   PATIENT_ID NUMBER(38)  NOT NULL,
   PROTOCOL_ID NUMBER(38)  NOT NULL,
```

```
   DESCRIPTION VARCHAR2(200)  NULL,
   DATE_CREATED DATE  NOT NULL,
   CURRENT_STATE  VARCHAR2(128)  NOT NULL,
   STATE_CHANGE_DATE DATE  NOT NULL,
   CONSTRAINT PK_PL_PLANS PRIMARY KEY (ID)
  ) ;

CREATE TABLE PL_SCHEDULE
  (
   ID NUMBER(38)  NOT NULL,
   PLAN_ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128)  NOT NULL UNIQUE,
   DESCRIPTION VARCHAR2(228)  NULL,
   END_DATE DATE  NOT NULL,
   DATE_CREATED DATE  NOT NULL,
   CURRENT_STATE  VARCHAR2(128)  NOT NULL,
   STATE_CHANGE_DATE DATE  NOT NULL,
   CONSTRAINT PK_PL_SCHEDULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PL_RULE
  (
   ID NUMBER(38)  NOT NULL,
   NAME VARCHAR2(128)  NOT NULL UNIQUE,
   RULE_TYPE VARCHAR2(60) NOT NULL,
   DESCRIPTION VARCHAR2(200)  NULL,
   DATE_CREATED DATE NOT NULL,
   CURRENT_STATE  VARCHAR2(128)  NOT NULL,
   STATE_CHANGE_DATE DATE  NOT NULL,
   CONSTRAINT PK_PL_RULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PL_RULE_NAME(
   RULE_ID NUMBER(38) NOT NULL,
   USRNAME VARCHAR2(128) NOT NULL,
   SYSNAME VARCHAR2(128) NOT NULL UNIQUE
);

CREATE TABLE PL_STATIC_RULE
  (
   ID NUMBER(38)  NOT NULL,
   SCHEDULE_ID NUMBER(38)  NOT NULL,
   START_DATE DATE  NOT NULL,
   END_DATE DATE  NOT NULL,
   INTERVAL NUMBER(38)  NOT NULL,
   CONSTRAINT PK_PL_STATIC_RULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PL_DYNAMIC_RULE
  (
   ID NUMBER(38)  NOT NULL,
   PLAN_ID NUMBER(38)  NOT NULL,
   CONSTRAINT PK_PL_DYNAMIC_RULE PRIMARY KEY (ID)
  ) ;

CREATE TABLE PL_RULE_TRIGGER
(
TRIGGER_NAME VARCHAR2(150) NOT NULL,
RULE_ID NUMER(38) NOT NULL,
CONSTRAINT PK_PL_RULE_TRIGGER PRIMARY KEY(TRIGGER_NAME)
);

CREATE TABLE TOPS.PL_REQUEST
(
ID NUMBER(38) NOT NULL,
AGENT VARCHAR2(128) NOT NULL,
MRN VARCHAR2(128)  NOT NULL,
PROTOCOL VARCHAR2(128)  NOT NULL,
ACTIVITY_ID VARCHAR2(10) NOT NULL,
DATE_REQUESTED DATE NOT NULL,
CONSTRAINT PK_PL_REQUEST PRIMARY KEY (ID)
);

CREATE TABLE TOPS.PL_PLAN_PROTOCOL_RULE
(
PL_RULE_ID NUMBER(38) NOT NULL,
PR_RULE_ID NUMBER(38) NOT NULL,
CONSTRAINT PK_PL_PLAN_PROTOCOL_RULE PRIMARY KEY (PL_RULE_ID, PR_RULE_ID)
);

CREATE TABLE PL_PLAN_SNAPSHOT
(
PLAN_ID NUMBER(38) NOT NULL,
PLAN_NAME VARCHAR(300) NOT NULL,
RULE_ID NUMBER(38) NOT NULL,
RULE_NAME VARCHAR2(300) NOT NULL,
RULE_TYPE VARCHAR2(300) NOT NULL,
RULE_STATE VARCHAR2(300) NOT NULL,
SNAP_TIME TIMESTAMP NOT NULL,
CONSTRAINT PK_PL_PLAN_SNAPSHOT PRIMARY KEY (PLAN_ID, RULE_ID, SNAP_TIME)
);

CREATE TABLE TOPS.TOPS_STATE_ACTION
(
STATE_ID NUMBER(38) NOT NULL,
ACTION_ID NUMBER(38) NOT NULL,
ACTION_PARAMETERS VARCHAR2(300),
CONSTRAINT PK_TOPS_STATE_ACTION PRIMARY KEY (STATE_ID, ACTION_ID)
);


ALTER TABLE TOPS_STATE_ACTION ADD (
 CONSTRAINT FK_TOPS_STATE_ACTION_STATE
    FOREIGN KEY (STATE_ID)
       REFERENCES TOPS_PATIENT_STATE (ID)
          ON DELETE CASCADE
);

ALTER TABLE TOPS_STATE_ACTION  ADD (
 CONSTRAINT FK_PR_TOPS_STATE_ACTION_ACTION
    FOREIGN KEY (ACTION_ID)
       REFERENCES PR_ACTION (ID)
          ON DELETE CASCADE
);

ALTER TABLE TOPS.PL_PLAN_SNAPSHOT ADD (
   CONSTRAINT FK_PL_PLAN_SNAPSHOT
      FOREIGN KEY (PLAN_ID)
         REFERENCES PL_PLAN (ID))  ;

ALTER TABLE TOPS.PL_PLAN_SNAPSHOT ADD (
   CONSTRAINT FK_PL_RULE_SNAPSHOT
      FOREIGN KEY (RULE_ID)
         REFERENCES PL_RULE (ID))  ;

ALTER TABLE TOPS.PL_RULE_TRIGGER ADD (
   CONSTRAINT FK_PL_RULE_TRIGGER_TR
      FOREIGN KEY (TRIGGER_NAME)
         REFERENCES USER_TRIGGERS (TRIGGER_NAME))  ;

ALTER TABLE TOPS.PL_RULE_TRIGGER ADD (
   CONSTRAINT FK_PL_RULE_TRIGGER_RL
      FOREIGN KEY (RULE_ID)
         REFERENCES PL_RULE (ID))  ;

ALTER TABLE TOPS.PL_PLAN_PROTOCOL_RULE ADD (
   CONSTRAINT FK_PL_PLAN_PROTOCOL_RULE_PL
      FOREIGN KEY (PL_RULE_ID)
         REFERENCES PL_RULE (ID))  ;

ALTER TABLE TOPS.PL_PLAN_PROTOCOL_RULE ADD (
   CONSTRAINT FK_PL_PLAN_PROTOCOL_RULE_PR
      FOREIGN KEY (PR_RULE_ID)
         REFERENCES PR_RULE (ID))  ;

ALTER TABLE PL_STATIC_RULE ADD (
   CONSTRAINT FK_PL_STATIC_RULE_SCHEDULE
      FOREIGN KEY (SCHEDULE_ID)
         REFERENCES PL_SCHEDULE (ID))  ;

ALTER TABLE PL_STATIC_RULE ADD (
   CONSTRAINT FK_PL_STATIC_RULE_RULE
      FOREIGN KEY (ID)
         REFERENCES PL_RULE (ID))  ;

ALTER TABLE PL_SCHEDULE ADD (
   CONSTRAINT FK_PL_SCHEDULE_PLAN
      FOREIGN KEY (PLAN_ID)
         REFERENCES PL_PLAN (ID)
            ON DELETE CASCADE);

ALTER TABLE PL_DYNAMIC_RULE ADD (
   CONSTRAINT FK_PL_DYNAMIC_RULE_PL_PLAN
      FOREIGN KEY (PLAN_ID)
         REFERENCES PL_PLAN (ID))  ;

ALTER TABLE PL_DYNAMIC_RULE ADD (
   CONSTRAINT FK_PL_DYNAMIC_RULE_RULE
      FOREIGN KEY (ID)
         REFERENCES PL_RULE(ID))  ;

ALTER TABLE PL_PLAN ADD (
   CONSTRAINT FK_PL_PLAN_TOPS_PATIENTS
      FOREIGN KEY (PATIENT_ID)
         REFERENCES TOPS_PATIENTS (ID)
            ON DELETE CASCADE)  ;

ALTER TABLE PL_PLAN ADD (
   CONSTRAINT FK_PL_PLAN_PROTOCOL
      FOREIGN KEY (PROTOCOL_ID)
         REFERENCES PR_PROTOCOL (ID)
            ON DELETE CASCADE)  ;
```

B.3. The TOPS patient database

```
CREATE TABLE TOPS.TOPS_CLINICIANS
(
ID NUMBER(10)  NOT NULL,
FIRST_NAME VARCHAR2(20)  NOT NULL,
SURNAME VARCHAR2(20)  NOT NULL,
ADDRESS VARCHAR2(200)  NOT NULL,
TELEPHONE NUMBER(20)  NOT NULL,
SPECIALTY VARCHAR2(50)  NOT NULL,
PASSWORD VARCHAR2(50)  NOT NULL,
CONSTRAINT PK_TOPS_CLINICIANS PRIMARY KEY (ID)
   );

CREATE TABLE TOPS_CATEGORIES
(
ID NUMBER(10)  NOT NULL,
NAME VARCHAR2(30)  NOT NULL,
DESCRIPTION VARCHAR2(255)  NOT NULL,
CREATOR VARCHAR2(30) NOT NULL,
DATE_CREATED DATE  NOT NULL,
CONSTRAINT PK_CATEGORY PRIMARY KEY (ID)
);

CREATE UNIQUE INDEX I_PK_CATEGORY ON TOPS_CATEGORIES (ID ASC);

CREATE TABLE TOPS.TOPS_PATIENTS
(
ID NUMBER(10)  NOT NULL,
MRN VARCHAR2(50)  NOT NULL UNIQUE,
FORENAME VARCHAR2(50)  NOT NULL,
FIRSTNAME VARCHAR2(50)  NOT NULL,
DOB DATE  NOT NULL,
SEX VARCHAR2(20)  NOT NULL,
CATEGORY_ID NUMBER(10)  NOT NULL,
TELEPHONE NUMBER(30)  NULL,
ADDRESS VARCHAR2(500)  NOT NULL,
ENTRY_DATE DATE  NULL,
```

```
CONSTRAINT PK_TOPS_PATIENTS PRIMARY KEY (ID)
);

CREATE TABLE TOPS.TOPS_ADMISSION
(
ID NUMBER(38) NOT NULL,      PATIENT_ID NUMBER(38) NOT NULL,
ADM_DATE DATE NOT NULL,
NOTES VARCHAR2(200),
CONTRAINT PRIMARY KEY(ID,ADM_DATE, PATIENT_ID)
);

CREATE TABLE TOPS.TOPS_DISCHARGE
(
ID NUMBER(38) NOT NULL,
PATIENT_ID NUMBER(38) NOT NULL,
DCG_DATE DATE NOT NULL,
NOTES VARCHAR2(200),
CONTRAINT PRIMARY KEY(ID,PATIENT_ID,DCG_DATE)
);

CREATE TABLE TOPS.TOPS_DIAGNOSTIC_HISTORY
(
ID NUMBER(38),
PATIENT_ID NUMBER(38) NOT NULL,
PROBLEM VARCHAR2(30) NOT NULL,
PREV_DIAGNOSIS VARCHAR2(50) NOT NULL,
DIAGNOSIS_DATE DATE,
CONSTRAINT PK_PATIENT_HISTORY PRIMARY KEY(ID)
);

CREATE TABLE TOPS.TOPS_DIAGNOSIS
(
ID NUMBER(38),
PATIENT_ID NUMBER(38) NOT NULL,
CLINICAL_PROBLEM VARCHAR2(100) NOT NULL,
DIAGNOSIS VARCHAR2(50) NOT NULL,
DIAGNOSIS_DATE DATE NOT NULL,
CONSTRAINT PK_TOPS-DIAGNOSIS PRIMARY KEY(ID)
);

CREATE TABLE TOPS.TOPS_DRUG
(
ID NUMBER(38),
NAME VARCHAR2(30) NOT NULL,
CODE VARCHAR2(10),
MIN_DOSE NUMBER(10,3),
MAX_DOSE NUMBER(10,3),
DOSE_UNITS VARCHAR2(10),
DOSE_FREQUENCY NUMBER(4) NOT NULL,
DOSE_FREQ_UNIT VARCHAR(9) NOT NULL,
CONSTRAINT PK_DRUG PRIMARY KEY(ID)
);

CREATE TABLE TOPS.TOPS_PRESCRIPTION
(
ID NUMBER(38),
PATIENT_ID NUMBER(38) NOT NULL,
DRUG_ID NUMBER(38) NOT NULL,
DATE_PRESCIRBED DATE NOT NULL,
CONSTRAINT PK_TOPS_PRESCRIPTION PRIMARY KEY(ID)
);

CREATE TABLE TOPS.TOPS_ADVICE
(
ID NUMBER(38) NOT NULL,
MSG VARCHAR2(300) NOT NULL,
MSG_CODE VARCHAR2(20) UNIQUE NOT NULL,
CONSTRAINT PK_TOPS_ADVICE PRIMARY KEY(ID)
);

CREATE TABLE TOPS.TOPS_PATIENT_ADVICE
(
ID NUMBER(38) NOT NULL,
PATIENT_ID NUMBER(38) NOT NULL,
ADVICE_ID NUMBER(38) NOT NULL,
RULE_ID NUMBER(38) NOT NULL,
DATE_GIVEN DATE NOT NULL,
CONSTRAINT PK_TOPS_PATIENT_ADVICE PRIMARY KEY (ID)
);

CREATE TABLE TOPS.TOPS_REFERRAL
(
ID NUMBER(38),
PATIENT_ID NUMBER(38) NOT NULL,
SPECIALIST VARCHAR2(100) NOT NULL,
MSG  VARCHAR2(300) NOT NULL,
DATE_REFERRED DATE NOT NULL,
RULE_NAME VARCHAR2(50) NOT NULL,
CONSTRAINT PK_TOPS_REFERRAL PRIMARY KEY(ID)
);

CREATE TABLE TOPS.TOPS_PATIENT_STATE
(
ID NUMBER(38) NOT NULL,
PATIENT_ID NUMBER(38) NOT NULL,
STATE_NAME VARCHAR2(100) NOT NULL,
CHANGE_DATE DATE NOT NULL,
RULE_NAME VARCHAR2(100) NOT NULL,
CONSTRAINT PK_TOPS_PATIENT_STATE PRIMARY KEY(ID)
);

CREATE TABLE TOPS.T_TEST
(
ID NUMBER(38)  NOT NULL,
CODE VARCHAR2(100)  UNIQUE NOT NULL,
MIN NUMBER(12,6) NOT NULL,
MAX NUMBER(12,6) NOT NULL,
UNITS VARCHAR2(10),
CONSTRAINT PK_T_TEST PRIMARY KEY (ID)
);

CREATE TABLE TOPS.T_RESULTS
(
ID NUMBER(38)  NOT NULL,
TEST_ID NUMBER(38)  NOT NULL,
ORDER_ID NUMBER(38)  NOT NULL,
RESULT_VALUE NUMBER(12,6)  NOT NULL,
RESULT_DATE DATE  NOT NULL,
CONSTRAINT PK_T_RESULTS PRIMARY KEY (ID)
);

CREATE TABLE TOPS.T_RESULT_STATS
(
ID NUMBER(38)  NOT NULL,
RESULT_ID NUMBER(38) NOT NULL,
PATIENT_ID NUMBER(38) NOT NULL,
CURR_RESULT NUMBER(12,6) NOT NULL,
CURR_RESULT_DATE DATE  NOT NULL,
PREV_RESULT NUMBER(12,6) ,
PREV_RESULT_DATE DATE,
RESULT_DELTA NUMBER(12,6),
RESULT_AVERAGE NUMBER(12,6),
CONSTRAINT PK_T_RESULT_STATS PRIMARY KEY (ID)
);

CREATE TABLE TOPS.T_ORDERED_TESTS
(
ID NUMBER(38)  NOT NULL,
PATIENT_ID NUMBER(38)  NOT NULL,
PROFILE_ID NUMBER(38)  NOT NULL,
ORDER_DATE DATE NOT  NULL,
CLIENT_ADDRESS VARCHAR2(20),
CONSTRAINT PK_T_ORDERED_TESTS PRIMARY KEY (ID)
);

CREATE TABLE TOPS.T_PROFILE
(
ID NUMBER(38)  NOT NULL,
NAME VARCHAR2(200) NOT NULL,
CODE VARCHAR2(100)  UNIQUE NOT NULL,
DESCRIPTION VARCHAR2(300) NULL,
CONSTRAINT PK_TEST_T_PROFILE PRIMARY KEY (ID)
);

CREATE TABLE TOPS.T_PROFILE_TEST
(
PROFILE_ID NUMBER(38)  NOT NULL,
TEST_ID NUMBER(38)  NOT NULL,
CONSTRAINT PK_PROFILE_TEST PRIMARY KEY (PROFILE_ID, TEST_ID)
);

CREATE TABLE TOPS.T_ACR2OF3_STATUS
(
ID NUMBER(38) NOT NULL,
PATIENT_ID NUMBER(38) NOT NULL,
RULE_ID NUMBER(38) NOT NULL,
STATUS VARCHAR2(100) NOT NULL,
DATE_CHECKED DATE NOT NULL,
CONSTRAINT PK_T_ACR2OF3CHECK_STATUS PRIMARY KEY(ID)
);

CREATE TABLE TOPS.T_ACR_RESULT
(
ID NUMBER(38) NOT NULL,
PATIENT_ID NUMBER(38) NOT NULL,
RESULT_ID NUMBER(38) NOT NULL,
RESULT_COUNT NUMBER(38) NOT NULL,
COUNT_DATE DATE NOT NULL,
CONSTRAINT PK_T_ACR_RESULT PRIMARY KEY (ID)
);

ALTER TABLE TOPS.TOPS_PATIENT_ADVICE ADD
(
CONSTRAINT FK_PATIENT_ADVICE_PATIENT
                FOREIGN KEY (PATIENT_ID)
                                REFERENCES TOPS.TOPS_PATIENTS(ID)
);

ALTER TABLE TOPS.TOPS_PATIENT_ADVICE ADD
(
CONSTRAINT FK_PATIENT_ADVICE_RULE
                FOREIGN KEY ( RULE_ID )
                                REFERENCES TOPS.PL_RULE (ID)
);

ALTER TABLE TOPS.TOPS_PATIENT_ADVICE ADD
(
CONSTRAINT FK_PATIENT_ADVICE_ADVICE
                FOREIGN KEY(ADVICE_ID)
                                REFERENCES TOPS.TOPS_ADVICE(ID)
);


ALTER TABLE TOPS.TOPS_DIAGNOSIS ADD(
CONSTRAINT FK_TOPS_DIAGNOSIS_PATIENT
                FOREIGN KEY (PATIENT_ID)
                                REFERENCE TOPS.TOPS_PATIENTS(ID)
);

ALTER TABLE TOPS.TOPS.TOPS_DISCHARGE ADD(
CONSTRAINT FK_TOPS_DISCHARGE_PATIENT
                FOREIGN KEY (PATIENT_ID)
                                REFERENCE TOPS.TOPS_PATIENTS(ID)
);

ALTER TABLE TOPS.TOPS_ADMISSION ADD(
CONSTRAINT FK_TOPS_ADMISSION_PATIENT
                FOREIGN KEY (PATIENT_ID)
                                REFERENCE TOPS.TOPS_PATIENTS(ID)
);

ALTER TABLE TOPS.TOPS_PATIENT_HISTORY ADD(
CONSTRAINT FK_PATIENT_HISTORY
                FOREIGN KEY (PATIENT_ID)
                                REFERENCE TOPS.TOPS_PATIENTS(ID)
);

ALTER TABLE TOPS.TOPS_REFERRAL ADD(
CONSTRAINT FK_TOPS_REFERRAL_PATIENT
                FOREIGN KEY (PATIENT_ID)
                                REFERENCES TOPS.TOPS_PATIENTS(ID)
);

ALTER TABLE TOPS.TOPS_PRESCRIPTION ADD(
CONSTRAINT FK_TOPS_PRESCRIPTION_PATIENT
                FOREIGN KEY (PATIENT_ID)
                                REFERENCES TOPS.TOPS_PATIENTS(ID)
);


ALTER TABLE TOPS.TOPS_PATIENTS ADD (
CONSTRAINT FK_PATIENTS_CATEGORIES
```

```
                FOREIGN KEY (CATEGORY_ID)
                        REFERENCES TOPS_CATEGORIES (ID)
);

ALTER TABLE TOPS.TOPS_PATIENT_STATE ADD(
CONSTRAINT FK_TOPS_PSTATE_PATIENT
                FOREIGN KEY (PATIENT_ID)
                        REFERENCE TOPS.TOPS_PATIENTS(ID)
);


ALTER TABLE TOPS.T_ACR_RESULT ADD (
   CONSTRAINT FK_T_ACR_RESULT_RESULT
      FOREIGN KEY (RESULT_ID)
         REFERENCES TOPS.T_RESULTS (ID)
            ON DELETE CASCADE);

ALTER TABLE TOPS.T_ACR_RESULT ADD (
   CONSTRAINT FK_T_ACR_RESULT_PATIENT
      FOREIGN KEY (PATIENT_ID)
         REFERENCES TOPS.TOPS_PATIENTS (ID)
            ON DELETE CASCADE);

ALTER TABLE TOPS.T_ACR2OF3_STATUS ADD (
   CONSTRAINT FK_ACR2OF3_STATUS_PATIENT
      FOREIGN KEY (PATIENT_ID)
         REFERENCES TOPS.TOPS_PATIENTS (ID)
            ON DELETE CASCADE);

ALTER TABLE T_ACR2OF3_STATUS ADD (
   CONSTRAINT FK_ACR2OF3_STATUS_RULE
      FOREIGN KEY (RULE_ID)
         REFERENCES PL_RULE (ID)
            ON DELETE CASCADE);

ALTER TABLE TOPS.T_PROFILE_TEST ADD (
   CONSTRAINT FK_PROF_CONTAINS_TEST_PRO
      FOREIGN KEY (PROFILE_ID)
         REFERENCES TOPS.T_PROFILE (ID)
            ON DELETE CASCADE);

ALTER TABLE TOPS.T_PROFILE_TEST ADD (
   CONSTRAINT FK_PROF_CONTAINS_TEST_TEST
      FOREIGN KEY (TEST_ID)
         REFERENCES TOPS.T_TEST (ID)
           ON DELETE CASCADE);

ALTER TABLE TOPS.T_ORDERED_TESTS ADD (
      CONSTRAINT FK_T_ORDERED_TESTS_PATIENT
         FOREIGN KEY (PATIENT_ID)
            REFERENCES TOPS.TOPS_PATIENTS(ID));

ALTER TABLE TOPS.T_ORDERED_TESTS ADD (
          CONSTRAINT FK_T_ORDERED_TESTS_PROFILE
             FOREIGN KEY (PROFILE_ID)
                REFERENCES T_PROFILE (ID));


ALTER TABLE TOPS.T_RESULTS ADD (
          CONSTRAINT FK_T_RESULTS_TESTS
             FOREIGN KEY (TEST_ID)
                REFERENCES TOPS.T_TEST(ID)
                   ON DELETE CASCADE);

ALTER TABLE TOPS.T_RESULTS ADD (
          CONSTRAINT FK_T_RESULTS_ORDER
             FOREIGN KEY (ORDER_ID)
                REFERENCES TOPS.T_ORDERED_TESTS (ID)
                   ON DELETE CASCADE);

ALTER TABLE TOPS.T_RESULT_STATS ADD (
   CONSTRAINT FK_RESULT_STATS_RESULT
      FOREIGN KEY (RESULT_ID)
         REFERENCES TOPS.T_RESULTS (ID)
            ON DELETE CASCADE);

ALTER TABLE TOPS.T_RESULT_STATS ADD (
   CONSTRAINT FK_RESULT_STATS_PATIENT
      FOREIGN KEY (PATIENT_ID)
         REFERENCES TOPS.TOPS_PATIENTS (ID)
            ON DELETE CASCADE);
```

B.4. The TOPS database views

```
CREATE OR REPLACE VIEW PR_DYNAMIC_RULE_VW
(
ID, NAME, EVENT_ID,
DESCRIPTION,DATE_CREATED,
RULE_TYPE, ECA_RULE_TYPE
)
AS
SELECT PR_RULE.ID, PR_RULE.NAME, PR_DYNAMIC_RULE.EVENT_ID, PR_RULE.DESCRIPTION,
PR_RULE.DATE_CREATED, PR_RULE.RULE_TYPE, PR_DYNAMIC_RULE.RULE_TYPE
FROM PR_RULE, PR_DYNAMIC_RULE
WHERE PR_RULE.ID = PR_DYNAMIC_RULE.ID ;

CREATE OR REPLACE VIEW PR_PROTOCOL_RULE_VW
(
ID, EVENT_ID, RULE_TYPE
)
AS
SELECT PR_DYNAMIC_RULE.ID, PR_DYNAMIC_RULE.EVENT_ID, PR_DYNAMIC_RULE.RULE_TYPE
FROM PR_DYNAMIC_RULE, PR_PROTOCOL_RULE
WHERE PR_DYNAMIC_RULE.ID = PR_PROTOCOL_RULE.ID;

CREATE OR REPLACE VIEW PRSCHEDULE_RULE_VW
(
ID, EVENT_ID, RULE_TYPE
)
AS
SELECT PR_DYNAMIC_RULE.ID, PR_DYNAMIC_RULE.EVENT_ID, PR_DYNAMIC_RULE.RULE_TYPE
FROM PR_DYNAMIC_RULE, PR_SCHEDULE_RULE
WHERE PR_DYNAMIC_RULE.ID = PR_SCHEDULE_RULE.ID ;

CREATE OR REPLACE VIEW PR_STATIC_RULE_VW
(
ID, NAME, DESCRIPTION,
RULE_TYPE, DATE_CREATED, ZERO_TIME_REF_TERM,
START_TIME, EXPIRY_TIME, INTERVAL
)
AS
SELECT PR_RULE.ID, PR_RULE.NAME, PR_RULE.DESCRIPTION,
PR_RULE.RULE_TYPE,PR_RULE.DATE_CREATED, PR_STATIC_RULE.ZERO_TIME_REF_TERM,
PR_STATIC_RULE.START_TIME, PR_STATIC_RULE.END_TIME, PR_STATIC_RULE.INTERVAL
FROM PR_RULE, PR_STATIC_RULE
WHERE PR_RULE.ID = PR_STATIC_RULE.ID ;

CREATE OR REPLACE VIEW TOPS.PR_ECA_VW
(
ID, EVENT_ID,
CONDITION_ID, ACTION_ID
)
AS
SELECT PR_DYNAMIC_RULE.ID, PR_DYNAMIC_RULE.EVENT_ID,
PR_RULE_CONDITION.CONDITION_ID, PR_RULE_ACTION.ACTION_ID
FROM TOPS.PR_DYNAMIC_RULE, TOPS.PR_RULE_ACTION, TOPS.PR_RULE_CONDITION
WHERE TOPS.PR_DYNAMIC_RULE.ID = TOPS.PR_RULE_CONDITION.RULE_ID
AND TOPS.PR_DYNAMIC_RULE.ID = TOPS.PR_RULE_ACTION.RULE_ID;

CREATE OR REPLACE VIEW TOPS.PL_PATIENT_DRULES_VW
(
PATIENT_ID,
DRULE_ID
)
AS
SELECT PLAN.PATIENT_ID PATIENT_ID, RULE.ID DRULE_ID
FROM TOPS.PL_DYNAMIC_RULE RULE, TOPS.PL_PLAN PLAN
WHERE PLAN.ID = RULE.PLAN_ID;

CREATE OR REPLACE VIEW TOPS.PL_PATIENT_SRULES_VW
(
PATIENT_ID,
SRULE_ID,
SCHEDULE_ID
)
AS
SELECT PLAN.PATIENT_ID PATIENT_ID, RULE.ID SRULE_ID, SCHEDULE.ID SCHEDULE_ID
FROM TOPS.PL_STATIC_RULE RULE, TOPS.PL_PLAN PLAN, TOPS.PL_SCHEDULE SCHEDULE
WHERE (PLAN.ID = SCHEDULE.PLAN_ID) AND (SCHEDULE.ID = RULE.SCHEDULE_ID);

CREATE OR REPLACE VIEW TOPS.PL_PLAN_SRULE_VW
(
PLAN_ID,
SRULE_ID
)
AS
SELECT SCHEDULE.PLAN_ID PLAN_ID, SRULE.ID SRULE_ID
FROM TOPS.PL_STATIC_RULE SRULE, TOPS.PL_SCHEDULE SCHEDULE
WHERE SCHEDULE.ID = SRULE.SCHEDULE_ID;

CREATE OR REPLACE VIEW TOPS.PL_PLAN_RULES_VW
(
PLAN_ID,
RULE_ID,
RULE_NAME,
RULE_TYPE
)
AS
SELECT PLAN.ID PLAN_ID, RULE.ID RULE_ID, RULE.NAME RULE_NAME, RULE.RULE_TYPE
FROM TOPS.PL_RULE RULE, TOPS.PL_PLAN PLAN, PL_PLAN_SRULE_VW
WHERE (PLAN.ID = PL_PLAN_SRULE_VW.PLAN_ID AND PL_PLAN_SRULE_VW.SRULE_ID = RULE.ID)
UNION
SELECT PLAN.ID PLAN_ID, RULE.ID RULE_ID, RULE.NAME RULE_NAME, RULE.RULE_TYPE
FROM TOPS.PL_RULE RULE, TOPS.PL_PLAN PLAN, PL_DYNAMIC_RULE
WHERE (PLAN.ID = PL_DYNAMIC_RULE.PLAN_ID  AND PL_DYNAMIC_RULE.ID = RULE.ID);



CREATE OR REPLACE VIEW PL_TABLE_RULE_VW
(
TABLE_NAME,
RULE, RULE_ID
)
AS
SELECT TABLE_NAME, TRIGGER_NAME RULE, PL_RULE.ID RULE_ID
FROM ALL_TRIGGERS, PL_RULE
WHERE (ALL_TRIGGERS.OWNER = 'TOPS' ) AND (SUBSTR(ALL_TRIGGERS.TRIGGER_NAME,1,2) = 'P$' )
AND
(SUBSTR(ALL_TRIGGERS.TRIGGER_NAME,-2,2) != 'ID' ) AND
(UPPER(ALL_TRIGGERS.TRIGGER_NAME) = UPPER(PL_RULE.NAME));

CREATE OR REPLACE VIEW TOPS.PL_CATEGORY_PLAN_VW
(
CATEGORY_ID,
PLAN_ID
)
AS
SELECT CATEGORY_ID, PL_PLAN.ID PLAN_ID
FROM TOPS.PR_PROTOCOL, TOPS.PL_PLAN
WHERE TOPS.PR_PROTOCOL.ID = TOPS.PL_PLAN.PROTOCOL_ID;
CREATE OR REPLACE VIEW TOPS.PL_ECA_TRIGGER_VW
(
PLAN_ID,
RULE_ID,
EVENT,
CONDITION,
ACTION
)
AS
SELECT PLAN_ID, RULE_ID, ALL_TRIGGERS.TRIGGERING_EVENT EVENT,
ALL_TRIGGERS.WHEN_CLAUSE CONDITION,
ALL_TRIGGERS.TRIGGER_BODY ACTION
FROM TOPS.PL_PLAN_RULES_VW, ALL_TRIGGERS
WHERE (UPPER(TOPS.PL_PLAN_RULES_VW.RULE_NAME) =
ALL_TRIGGERS.TRIGGER_NAME) AND (ALL_TRIGGERS.OWNER = 'TOPS');

CREATE OR REPLACE VIEW PR_DYNAMIC_RULE_VW
(
ID,
NAME,
EVENT_ID,
SPECIFICATION,
DESCRIPTION,
VERSION,
DATE_CREATED,
RULE_TYPE,
ECA_RULE_TYPE
)
AS
 SELECT PR_RULE.ID, PR_RULE.NAME, PR_DYNAMIC_RULE.EVENT_ID,
PR_RULE.SPECIFICATION, PR_RULE.DESCRIPTION,
PR_RULE.DATE_CREATED, PR_RULE.RULE_TYPE,
PR_DYNAMIC_RULE.RULE_TYPE
FROM PR_RULE, PR_DYNAMIC_RULE
WHERE PR_RULE.ID = PR_DYNAMIC_RULE.ID ;

CREATE VIEW PR_PROTOCOL_RULE_VW
(
ID,
EVENT_ID,
RULE_TYPE
)
AS
SELECT PR_DYNAMIC_RULE.ID, PR_DYNAMIC_RULE.EVENT_ID,
PR_DYNAMIC_RULE.RULE_TYPE
FROM PR_DYNAMIC_RULE, PR_PROTOCOL_RULE
WHERE  PR_DYNAMIC_RULE.ID = PR_PROTOCOL_RULE.ID;

CREATE OR REPLACE VIEW PRSCHEDULE_RULE_VW
(
ID,
EVENT_ID,
RULE_TYPE
)
AS
SELECT PR_DYNAMIC_RULE.ID, PR_DYNAMIC_RULE.EVENT_ID,
PR_DYNAMIC_RULE.RULE_TYPE
FROM PR_DYNAMIC_RULE, PR_SCHEDULE_RULE
WHERE PR_DYNAMIC_RULE.ID = PR_SCHEDULE_RULE.ID ;

CREATE OR REPLACE VIEW PR_STATIC_RULE_VW
(
ID, NAME, SPECIFICATION,
DESCRIPTION, RULE_TYPE, VERSION,
DATE_CREATED, ZERO_TIME_REF_TERM, START_TIME,
EXPIRY_TIME, INTERVAL
)
AS
SELECT PR_RULE.ID, PR_RULE.NAME, PR_RULE.SPECIFICATION,
PR_RULE.DESCRIPTION, PR_RULE.RULE_TYPE,
PR_RULE.DATE_CREATED, PR_STATIC_RULE.ZERO_TIME_REF_TERM,
PR_STATIC_RULE.START_TIME, PR_STATIC_RULE.EXPIRY_TIME,
PR_STATIC_RULE.INTERVAL
FROM PR_RULE, PR_STATIC_RULE
WHERE PR_RULE.ID = PR_STATIC_RULE.ID ;

CREATE OR REPLACE VIEW TOPS.PR_ECA_VW
(
ID, EVENT_ID,
CONDITION_ID, ACTION_ID
)
AS
SELECT PR_DYNAMIC_RULE.ID, PR_DYNAMIC_RULE.EVENT_ID,
PR_RULE_CONDITION.CONDITION_ID, PR_RULE_ACTION.ACTION_ID
FROM TOPS.PR_DYNAMIC_RULE, TOPS.PR_RULE_ACTION,
TOPS.PR_RULE_CONDITION
WHERE (TOPS.PR_DYNAMIC_RULE.ID =
TOPS.PR_RULE_CONDITION.RULE_ID) AND (TOPS.PR_DYNAMIC_RULE.ID =
TOPS.PR_RULE_ACTION.RULE_ID;)

CREATE OR REPLACE VIEW TOPS.PATIENT_ORDER_TEST_RESULT_VW
(
PATIENT_ID, ORDER_ID,
PROFILE_ID, TEST_ID,
RESULT_ID, RESULT,
RESULT_DATE
)
AS
SELECT PATIENT_ID, ORDER_ID, TOPS.T_ORDERED_TESTS.PROFILE_ID,
TOPS.T_PROFILE_TEST.TEST_ID,
TOPS.T_RESULTS.ID RESULT_ID,RESULT_VALUE RESULT, RESULT_DATE
FROM TOPS.T_ORDERED_TESTS, TOPS.T_RESULTS, TOPS.T_PROFILE_TEST
WHERE (TOPS.T_ORDERED_TESTS.ID = TOPS.T_RESULTS.ORDER_ID) AND
```

```
(TOPS.T_ORDERED_TESTS.PROFILE_ID = TOPS.T_PROFILE_TEST.PROFILE_ID) AND
(TOPS.T_RESULTS.TEST_ID = TOPS.T_PROFILE_TEST.TEST_ID);

CREATE OR REPLACE VIEW PL_STATIC_RULE_NAMES
(
STATIC_RULE
)
AS
SELECT NAME STATIC_RULE FROM PL_RULE, PL_STATIC_RULE
WHERE PL_RULE.ID=PL_STATIC_RULE.ID;

CREATE OR REPLACE VIEW PL_RULE_ACTION_VW
(
PL_RULE_ID, PR_ACTION_ID,
PR_ACTION_NAME, PARAMETERS
)
AS
SELECT DISTINCT PL_RULE.ID PL_RULE_ID, PR_RULE_ACTION.ACTION_ID PR_ACTION_ID,
PR_ACTION.NAME PR_ACTION_NAME,
PR_RULE_ACTION.ACTION_PARAMETERS PARAMETERS
FROM PR_RULE_ACTION,PR_ACTION, PL_PLAN_PROTOCOL_RULE, PL_RULE
WHERE (PL_PLAN_PROTOCOL_RULE.PL_RULE_ID = PL_RULE.ID ) AND
(PR_ACTION.ID=PR_RULE_ACTION.ACTION_ID ) AND
(PL_PLAN_PROTOCOL_RULE.PR_RULE_ID=PR_RULE_ACTION.RULE_ID);

CREATE OR REPLACE VIEW PL_DR_EVENT_VW
(
PL_RULE_ID, PR_EVENT_ID,
PR_EVENT_NAME
)
AS
SELECT DISTINCT PL_DYNAMIC_RULE.ID PL_RULE_ID, PR_DYNAMIC_RULE.EVENT_ID
PR_EVENT_ID, PR_EVENT.NAME PR_EVENT_NAME
FROM PR_DYNAMIC_RULE,PR_EVENT, PL_PLAN_PROTOCOL_RULE, PL_DYNAMIC_RULE
WHERE (PL_PLAN_PROTOCOL_RULE.PL_RULE_ID = PL_DYNAMIC_RULE.ID) AND
(PR_EVENT.ID=PR_DYNAMIC_RULE.EVENT_ID);

CREATE OR REPLACE VIEW PL_ECA_VW
(
PLAN_ID, RULE_ID,
EVENT, CONDITION,
ACTION, ACTION_PARAMS
)
AS
SELECT DISTINCT PLAN_ID, PL_DR_EVENT_VW.PL_RULE_ID RULE_ID, PR_EVENT_NAME EVENT,
CONDITION, PR_ACTION_NAME ACTION,
PARAMETERS ACTION_PARAMS
FROM PL_DR_EVENT_VW, PL_ECA_TRIGGER_VW, PL_RULE_ACTION_VW
WHERE (PL_DR_EVENT_VW.PL_RULE_ID=PL_RULE_ACTION_VW.PL_RULE_ID ) AND
(PL_DR_EVENT_VW.PL_RULE_ID=PL_ECA_TRIGGER_VW.RULE_ID);

CREATE OR REPLACE VIEW PL_HISTORY_DR_VW
(
LOG_NO, PLAN_ID,
RULE_ID, RULE_NAME,
EVENT, ACTION,
EXEC_DATE
)
AS
SELECT TOPS.PL_ACTIVITY_LOG.ID LOG_NO, TOPS.PL_ACTIVITY_LOG.PLAN_ID,
TOPS.PL_ACTIVITY_LOG.RULE_ID_EXECUTED RULE_ID,
TOPS.PL_RULE.NAME RULE_NAME, EVENT, ACTION, TIME_EXECUTED
FROM TOPS.PL_ACTIVITY_LOG, TOPS.PL_ECA_VW, TOPS.PL_RULE
WHERE (RULE_ID_EXECUTED=TOPS.PL_ECA_VW.RULE_ID) AND
(RULE_ID_EXECUTED=TOPS.PL_RULE.ID);

CREATE OR REPLACE VIEW PL_HISTORY_SR_VW
(
LOG_NO, PLAN_ID,
RULE_ID, RULE_NAME,
ACTION, EXEC_DATE
)
AS
SELECT LOG_NO, PLAN_ID, RULE_ID, RULE_NAME, PR_ACTION_NAME ACTION, TIME_EXECUTED
EXEC_DATE
FROM
(
SELECT TOPS.PL_ACTIVITY_LOG.ID LOG_NO, TOPS.PL_ACTIVITY_LOG.PLAN_ID,
TOPS.PL_ACTIVITY_LOG.RULE_ID_EXECUTED RULE_ID,
TOPS.PL_RULE.NAME RULE_NAME, TOPS.PL_ACTIVITY_LOG.TIME_EXECUTED
FROM TOPS.PL_ACTIVITY_LOG, TOPS.PL_PLAN_SRULE_VW, TOPS.PL_RULE
WHERE (RULE_ID_EXECUTED=TOPS.PL_PLAN_SRULE_VW.SRULE_ID) AND
(TOPS.PL_RULE.ID=RULE_ID_EXECUTED
),
TOPS.PL_RULE_ACTION_VW
WHERE PL_RULE_ACTION_VW.PL_RULE_ID = RULE_ID;

CREATE OR REPLACE VIEW PL_HISTORY_VW
(
LOG_NO, PLAN_ID,
RULE_ID, RULE_NAME,
ACTION, EXEC_DATE
)
AS
SELECT *
FROM PL_HISTORY_SR_VW
UNION
(
SELECT LOG_NO, PLAN_ID, RULE_ID, RULE_NAME, ACTION, EXEC_DATE
FROM PL_HISTORY_DR_VW
);

CREATE OR REPLACE VIEW PL_HISTORY_PATIENT_VW
(
LOG_NO, PATIENT_ID,
PLAN_ID, RULE_NAME,
ACTION, EXEC_DATE
)
AS
SELECT TOPS.PL_HISTORY_VW.LOG_NO, TOPS.PL_PLAN.PATIENT_ID,
PLAN_ID, RULE_NAME, ACTION, EXEC_DATE
FROM TOPS.PL_HISTORY_VW, TOPS.PL_PLAN
WHERE TOPS.PL_PLAN.ID=TOPS.PL_HISTORY_VW.PLAN_ID;

CREATE OR REPLACE VIEW PL_HISTORY_MRN_VW
(
LOG_NO, PATIENT_ID,
MRN,PLAN_ID,RULE_NAME,
ACTION, EXEC_DATE
)
AS
SELECT LOG_NO, PATIENT_ID, MRN, PLAN_ID, RULE_NAME, ACTION,
EXEC_DATE
FROM TOPS.PL_HISTORY_PATIENT_VW, TOPS.TOPS_PATIENTS
WHERE TOPS.PL_HISTORY_PATIENT_VW.PATIENT_ID =
TOPS.TOPS_PATIENTS.ID;

CREATE OR REPLACE VIEW PL_HISTORY_PLAN_VW
(
LOG_NO, A_DATE,
PATIENT_ID, MRN,
PLAN_ID, DR,
SR, EXPLANATION
)
AS
SELECT DISTINCT TOPS.PL_ACTIVITY_LOG.ID LOG_NO, TIME_EXECUTED
A_DATE, TOPS.PL_HISTORY_MRN_VW.PATIENT_ID,
TOPS.PL_HISTORY_MRN_VW.MRN,
TOPS.PL_ACTIVITY_LOG.PLAN_ID, DYN_RULES DR, STC_RULES SR,
TOPS.PL_PLAN_RULES_VW.RULE_TYPE || ' RULE ' || RULE_ID_EXECUTED || '
EXECUTED.' EXPLANATION
FROM TOPS.PL_ACTIVITY_LOG, TOPS.PL_PLAN_RULES_VW,
TOPS.PL_HISTORY_MRN_VW
WHERE (RULE_ID_EXECUTED=TOPS.PL_PLAN_RULES_VW.RULE_ID) AND
(TOPS.PL_HISTORY_MRN_VW.PLAN_ID=TOPS.PL_ACTIVITY_LOG.PLAN_ID);

CREATE OR REPLACE VIEW PL_PLAN_RULE_ORDER_VW AS
SELECT PLAN_ID, PL_RULE_ORDER_LOG.RULE_ID, ORDER_ID, EXEC_DATE
FROM PL_PLAN_RULES_VW, PL_RULE_ORDER_LOG
WHERE PL_PLAN_RULES_VW.RULE_ID = PL_RULE_ORDER_LOG.RULE_ID;

CREATE OR REPLACE VIEW PL_PLAN_SNAPSHOT_VW
(
PLAN_ID, PLAN_NAME, RULE_ID,
RULE_NAME, RULE_TYPE, RULE_STATE,
SNAP_TIME
)
AS
SELECT PLAN_ID, PL_PLAN.NAME PLAN_NAME, PL_RULE.ID RULE_ID,
PL_PLAN_RULES_VW.RULE_NAME, PL_PLAN_RULES_VW.RULE_TYPE,
PL_RULE.CURRENT_STATE RULE_STATE, SYSDATE SNAP_TIME
FROM PL_PLAN_RULES_VW, PL_RULE, PL_PLAN
WHERE (PL_PLAN_RULES_VW.RULE_ID=PL_RULE.ID AND )
(PL_PLAN_RULES_VW.PLAN_ID=PL_PLAN.ID);

CREATE OR REPLACE VIEW PL_PATIENT_PLAN_VW
(
PATIENT_ID, PLAN_ID, PLAN_NAME,
RULE_ID, RULE_NAME, RULE_TYPE
)
AS
SELECT UNIQUE PL_PLAN.PATIENT_ID, PL_PLAN.ID
PLAN_ID,PL_PLAN.NAME PLAN_NAME, RULE_ID,RULE_NAME,RULE_TYPE
FROM PL_PLAN, PL_PLAN_RULES_VW, PL_PATIENT_SRULES_VW,
PL_PATIENT_DRULES_VW
WHERE
(PL_PLAN_RULES_VW.RULE_ID=PL_PATIENT_SRULES_VW.SRULE_ID AND
PL_PATIENT_SRULES_VW.PATIENT_ID=PL_PLAN.PATIENT_ID
AND PL_PLAN.ID=PL_PLAN_RULES_VW.PLAN_ID) OR
(PL_PLAN_RULES_VW.RULE_ID=PL_PATIENT_DRULES_VW.DRULE_ID
AND PL_PATIENT_DRULES_VW.PATIENT_ID=PL_PLAN.PATIENT_ID AND
PL_PLAN.ID=PL_PLAN_RULES_VW.PLAN_ID) ;

CREATE OR REPLACE VIEW T_ACR_RESULT_VW
(
PATIENT_ID, RESULT_ID, ORDER_ID,
RESULT_VALUE, RESULT_DATE
)
AS
SELECT PATIENT_ID, RESULT_ID, ORDER_ID, RESULT RESULT_VALUE,
RESULT_DATE
FROM PATIENT_ORDER_TEST_RESULT_VW, T_TEST
WHERE (T_TEST.CODE='ACR') AND (TEST_ID=ID);
```

C. The MAP Specification in PLAN

```
@PROTOCOL@ MAP2;
DESCRIPTION: This is a protocol for the diagnosis and management of microalbuminuria in diabetes patients;
CREATOR: DR JOHN NOLAN;
CATEGORY: DIABETIC_NEPHROPATHY;
#SCHEDULE_SET#
^SCHEDULE^ AUS,
DESCRIPTION: This is a microalbuminuria protocol schedule called AUS for Annual dipstick Urine Screening;
RULE AUS2,
DESCRIPTION: if dipstick urine test shows presence of blood and leucocytes check presence or absence of other
infections e.g. urinary tract infections,
ON: result_arrival('DSU'),
IF: DSU%result%database%t_results =  positive%string,
DO: patient_state ( 'other_infections_screening' );
RULE AUS3,
DESCRIPTION: if dipstick urine test is negative then screen for microalbuminuria,
ON: result_arrival('DSU'),
IF: DSU%RESULT%DATABASE%T_RESULT = NEGATIVE%STRING,
DO: PATIENT_STATE('microalbuminuria_screening' );
^END SCHEDULE^
^SCHEDULE^ OIS,
DESCRIPTION: This is a microalbuminuria protocol schedule called OIS for SCREENING OTHER INFECTIONS
in the diagnosis of microalbuminuria and proteinuria;
RULE OIS2,
DESCRIPTION: if UTI is not present then perform  24 hour creatinine and 24 hour protein loss tests,
ON: result_arrival('UTI'),
IF: UTI%result%database%t_result  =  negative%string,
DO: order_test('24CRCL_PL' );
RULE  OIS3,
DESCRIPTION: if UTI is present then place back on annual screening,
ON: result_arrival('UTI'),
IF: UTI%result%database%t_result  =  positive%string,
DO: patient_state( 'annual_urine_screening');
RULE OIS4,
DESCRIPTION: if 24 hour creatine clearance and 24 hour protein loss tests are positive then proteinuria is confirmed
and refer patient to nephrologist,
ON: result_arrival('24CRCL_PL'),
IF: 24CRCL_PL%RESULT%DATABASE%T_TEST = POSITIVE%STRING,
DO: patient_state ('nephrology_referral');
RULE 0IS5,
DESCRIPTION: if 24 hour creatine clearance and 24 hour protein loss is negative then return patient to annual
screening,
ON: result_arrival('24CRCL_PL'),
IF: 24CRCL_PL%RESULT%DATABASE%T_TEST = NEGATIVE%STRING,
DO: patient_state ('annual_urine_screening');
^END SCHEDULE^
^SCHEDULE^ MAS,
DESCRIPTION: This is a microalbuminuria protocol schedule called MAS for the screening of microalbuminuria;
RULE MAS2,
DESCRIPTION: if the first ACR result is > 20 mg/l order two more tests within the next six months,
ON: result_arrival('ACR'),
IF: ACR%RESULT%DATABASE%T_RESULTS > 20%DOUBLE,
DO: ADD_RULE
{
STATIC_RULE  MAS2a
*DESCRIPTION* rule orders ACR test during the next 6 month period
*FROM time_rule_added
*STARTING now
*ENDING 6 months
*ON_EVERY 3 months
*DO order_test ('ACR')
};
RULE MAS3,
DESCRIPTION: if ACR < 20 mg/l then place patient on annual screening,
ON: result_arrival('ACR'),
IF: ACR%RESULT%DATABASE%T_RESULTS > 20%DOUBLE,
DO: PATIENT_STATE('annual_urine_screening');
RULE MAS4,
DESCRIPTION: if 2 of 3 ACR in 20-200 mg/l within 6 months then microalbuminuria is confirmed,
ON: result_arrival('ACR'),
DO: CHECK_2OF3_ACR ();
RULE MAS5,
DESCRIPTION: if ACR > 200 mg/l then refer patient to nephrologist for possible proteinuria,
ON: RESULT_ARRIVAL('ACR'),
IF: ACR%RESULT%DATABASE%T_TEST  > 200%DOUBLE,
DO: PATIENT_STATE('nephrology_referral' );
^END SCHEDULE^
^SCHEDULE^ CMA,
DESCRIPTION: This is a microalbuminuria protocol schedule named CMA for confirmed microalbuminuria –
handles treatment and control of microalbuminuria;
RULE CMA5,
DESCRIPTION: if becomes normal (ACR < 20 mg/l) at any time  then the patient  is placed on annual screening,
ON: result_arrival('ACR'),
IF: ACR%RESULT%DATABASE%T_RESULT  <  20%DOUBLE,
DO: PATIENT_STATE('annual_urine_screening' );
RULE CMA6,
DESCRIPTION: if becomes abnormal (ACR > 200 mg/l) at any time  then the patient  is placed on nephrology
referral,
ON: result_arrival('ACR'),
IF: ACR%RESULT%DATABASE%T_RESULT >  200%DOUBLE,
DO: PATIENT_STATE('nephrology_referral' );
^END SCHEDULE^
^SCHEDULE^ NPH,
DESCRIPTION: This is a microalbuminuria protocol schedule named NPH for
nephrology referral – handles preparation and transmission of the necessary
documentation for the referral;
RULE NPH2,
DESCRIPTION: when a referral note is created it must immediately be sent to the
specialist either by post or e-mail,
ON: new_referral_note(),
DO: send_referral_note();
^END SCHEDULE^
#END SCHEDULE_SET#
~RULE_SET~
STATIC_RULE AUS1,
DESCRIPTION: dip-stick urine test at the end of every year for screening renal
complications in diabetes patients,
FROM: annual_screening_start_date,
STARTING: 0 minutes,
ENDING: 30 minutes,
ON_EVERY: 2 minutes,
DO: order_test('DSU');
RULE OIS1,
DESCRIPTION: on entry to the OIS schedule the patient is tested for other urinary tract
inections (UTI),
ON: state_change(),
IF: state_name%patient_state%database%tops_patient_state =
other_infections_screening%string,
DO: order_test('UTI');
RULE MAS1a,
DESCRIPTION: at the start of this schedule MAS order the two ACR and SCR tests,
ON: state_change(),
 IF: state_name%patient_state%database%tops_patient_state =
microalbuminuria_screening%string,
DO: order_test('ACR');
RULE MAS1b,
DESCRIPTION: at the start of this schedule MAS order the two ACR and SCR tests,
ON: state_change(),
IF: state_name%patient_state%database%tops_patient_state =
microalbuminuria_screening%string,
DO: order_test('SCR');
RULE CMA1,
DESCRIPTION: at the start of this schedule suggest optimisation of glycaemic control,
ON: state_change(),
IF: state_name%patient_state%database%tops_patient_state =
confirmed_microalbuminuria%string,
DO: suggest ('optimisation_of_glycaemic_control');
RULE CMA2,
DESCRIPTION: at the start of this schedule suggest BP measurement,
ON: state_change(),
IF: state_name%patient_state%database%tops_patient_state =
confirmed_microalbuminuria%string,
DO: ORDER_TEST ( 'BP');
RULE CMA3,
DESCRIPTION: If patient suffers from diabetes type 1 then prescribe ACE inhibitor,
ON: state_change(),
IF: state_name%patient_state%database%tops_patient_state =
confirmed_microalbuminuria%string,
DO: prescribe_medication('ACE_inhibitor');
RULE CMA4a,
DESCRIPTION: ACR and SCR tests are performed every month for all
microalbuminuria patients,
ON: state_change(),
IF: state_name%patient_state%database%tops_patient_state =
confirmed_microalbuminuria%string,
DO: order_test ('ACR');
RULE CMA4b,
DESCRIPTION: ACR and SCR tests are performed every month for all
microalbuminuria patients,
ON: state_change(),
IF: state_name%patient_state%database%tops_patient_state =
confirmed_microalbuminuria%string,
DO: order_test ('SCR');
RULE NPH1,
DESCRIPTION: when a patient is referred to a specialist a patient referral note is
created,
ON: state_change(),
IF: state_name%patient_state%database%tops_patient_state =
confirmed_microalbuminuria%string,
DO: create_referral_note ('nephrologist' );
~END RULE_SET~
@END PROTOCOL@
```

D. TOPS Session for Parsing the MAP

```
[2004-02-28 02:08:38.902] : Session Starting at 2004-02-28 2:08:38.552
[2004-02-28 02:08:40.004] : Getting confirmation to create the TOPS database objects.
[2004-02-28 02:11:13.725] : TOPS rule execution listener activated ...
[2004-02-28 02:11:13.885] : Rule listener waiting ...
[2004-02-28 02:12:22.584] : Analysing command: PARSE ...
[2004-02-28 02:12:22.604] : Executing command: PARSE(MAP2.TXT)
[2004-02-28 02:12:22.985] : Parsing protocol specification: D:\TOPS\\specs\\MAP2.TXT
[2004-02-28 02:12:23.636] : PROTOCOL SPECIFICATION
[2004-02-28 02:12:23.906] : Parsing: DESCRIPTION
[2004-02-28 02:12:23.976] : Parsing:  This is a protocol for the diagnosis and management of microalbuminuria in diabetes patients
[2004-02-28 02:12:24.046] : Parsing: CREATOR
[2004-02-28 02:12:24.116] : Parsing:  DR JOHN NOLAN
[2004-02-28 02:12:24.176] : Parsing: DR
[2004-02-28 02:12:24.256] : Parsing: JOHN
[2004-02-28 02:12:24.527] : Parsing: CATEGORY
[2004-02-28 02:12:24.597] : Parsing:  DIABETIC_NEPHROPATHY
[2004-02-28 02:12:24.967] : Category DIABETIC_NEPHROPATHY does not exists.
[2004-02-28 02:12:25.048] : Creating category DIABETIC_NEPHROPATHY.
[2004-02-28 02:12:25.298] : <add new category>
[2004-02-28 02:14:21.185] : Parsing: SCHEDULE_SET
[2004-02-28 02:14:21.365] : Parsing: ^SCHEDULE^
[2004-02-28 02:14:21.465] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:21.685] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:21.896] : Parsing: AUS
[2004-02-28 02:14:21.976] : Parsing: DESCRIPTION: This is a microalbuminuria protocol schedule called AUS for Annual dipstick Urine Screening
[2004-02-28 02:14:22.056] : Parsing: DESCRIPTION
[2004-02-28 02:14:22.126] : Parsing:  This is a microalbuminuria protocol schedule called AUS for Annual dipstick Urine Screening
[2004-02-28 02:14:22.517] : Parsing: RULE AUS2
[2004-02-28 02:14:22.747] : Parsing: RULE
[2004-02-28 02:14:22.827] : Parsing: AUS2
[2004-02-28 02:14:23.157] : Parsing: DESCRIPTION
[2004-02-28 02:14:23.238] : Parsing:  if dipstick urine test shows presence of blood and leucocytes check presence or absence of other infections e.g. urinary tract infections
[2004-02-28 02:14:23.518] : Parsing: ON
[2004-02-28 02:14:23.598] : Parsing:  result_arrival('DSU')
[2004-02-28 02:14:23.688] : Parsing: result_arrival
[2004-02-28 02:14:23.768] : Parsing: 'DSU')
[2004-02-28 02:14:23.848] : Parsing: 'DSU'
[2004-02-28 02:14:23.929] : Parsing: IF
[2004-02-28 02:14:24.019] : Parsing: IF
[2004-02-28 02:14:24.099] : Parsing:  DSU%result%database%t_results =  positive%string
[2004-02-28 02:14:24.179] : Parsing: DSU%result%database%t_results
[2004-02-28 02:14:24.259] : Parsing: =
[2004-02-28 02:14:24.339] : Parsing: positive%string
[2004-02-28 02:14:24.419] : Parsing: DSU
[2004-02-28 02:14:24.499] : Parsing: result
[2004-02-28 02:14:24.579] : Parsing: database
[2004-02-28 02:14:24.66] : Parsing: t_results
[2004-02-28 02:14:24.74] : Parsing: positive
[2004-02-28 02:14:24.82] : Parsing: string
[2004-02-28 02:14:24.95] : Parsing: DO
[2004-02-28 02:14:25.1] : Parsing:  patient_state ( 'other_infections_screening' )
[2004-02-28 02:14:25.19] : Parsing: patient_state
[2004-02-28 02:14:25.26] : Parsing:  'other_infections_screening' )
[2004-02-28 02:14:25.341] : Parsing: 'other_infections_screening'
[2004-02-28 02:14:25.431] : Parsing: RULE
[2004-02-28 02:14:25.591] : Parsing: AUS3
[2004-02-28 02:14:25.681] : Parsing: DESCRIPTION
[2004-02-28 02:14:25.771] : Parsing:  if dipstick urine test is negative then screen for microalbuminuria
[2004-02-28 02:14:25.851] : Parsing: ON
[2004-02-28 02:14:25.941] : Parsing:  result_arrival('DSU')
[2004-02-28 02:14:26.012] : Parsing: result_arrival
[2004-02-28 02:14:26.092] : Parsing: 'DSU')
[2004-02-28 02:14:26.172] : Parsing: 'DSU'
[2004-02-28 02:14:26.252] : Parsing: IF
[2004-02-28 02:14:26.332] : Parsing: IF
[2004-02-28 02:14:26.412] : Parsing:  DSU%RESULT%DATABASE%T_RESULT = NEGATIVE%STRING
[2004-02-28 02:14:26.492] : Parsing: DSU%RESULT%DATABASE%T_RESULT
[2004-02-28 02:14:26.572] : Parsing: =
[2004-02-28 02:14:26.652] : Parsing: NEGATIVE%STRING
[2004-02-28 02:14:26.733] : Parsing: DSU
[2004-02-28 02:14:26.893] : Parsing: RESULT
[2004-02-28 02:14:26.973] : Parsing: DATABASE
[2004-02-28 02:14:27.053] : Parsing: T_RESULT
[2004-02-28 02:14:27.133] : Parsing: NEGATIVE
[2004-02-28 02:14:27.203] : Parsing: STRING
[2004-02-28 02:14:27.283] : Parsing: DO
[2004-02-28 02:14:27.363] : Parsing:  PATIENT_STATE('microalbuminuria_screening' )
[2004-02-28 02:14:27.434] : Parsing: PATIENT_STATE
[2004-02-28 02:14:27.514] : Parsing: 'microalbuminuria_screening' )
[2004-02-28 02:14:27.604] : Parsing: 'microalbuminuria_screening'
[2004-02-28 02:14:27.734] : Schedule : AUS
[2004-02-28 02:14:27.834] : No. of Schedule Static Rules: 0
[2004-02-28 02:14:27.904] : No. of Schedule Dynamic Rules: 2
[2004-02-28 02:14:27.984] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:28.195] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:28.395] : Parsing: OIS
[2004-02-28 02:14:28.465] : Parsing: DESCRIPTION: This is a microalbuminuria protocol schedule called OIS for SCREENING OTHER INFECTIONS in the diagnosis of microalbuminuria and proteinuria
[2004-02-28 02:14:28.545] : Parsing: DESCRIPTION
[2004-02-28 02:14:28.645] : Parsing:  This is a microalbuminuria protocol schedule called OIS for SCREENING OTHER INFECTIONS in the diagnosis of microalbuminuria and proteinuria
[2004-02-28 02:14:28.806] : Parsing: RULE OIS2
[2004-02-28 02:14:28.886] : Parsing: RULE
[2004-02-28 02:14:28.996] : Parsing: OIS2
[2004-02-28 02:14:29.066] : Parsing: DESCRIPTION
[2004-02-28 02:14:29.156] : Parsing:  if UTI is not present then perform  24 hour creatinine and 24 hour protein loss tests
[2004-02-28 02:14:29.236] : Parsing: ON
[2004-02-28 02:14:29.336] : Parsing:  result_arrival('UTI')
[2004-02-28 02:14:29.406] : Parsing: result_arrival
[2004-02-28 02:14:29.497] : Parsing: 'UTI')
[2004-02-28 02:14:29.577] : Parsing: 'UTI'
[2004-02-28 02:14:29.657] : Parsing: IF
[2004-02-28 02:14:29.747] : Parsing: IF
[2004-02-28 02:14:29.827] : Parsing:  UTI%result%database%t_result  =  negative%string
[2004-02-28 02:14:29.907] : Parsing: UTI%result%database%t_result
[2004-02-28 02:14:29.997] : Parsing: =
[2004-02-28 02:14:30.087] : Parsing: negative%string
[2004-02-28 02:14:30.178] : Parsing: UTI
[2004-02-28 02:14:30.268] : Parsing: result
[2004-02-28 02:14:30.358] : Parsing: database
[2004-02-28 02:14:30.438] : Parsing: t_result
[2004-02-28 02:14:30.558] : Parsing: negative
[2004-02-28 02:14:30.678] : Parsing: string
[2004-02-28 02:14:30.778] : Parsing: DO
[2004-02-28 02:14:30.859] : Parsing:  order_test('24CRCL_PL' )
[2004-02-28 02:14:30.939] : Parsing: order_test
[2004-02-28 02:14:31.019] : Parsing: '24CRCL_PL' )
[2004-02-28 02:14:31.099] : Parsing: '24CRCL_PL'
[2004-02-28 02:14:31.179] : Parsing: RULE
[2004-02-28 02:14:31.329] : Parsing: OIS3
[2004-02-28 02:14:31.409] : Parsing: DESCRIPTION
[2004-02-28 02:14:31.479] : Parsing:  if UTI is present then place back on annual screening
[2004-02-28 02:14:31.56] : Parsing: ON
[2004-02-28 02:14:31.64] : Parsing:  result_arrival('UTI')
[2004-02-28 02:14:31.72] : Parsing: result_arrival
[2004-02-28 02:14:31.8] : Parsing: 'UTI')
[2004-02-28 02:14:31.87] : Parsing: 'UTI'
[2004-02-28 02:14:31.95] : Parsing: IF
[2004-02-28 02:14:32.02] : Parsing: IF
[2004-02-28 02:14:32.1] : Parsing:  UTI%result%database%t_result  =  positive%string
[2004-02-28 02:14:32.18] : Parsing: UTI%result%database%t_result
[2004-02-28 02:14:32.261] : Parsing: =
[2004-02-28 02:14:32.341] : Parsing: positive%string
[2004-02-28 02:14:32.431] : Parsing: UTI
[2004-02-28 02:14:32.581] : Parsing: result
[2004-02-28 02:14:32.661] : Parsing: database
[2004-02-28 02:14:32.741] : Parsing: t_result
[2004-02-28 02:14:32.861] : Parsing: positive
[2004-02-28 02:14:32.942] : Parsing: string
[2004-02-28 02:14:33.022] : Parsing: DO
[2004-02-28 02:14:33.102] : Parsing:  patient_state( 'annual_urine_screening')
[2004-02-28 02:14:33.172] : Parsing: patient_state
[2004-02-28 02:14:33.252] : Parsing:  'annual_urine_screening')
[2004-02-28 02:14:33.322] : Parsing: 'annual_urine_screening'
[2004-02-28 02:14:33.402] : Parsing: RULE
[2004-02-28 02:14:33.562] : Parsing: OIS4
[2004-02-28 02:14:33.633] : Parsing: DESCRIPTION
[2004-02-28 02:14:33.723] : Parsing:  if 24 hour creatine clearance and 24 hour protein loss tests are positive then proteinuria is confirmed and refer patient to nephrologist
[2004-02-28 02:14:33.823] : Parsing: ON
[2004-02-28 02:14:33.893] : Parsing:  result_arrival('24CRCL_PL')
[2004-02-28 02:14:33.973] : Parsing: result_arrival
[2004-02-28 02:14:34.053] : Parsing: '24CRCL_PL')
[2004-02-28 02:14:34.133] : Parsing: '24CRCL_PL'
[2004-02-28 02:14:34.213] : Parsing: IF
[2004-02-28 02:14:34.293] : Parsing: IF
[2004-02-28 02:14:34.374] : Parsing:  24CRCL_PL%RESULT%DATABASE%T_TEST = POSITIVE%STRING
[2004-02-28 02:14:34.534] : Parsing: 24CRCL_PL%RESULT%DATABASE%T_TEST
[2004-02-28 02:14:34.614] : Parsing: =
[2004-02-28 02:14:34.704] : Parsing: POSITIVE%STRING
[2004-02-28 02:14:34.794] : Parsing: 24CRCL_PL
[2004-02-28 02:14:34.874] : Parsing: RESULT
[2004-02-28 02:14:34.954] : Parsing: DATABASE
[2004-02-28 02:14:35.035] : Parsing: T_TEST
[2004-02-28 02:14:35.115] : Parsing: POSITIVE
[2004-02-28 02:14:35.195] : Parsing: STRING
[2004-02-28 02:14:35.275] : Parsing: DO
[2004-02-28 02:14:35.355] : Parsing:  patient_state ('nephrology_referral')
[2004-02-28 02:14:35.435] : Parsing: patient_state
[2004-02-28 02:14:35.505] : Parsing: 'nephrology_referral')
[2004-02-28 02:14:35.585] : Parsing: 'nephrology_referral'
[2004-02-28 02:14:35.665] : Parsing: RULE
[2004-02-28 02:14:35.806] : Parsing: 0IS5
[2004-02-28 02:14:35.886] : Parsing: DESCRIPTION
[2004-02-28 02:14:35.966] : Parsing:  if 24 hour creatine clearance and 24 hour protein loss is negative then return patient to annual screening
[2004-02-28 02:14:36.036] : Parsing: ON
[2004-02-28 02:14:36.116] : Parsing:  result_arrival('24CRCL_PL')
[2004-02-28 02:14:36.276] : Parsing: result_arrival
[2004-02-28 02:14:36.346] : Parsing: '24CRCL_PL')
[2004-02-28 02:14:36.417] : Parsing: '24CRCL_PL'
[2004-02-28 02:14:36.497] : Parsing: IF
[2004-02-28 02:14:36.567] : Parsing: IF
[2004-02-28 02:14:36.657] : Parsing:  24CRCL_PL%RESULT%DATABASE%T_TEST = NEGATIVE%STRING
[2004-02-28 02:14:36.747] : Parsing: 24CRCL_PL%RESULT%DATABASE%T_TEST
[2004-02-28 02:14:36.827] : Parsing: =
[2004-02-28 02:14:36.897] : Parsing: NEGATIVE%STRING
[2004-02-28 02:14:36.967] : Parsing: 24CRCL_PL
[2004-02-28 02:14:37.047] : Parsing: RESULT
[2004-02-28 02:14:37.128] : Parsing: DATABASE
[2004-02-28 02:14:37.208] : Parsing: T_TEST
[2004-02-28 02:14:37.278] : Parsing: NEGATIVE
[2004-02-28 02:14:37.358] : Parsing: STRING
[2004-02-28 02:14:37.428] : Parsing: DO
[2004-02-28 02:14:37.508] : Parsing:  patient_state ('annual_urine_screening')
[2004-02-28 02:14:37.578] : Parsing: patient_state
[2004-02-28 02:14:37.658] : Parsing: 'annual_urine_screening')
[2004-02-28 02:14:37.738] : Parsing: 'annual_urine_screening'
[2004-02-28 02:14:37.829] : Schedule : OIS
[2004-02-28 02:14:37.979] : No. of Schedule Static Rules: 0
[2004-02-28 02:14:38.139] : No. of Schedule Dynamic Rules: 4
[2004-02-28 02:14:38.219] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:38.52] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:38.73] : Parsing: MAS
[2004-02-28 02:14:38.81] : Parsing: DESCRIPTION: This is a microalbuminuria protocol schedule called MAS for the screening of microalbuminuria
[2004-02-28 02:14:38.88] : Parsing: DESCRIPTION
[2004-02-28 02:14:38.96] : Parsing:  This is a microalbuminuria protocol schedule called MAS for the screening of microalbuminuria
[2004-02-28 02:14:39.03] : Parsing: RULE MAS2
[2004-02-28 02:14:39.11] : Parsing: RULE
[2004-02-28 02:14:39.211] : Parsing: MAS2
[2004-02-28 02:14:39.291] : Parsing: DESCRIPTION
[2004-02-28 02:14:39.451] : Parsing:  if the first ACR result is > 20 mg/l order two more tests within the next six months
[2004-02-28 02:14:39.521] : Parsing: ON
[2004-02-28 02:14:39.601] : Parsing:  result_arrival('ACR')
```

```
[2004-02-28 02:14:39.681] : Parsing: result_arrival
[2004-02-28 02:14:39.751] : Parsing: 'ACR')
[2004-02-28 02:14:39.831] : Parsing: 'ACR'
[2004-02-28 02:14:39.902] : Parsing: IF
[2004-02-28 02:14:39.982] : Parsing: IF
[2004-02-28 02:14:40.062] : Parsing:  ACR%RESULT%DATABASE%T_RESULTS > 20%DOUBLE
[2004-02-28 02:14:40.132] : Parsing: ACR%RESULT%DATABASE%T_RESULTS
[2004-02-28 02:14:40.222] : Parsing: >
[2004-02-28 02:14:40.292] : Parsing: 20%DOUBLE
[2004-02-28 02:14:40.362] : Parsing: ACR
[2004-02-28 02:14:40.442] : Parsing: RESULT
[2004-02-28 02:14:40.522] : Parsing: DATABASE
[2004-02-28 02:14:40.603] : Parsing: T_RESULTS
[2004-02-28 02:14:40.703] : Parsing: 20
[2004-02-28 02:14:40.783] : Parsing: DOUBLE
[2004-02-28 02:14:40.863] : Parsing: DO
[2004-02-28 02:14:40.943] : Parsing:  ADD_RULE
[2004-02-28 02:14:41.183] : Parsing:  'ACR' )}
[2004-02-28 02:14:41.263] : Parsing: 'ACR'
[2004-02-28 02:14:41.394] : DO: ADD_RULE
[2004-02-28 02:14:41.474] : Parsing: ADD_RULE
[2004-02-28 02:14:41.554] : Parsing: STATIC_RULE  [2004-02-28 02:14:41.644] : Parsing:
STATIC_RULE
[2004-02-28 02:14:41.724] : Parsing: STATIC_RULE  MAS2a
[2004-02-28 02:14:41.804] : Parsing: STATIC_RULE
[2004-02-28 02:14:41.884] : Parsing: MAS2a
[2004-02-28 02:14:41.954] : Parsing: DESCRIPTION
[2004-02-28 02:14:42.035] : Parsing:  rule orders ACR test during the next 6 month period
[2004-02-28 02:14:42.135] : Parsing: FROM time_rule_added
[2004-02-28 02:14:42.215] : Parsing: FROM time_rule_added
[2004-02-28 02:14:42.295] : Warning: Found <NOTHING> while expecting a value after "FROM"
[2004-02-28 02:14:42.545] : Parsing: STARTING now
[2004-02-28 02:14:42.625] : Parsing: STARTING now
[2004-02-28 02:14:42.706] : Warning: Found <NOTHING> while expecting a value after
"STARTING"
[2004-02-28 02:14:42.796] : Parsing: now
[2004-02-28 02:14:42.876] : Error: number expected instead of now
[2004-02-28 02:14:42.956] : Unexpected end of statement: parsing stopped
[2004-02-28 02:14:43.036] : 0  = 0 MilliSeconds
[2004-02-28 02:14:43.146] : Parsing: ENDING 6 months
[2004-02-28 02:14:43.226] : Parsing: ENDING 6 months
[2004-02-28 02:14:43.306] : Warning: Found <NOTHING> while expecting a value after "ENDING"
[2004-02-28 02:14:43.397] : Parsing: 6
[2004-02-28 02:14:43.547] : Parsing: months
[2004-02-28 02:14:43.627] : 6 months = 15552000000 MilliSeconds
[2004-02-28 02:14:43.727] : Parsing: ON_EVERY 3 months
[2004-02-28 02:14:43.807] : Parsing: ON_EVERY 3 months
[2004-02-28 02:14:43.887] : Warning: Found <NOTHING> while expecting a value after
"ON_EVERY"
[2004-02-28 02:14:43.977] : Parsing: 3
[2004-02-28 02:14:44.047] : Parsing: months
[2004-02-28 02:14:44.158] : 3 months = 7776000000 MilliSeconds
[2004-02-28 02:14:44.228] : Parsing: DO order_test ( 'ACR' )
[2004-02-28 02:14:44.318] : Parsing: DO order_test ( 'ACR' )
[2004-02-28 02:14:44.398] : Warning: Found <NOTHING> while expecting a value after "DO"
[2004-02-28 02:14:44.498] : Parsing: order_test
[2004-02-28 02:14:44.588] : Parsing:  'ACR' )
[2004-02-28 02:14:44.668] : Parsing: 'ACR'
[2004-02-28 02:14:44.839] : ADDED RULE SPEC:
'MAS2','ADD_RULE*MAS2/STATIC/MAS2alnull|time_rule_added|0|15552000000|7776000000|OR
DER_TEST;"ACR";|rule orders ACR test during the next 6 month period/*'
[2004-02-28 02:14:44.919] : ACTION: ADD_RULE
[2004-02-28 02:14:45.009] : ACTION PARAMETERS:
'MAS2','ADD_RULE*MAS2/STATIC/MAS2alnull|time_rule_added|0|15552000000|7776000000|OR
DER_TEST;"ACR";|rule orders ACR test during the next 6 month period/*'
[2004-02-28 02:14:45.089] : parsed ACTION:
ADD_RULE('MAS2','ADD_RULE*MAS2/STATIC/MAS2alnull|time_rule_added|0|15552000000|777
6000000|ORDER_TEST;"ACR";|rule orders ACR test during the next 6 month period/*')
[2004-02-28 02:14:45.179] : Parsing: RULE
[2004-02-28 02:14:45.349] : Parsing: MAS3
[2004-02-28 02:14:45.429] : Parsing: DESCRIPTION
[2004-02-28 02:14:45.52] : Parsing:  if ACR < 20 mg/l then place patient on annual screening
[2004-02-28 02:14:45.6] : Parsing: ON
[2004-02-28 02:14:45.69] : Parsing:  result_arrival('ACR')
[2004-02-28 02:14:45.76] : Parsing: result_arrival
[2004-02-28 02:14:45.84] : Parsing: 'ACR')
[2004-02-28 02:14:45.91] : Parsing: 'ACR'
[2004-02-28 02:14:45.99] : Parsing: IF
[2004-02-28 02:14:46.07] : Parsing: IF
[2004-02-28 02:14:46.14] : Parsing:  ACR%RESULT%DATABASE%T_RESULTS > 20%DOUBLE
[2004-02-28 02:14:46.231] : Parsing: ACR%RESULT%DATABASE%T_RESULTS
[2004-02-28 02:14:46.301] : Parsing: >
[2004-02-28 02:14:46.381] : Parsing: 20%DOUBLE
[2004-02-28 02:14:46.461] : Parsing: ACR
[2004-02-28 02:14:46.611] : Parsing: RESULT
[2004-02-28 02:14:46.691] : Parsing: DATABASE
[2004-02-28 02:14:46.771] : Parsing: T_RESULTS
[2004-02-28 02:14:46.852] : Parsing: 20
[2004-02-28 02:14:46.922] : Parsing: DOUBLE
[2004-02-28 02:14:47.002] : Parsing: DO
[2004-02-28 02:14:47.082] : Parsing:  PATIENT_STATE('annual_urine_screening')
[2004-02-28 02:14:47.152] : Parsing: PATIENT_STATE
[2004-02-28 02:14:47.242] : Parsing: 'annual_urine_screening')
[2004-02-28 02:14:47.312] : Parsing: 'annual_urine_screening'
[2004-02-28 02:14:47.402] : Parsing: RULE
[2004-02-28 02:14:47.553] : Parsing: MAS4
[2004-02-28 02:14:47.633] : Parsing: DESCRIPTION
[2004-02-28 02:14:47.723] : Parsing:  if 2 of 3 ACR in 20-200 mg/l within 6 months then
microalbuminuria is confirmed
[2004-02-28 02:14:47.803] : Parsing: ON
[2004-02-28 02:14:47.883] : Parsing:  result_arrival('ACR')
[2004-02-28 02:14:47.963] : Parsing: result_arrival
[2004-02-28 02:14:48.043] : Parsing: 'ACR')
[2004-02-28 02:14:48.113] : Parsing: 'ACR'
[2004-02-28 02:14:48.193] : Parsing: DO
[2004-02-28 02:14:48.344] : Parsing: DO
[2004-02-28 02:14:48.434] : Parsing:  2_OF_3_ACR_CHECK ('ACR')
[2004-02-28 02:14:48.504] : Parsing: 2_OF_3_ACR_CHECK
[2004-02-28 02:14:48.604] : Parsing: 'ACR')
[2004-02-28 02:14:48.684] : Parsing: 'ACR'
[2004-02-28 02:14:48.764] : Parsing: RULE
[2004-02-28 02:14:48.914] : Parsing: MAS5
[2004-02-28 02:14:49.005] : Parsing: DESCRIPTION
[2004-02-28 02:14:49.085] : Parsing:  if ACR > 200 mg/l then refer patient to nephrologist for possible
proteinuria
[2004-02-28 02:14:49.165] : Parsing: ON
[2004-02-28 02:14:49.245] : Parsing:  RESULT_ARRIVAL('ACR')
[2004-02-28 02:14:49.335] : Parsing: RESULT_ARRIVAL
[2004-02-28 02:14:49.415] : Parsing: 'ACR')
[2004-02-28 02:14:49.515] : Parsing: 'ACR'
[2004-02-28 02:14:49.595] : Parsing: IF
```

```
[2004-02-28 02:14:49.676] : Parsing: IF
[2004-02-28 02:14:49.756] : Parsing:  ACR%RESULT%DATABASE%T_TEST  > 200%DOUBLE
[2004-02-28 02:14:49.836] : Parsing: ACR%RESULT%DATABASE%T_TEST
[2004-02-28 02:14:49.916] : Parsing: >
[2004-02-28 02:14:50.006] : Parsing: 200%DOUBLE
[2004-02-28 02:14:50.076] : Parsing: ACR
[2004-02-28 02:14:50.246] : Parsing: RESULT
[2004-02-28 02:14:50.316] : Parsing: DATABASE
[2004-02-28 02:14:50.397] : Parsing: T_TEST
[2004-02-28 02:14:50.467] : Parsing: 200
[2004-02-28 02:14:50.557] : Parsing: DOUBLE
[2004-02-28 02:14:50.627] : Parsing: DO
[2004-02-28 02:14:50.707] : Parsing:  PATIENT_STATE('nephrology_referral' )
[2004-02-28 02:14:50.787] : Parsing: PATIENT_STATE
[2004-02-28 02:14:50.867] : Parsing: 'nephrology_referral' )
[2004-02-28 02:14:50.947] : Parsing: 'nephrology_referral'
[2004-02-28 02:14:51.028] : Schedule : MAS
[2004-02-28 02:14:51.178] : No. of Schedule Static Rules: 0
[2004-02-28 02:14:51.268] : No. of Schedule Dynamic Rules: 4
[2004-02-28 02:14:51.338] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:51.558] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:51.769] : Parsing: CMA
[2004-02-28 02:14:51.849] : Parsing: DESCRIPTION: This is a microalbuminuria protocol schedule
named CMA for confirmed microalbuminuria – handles treatment and control of microalbuminuria
[2004-02-28 02:14:52.009] : Parsing: DESCRIPTION
[2004-02-28 02:14:52.079] : Parsing:  This is a microalbuminuria protocol schedule named CMA for
confirmed microalbuminuria – handles treatment and control of microalbuminuria
[2004-02-28 02:14:52.159] : Parsing: RULE CMA5
[2004-02-28 02:14:52.229] : Parsing: RULE
[2004-02-28 02:14:52.379] : Parsing: CMA5
[2004-02-28 02:14:52.47] : Parsing: DESCRIPTION
[2004-02-28 02:14:52.54] : Parsing:  if becomes normal (ACR < 20 mg/l) at any time  then the patient
is placed on annual screening
[2004-02-28 02:14:52.63] : Parsing: ON
[2004-02-28 02:14:52.72] : Parsing:  result_arrival('ACR')
[2004-02-28 02:14:52.8] : Parsing: result_arrival
[2004-02-28 02:14:52.88] : Parsing: 'ACR')
[2004-02-28 02:14:52.96] : Parsing: 'ACR'
[2004-02-28 02:14:53.07] : Parsing: IF
[2004-02-28 02:14:53.151] : Parsing: IF
[2004-02-28 02:14:53.271] : Parsing:  ACR%RESULT%DATABASE%T_RESULT  <  20%DOUBLE
[2004-02-28 02:14:53.351] : Parsing: ACR%RESULT%DATABASE%T_RESULT
[2004-02-28 02:14:53.421] : Parsing: <
[2004-02-28 02:14:53.501] : Parsing: 20%DOUBLE
[2004-02-28 02:14:53.581] : Parsing: ACR
[2004-02-28 02:14:53.661] : Parsing: RESULT
[2004-02-28 02:14:53.741] : Parsing: DATABASE
[2004-02-28 02:14:53.832] : Parsing: T_RESULT
[2004-02-28 02:14:53.982] : Parsing: 20
[2004-02-28 02:14:54.062] : Parsing: DOUBLE
[2004-02-28 02:14:54.142] : Parsing: DO
[2004-02-28 02:14:54.232] : Parsing:  PATIENT_STATE('annual_urine_screening' )
[2004-02-28 02:14:54.302] : Parsing: PATIENT_STATE
[2004-02-28 02:14:54.382] : Parsing: 'annual_urine_screening' )
[2004-02-28 02:14:54.462] : Parsing: 'annual_urine_screening'
[2004-02-28 02:14:54.543] : Parsing: RULE
[2004-02-28 02:14:54.703] : Parsing: CMA6
[2004-02-28 02:14:54.803] : Parsing: DESCRIPTION
[2004-02-28 02:14:54.883] : Parsing:  if becomes abnormal (ACR > 200 mg/l) at any time  then the
patient  is placed on nephrology referral
[2004-02-28 02:14:54.963] : Parsing: ON
[2004-02-28 02:14:55.033] : Parsing:  result_arrival('ACR')
[2004-02-28 02:14:55.103] : Parsing: result_arrival
[2004-02-28 02:14:55.183] : Parsing: 'ACR')
[2004-02-28 02:14:55.264] : Parsing: 'ACR'
[2004-02-28 02:14:55.334] : Parsing: IF
[2004-02-28 02:14:55.414] : Parsing: IF
[2004-02-28 02:14:55.494] : Parsing:  ACR%RESULT%DATABASE%T_RESULT >
200%DOUBLE
[2004-02-28 02:14:55.564] : Parsing: ACR%RESULT%DATABASE%T_RESULT
[2004-02-28 02:14:55.644] : Parsing: >
[2004-02-28 02:14:55.724] : Parsing: 200%DOUBLE
[2004-02-28 02:14:55.794] : Parsing: ACR
[2004-02-28 02:14:55.955] : Parsing: RESULT
[2004-02-28 02:14:56.035] : Parsing: DATABASE
[2004-02-28 02:14:56.105] : Parsing: T_RESULT
[2004-02-28 02:14:56.185] : Parsing: 200
[2004-02-28 02:14:56.255] : Parsing: DOUBLE
[2004-02-28 02:14:56.335] : Parsing: DO
[2004-02-28 02:14:56.405] : Parsing:  PATIENT_STATE('nephrology_referral' )
[2004-02-28 02:14:56.485] : Parsing: PATIENT_STATE
[2004-02-28 02:14:56.555] : Parsing: 'nephrology_referral' )
[2004-02-28 02:14:56.636] : Parsing: 'nephrology_referral'
[2004-02-28 02:14:56.796] : Schedule : CMA
[2004-02-28 02:14:56.866] : No. of Schedule Static Rules: 0
[2004-02-28 02:14:56.946] : No. of Schedule Dynamic Rules: 2
[2004-02-28 02:14:57.016] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:57.266] : Checking if SCHEDULE [ID = 0] exists ...
[2004-02-28 02:14:57.477] : Parsing: NPH
[2004-02-28 02:14:57.547] : Parsing: DESCRIPTION: This is a microalbuminuria protocol schedule
named NPH for nephrology referral – handles preparation and transmission of the necessary
documentation for the referral
[2004-02-28 02:14:57.647] : Parsing: DESCRIPTION
[2004-02-28 02:14:57.737] : Parsing:  This is a microalbuminuria protocol schedule named NPH for
nephrology referral – handles preparation and transmission of the necessary documentation for the
referral
[2004-02-28 02:14:57.897] : Parsing: RULE NPH2
[2004-02-28 02:14:57.998] : Parsing: RULE
[2004-02-28 02:14:58.078] : Parsing: NPH2
[2004-02-28 02:14:58.158] : Parsing: DESCRIPTION
[2004-02-28 02:14:58.238] : Parsing:  when a referral note is created it must immediately be sent to the
specialist either by post or e-mail
[2004-02-28 02:14:58.318] : Parsing: ON
[2004-02-28 02:14:58.398] : Parsing:  new_referral_note()
[2004-02-28 02:14:58.488] : Parsing: new_referral_note
[2004-02-28 02:14:58.568] : Parsing: )
[2004-02-28 02:14:58.648] : Parsing: DO
[2004-02-28 02:14:58.729] : Parsing: DO
[2004-02-28 02:14:58.829] : Parsing:  send_referral_note()
[2004-02-28 02:14:58.909] : Parsing: send_referral_note
[2004-02-28 02:14:58.999] : Parsing: )
[2004-02-28 02:14:59.079] : Schedule [NPH] has no rules. It should not be declared.
[2004-02-28 02:14:59.219] : Schedule : NPH
[2004-02-28 02:14:59.299] : No. of Schedule Static Rules: 0
[2004-02-28 02:14:59.37] : No. of Schedule Dynamic Rules: 1
[2004-02-28 02:14:59.45] : Parsing: END SCHEDULE_SET
[2004-02-28 02:14:59.73] : Parsing: ~RULE_SET~
 [2004-02-28 02:14:59.82] : Parsing: STATIC_RULE
[2004-02-28 02:14:59.89] : Parsing: AUS1
[2004-02-28 02:15:00.071] : Parsing: FROM
[2004-02-28 02:15:00.151] : Parsing: FROM
```

[2004-02-28 02:15:00.231] : Parsing: annual_screening_start_date
[2004-02-28 02:15:00.321] : Parsing: STARTING
[2004-02-28 02:15:00.391] : Parsing: 0 year
[2004-02-28 02:15:00.471] : Parsing: 0
[2004-02-28 02:15:00.551] : Parsing: year
[2004-02-28 02:15:00.711] : 0 year = 0 MilliSeconds
[2004-02-28 02:15:00.782] : Parsing: ENDING
[2004-02-28 02:15:00.862] : Parsing: 1 year
[2004-02-28 02:15:00.942] : Parsing: 1
[2004-02-28 02:15:01.022] : Parsing: year
[2004-02-28 02:15:01.092] : 1 year = 31536000000 MilliSeconds
[2004-02-28 02:15:01.172] : Parsing: ON_EVERY
[2004-02-28 02:15:01.252] : Parsing: 1 year
[2004-02-28 02:15:01.322] : Parsing: 1
[2004-02-28 02:15:01.392] : Parsing: year
[2004-02-28 02:15:01.483] : 1 year = 31536000000 MilliSeconds
[2004-02-28 02:15:01.563] : Parsing: DO
[2004-02-28 02:15:01.643] : Parsing: order_test('DSU')
[2004-02-28 02:15:01.743] : Parsing: order_test
[2004-02-28 02:15:01.823] : Parsing: 'DSU')
[2004-02-28 02:15:01.903] : Parsing: 'DSU'
[2004-02-28 02:15:01.983] : Parsing: RULE
[2004-02-28 02:15:02.164] : Parsing: OIS1
[2004-02-28 02:15:02.244] : Parsing: DESCRIPTION
[2004-02-28 02:15:02.324] : Parsing: on entry to the OIS schedule the patient is tested for other urinary tract inections (UTI)
[2004-02-28 02:15:02.404] : Parsing: ON
[2004-02-28 02:15:02.564] : Parsing: state_change()
[2004-02-28 02:15:02.644] : Parsing: state_change
[2004-02-28 02:15:02.724] : Parsing:)
[2004-02-28 02:15:02.814] : Parsing: IF
[2004-02-28 02:15:02.895] : Parsing: IF
[2004-02-28 02:15:02.975] : Parsing: state_name%patient_state%database%tops_patient_state = other_infections_screening%string
[2004-02-28 02:15:03.045] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:03.125] : Parsing: =
[2004-02-28 02:15:03.215] : Parsing: other_infections_screening%string
[2004-02-28 02:15:03.305] : Parsing: state_name
[2004-02-28 02:15:03.375] : Parsing: patient_state
[2004-02-28 02:15:03.455] : Parsing: database
[2004-02-28 02:15:03.525] : Parsing: tops_patient_state
[2004-02-28 02:15:03.596] : Parsing: other_infections_screening
[2004-02-28 02:15:03.676] : Parsing: string
[2004-02-28 02:15:03.756] : Parsing: DO
[2004-02-28 02:15:03.836] : Parsing: order_test ('UTI')
[2004-02-28 02:15:03.906] : Parsing: order_test
[2004-02-28 02:15:03.986] : Parsing: 'UTI')
[2004-02-28 02:15:04.066] : Parsing: 'UTI'
[2004-02-28 02:15:04.136] : Parsing: RULE
[2004-02-28 02:15:04.367] : Parsing: MAS1a
[2004-02-28 02:15:04.447] : Parsing: DESCRIPTION
[2004-02-28 02:15:04.527] : Parsing: at the start of this schedule MAS order the two ACR and SCR tests
[2004-02-28 02:15:04.597] : Parsing: ON
[2004-02-28 02:15:04.677] : Parsing: state_change()
[2004-02-28 02:15:04.757] : Parsing: state_change
[2004-02-28 02:15:04.847] : Parsing:)
[2004-02-28 02:15:04.928] : Parsing: IF
[2004-02-28 02:15:04.998] : Parsing: IF
[2004-02-28 02:15:05.078] : Parsing: state_name%patient_state%database%tops_patient_state = microalbuminuria_screening%string
[2004-02-28 02:15:05.148] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:05.228] : Parsing: =
[2004-02-28 02:15:05.298] : Parsing: microalbuminuria_screening%string
[2004-02-28 02:15:05.378] : Parsing: state_name
[2004-02-28 02:15:05.468] : Parsing: patient_state
[2004-02-28 02:15:05.538] : Parsing: database
[2004-02-28 02:15:05.618] : Parsing: tops_patient_state
[2004-02-28 02:15:05.689] : Parsing: microalbuminuria_screening
[2004-02-28 02:15:05.769] : Parsing: string
[2004-02-28 02:15:05.849] : Parsing: DO
[2004-02-28 02:15:05.919] : Parsing: order_test('ACR')
[2004-02-28 02:15:05.999] : Parsing: order_test
[2004-02-28 02:15:06.069] : Parsing: 'ACR')
[2004-02-28 02:15:06.249] : Parsing: 'ACR'
[2004-02-28 02:15:06.33] : Parsing: RULE
[2004-02-28 02:15:06.48] : Parsing: MAS1b
[2004-02-28 02:15:06.56] : Parsing: DESCRIPTION
[2004-02-28 02:15:06.64] : Parsing: at the start of this schedule MAS order the two ACR and SCR tests
[2004-02-28 02:15:06.72] : Parsing: ON
[2004-02-28 02:15:06.8] : Parsing: state_change()
[2004-02-28 02:15:06.88] : Parsing: state_change
[2004-02-28 02:15:06.96] : Parsing:)
[2004-02-28 02:15:07.041] : Parsing: IF
[2004-02-28 02:15:07.121] : Parsing: IF
[2004-02-28 02:15:07.201] : Parsing: state_name%patient_state%database%tops_patient_state = microalbuminuria_screening%string
[2004-02-28 02:15:07.281] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:07.361] : Parsing: =
[2004-02-28 02:15:07.441] : Parsing: microalbuminuria_screening%string
[2004-02-28 02:15:07.521] : Parsing: state_name
[2004-02-28 02:15:07.601] : Parsing: patient_state
[2004-02-28 02:15:07.691] : Parsing: database
[2004-02-28 02:15:07.772] : Parsing: tops_patient_state
[2004-02-28 02:15:07.852] : Parsing: microalbuminuria_screening
[2004-02-28 02:15:07.922] : Parsing: string
[2004-02-28 02:15:08.072] : Parsing: DO
[2004-02-28 02:15:08.152] : Parsing: order_test('SCR')
[2004-02-28 02:15:08.232] : Parsing: order_test
[2004-02-28 02:15:08.312] : Parsing: 'SCR')
[2004-02-28 02:15:08.392] : Parsing: 'SCR'
[2004-02-28 02:15:08.463] : Parsing: RULE
[2004-02-28 02:15:08.623] : Parsing: CMA1
[2004-02-28 02:15:08.703] : Parsing: DESCRIPTION
[2004-02-28 02:15:08.793] : Parsing: at the start of this schedule suggest optimisation of glycaemic control
[2004-02-28 02:15:08.863] : Parsing: ON
[2004-02-28 02:15:08.943] : Parsing: state_change()
[2004-02-28 02:15:09.023] : Parsing: state_change
[2004-02-28 02:15:09.093] : Parsing:)
[2004-02-28 02:15:09.174] : Parsing: IF
[2004-02-28 02:15:09.244] : Parsing: IF
[2004-02-28 02:15:09.324] : Parsing: state_name%patient_state%database%tops_patient_state = confirmed_microalbuminuria%string
[2004-02-28 02:15:09.394] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:09.474] : Parsing: =
[2004-02-28 02:15:09.544] : Parsing: confirmed_microalbuminuria%string
[2004-02-28 02:15:09.624] : Parsing: state_name
[2004-02-28 02:15:09.704] : Parsing: patient_state
[2004-02-28 02:15:09.865] : Parsing: database
[2004-02-28 02:15:09.935] : Parsing: tops_patient_state
[2004-02-28 02:15:10.015] : Parsing: confirmed_microalbuminuria
[2004-02-28 02:15:10.085] : Parsing: string
[2004-02-28 02:15:10.165] : Parsing: DO
[2004-02-28 02:15:10.235] : Parsing: suggest ('optimisation_of_glycaemic_control')
[2004-02-28 02:15:10.315] : Parsing: suggest
[2004-02-28 02:15:10.385] : Parsing: 'optimisation_of_glycaemic_control')
[2004-02-28 02:15:10.465] : Parsing: 'optimisation_of_glycaemic_control'
[2004-02-28 02:15:10.546] : Parsing: RULE
[2004-02-28 02:15:10.726] : Parsing: CMA2
[2004-02-28 02:15:10.816] : Parsing: DESCRIPTION
[2004-02-28 02:15:10.886] : Parsing: at the start of this schedule suggest BP measurement
[2004-02-28 02:15:10.976] : Parsing: ON
[2004-02-28 02:15:11.046] : Parsing: state_change()
[2004-02-28 02:15:11.126] : Parsing: state_change
[2004-02-28 02:15:11.197] : Parsing:)
[2004-02-28 02:15:11.277] : Parsing: IF
[2004-02-28 02:15:11.357] : Parsing: IF
[2004-02-28 02:15:11.437] : Parsing: state_name%patient_state%database%tops_patient_state = confirmed_microalbuminuria%string
[2004-02-28 02:15:11.517] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:11.597] : Parsing: =
[2004-02-28 02:15:11.687] : Parsing: confirmed_microalbuminuria%string
[2004-02-28 02:15:11.847] : Parsing: state_name
[2004-02-28 02:15:11.928] : Parsing: patient_state
[2004-02-28 02:15:12.008] : Parsing: database
[2004-02-28 02:15:12.078] : Parsing: tops_patient_state
[2004-02-28 02:15:12.168] : Parsing: confirmed_microalbuminuria
[2004-02-28 02:15:12.248] : Parsing: string
[2004-02-28 02:15:12.328] : Parsing: DO
[2004-02-28 02:15:12.398] : Parsing: ORDER_TEST ('BP')
[2004-02-28 02:15:12.478] : Parsing: ORDER_TEST
[2004-02-28 02:15:12.558] : Parsing: 'BP')
[2004-02-28 02:15:12.629] : Parsing: 'BP'
[2004-02-28 02:15:12.719] : Parsing: RULE
[2004-02-28 02:15:12.869] : Parsing: CMA3
[2004-02-28 02:15:12.949] : Parsing: DESCRIPTION
[2004-02-28 02:15:13.029] : Parsing: If patient suffers from diabetes type 1 then prescribe ACE inhibitor
[2004-02-28 02:15:13.109] : Parsing: ON
[2004-02-28 02:15:13.179] : Parsing: state_change()
[2004-02-28 02:15:13.259] : Parsing: state_change
[2004-02-28 02:15:13.34] : Parsing:)
[2004-02-28 02:15:13.42] : Parsing: IF
[2004-02-28 02:15:13.57] : Parsing: IF
[2004-02-28 02:15:13.65] : Parsing: state_name%patient_state%database%tops_patient_state = confirmed_microalbuminuria%string
[2004-02-28 02:15:13.78] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:13.85] : Parsing: =
[2004-02-28 02:15:13.93] : Parsing: confirmed_microalbuminuria%string
[2004-02-28 02:15:14.001] : Parsing: state_name
[2004-02-28 02:15:14.081] : Parsing: patient_state
[2004-02-28 02:15:14.151] : Parsing: database
[2004-02-28 02:15:14.231] : Parsing: tops_patient_state
[2004-02-28 02:15:14.301] : Parsing: confirmed_microalbuminuria
[2004-02-28 02:15:14.391] : Parsing: string
[2004-02-28 02:15:14.461] : Parsing: DO
[2004-02-28 02:15:14.531] : Parsing: prescribe_medication('ACE_inhibitor')
[2004-02-28 02:15:14.601] : Parsing: prescribe_medication
[2004-02-28 02:15:14.682] : Parsing: 'ACE_inhibitor')
[2004-02-28 02:15:14.772] : Parsing: 'ACE_inhibitor'
[2004-02-28 02:15:14.852] : Parsing: RULE
[2004-02-28 02:15:15.002] : Parsing: CMA4a
[2004-02-28 02:15:15.082] : Parsing: DESCRIPTION
[2004-02-28 02:15:15.182] : Parsing: ACR and SCR tests are performed every month for all microalbuminuria patients
[2004-02-28 02:15:15.272] : Parsing: ON
[2004-02-28 02:15:15.433] : Parsing: state_change()
[2004-02-28 02:15:15.523] : Parsing: state_change
[2004-02-28 02:15:15.593] : Parsing:)
[2004-02-28 02:15:15.683] : Parsing: IF
[2004-02-28 02:15:15.773] : Parsing: IF
[2004-02-28 02:15:15.853] : Parsing: state_name%patient_state%database%tops_patient_state = confirmed_microalbuminuria%string
[2004-02-28 02:15:15.933] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:16.003] : Parsing: =
[2004-02-28 02:15:16.094] : Parsing: confirmed_microalbuminuria%string
[2004-02-28 02:15:16.174] : Parsing: state_name
[2004-02-28 02:15:16.254] : Parsing: patient_state
[2004-02-28 02:15:16.334] : Parsing: database
[2004-02-28 02:15:16.404] : Parsing: tops_patient_state
[2004-02-28 02:15:16.484] : Parsing: confirmed_microalbuminuria
[2004-02-28 02:15:16.564] : Parsing: string
[2004-02-28 02:15:16.654] : Parsing: DO
[2004-02-28 02:15:16.724] : Parsing: order_test ('ACR')
[2004-02-28 02:15:16.815] : Parsing: order_test
[2004-02-28 02:15:16.885] : Parsing: 'ACR')
[2004-02-28 02:15:16.965] : Parsing: 'ACR'
[2004-02-28 02:15:17.035] : Parsing: RULE
[2004-02-28 02:15:17.245] : Parsing: CMA4b
[2004-02-28 02:15:17.325] : Parsing: DESCRIPTION
[2004-02-28 02:15:17.405] : Parsing: ACR and SCR tests are performed every month for all microalbuminuria patients
[2004-02-28 02:15:17.476] : Parsing: ON
[2004-02-28 02:15:17.546] : Parsing: state_change()
[2004-02-28 02:15:17.626] : Parsing: state_change
[2004-02-28 02:15:17.706] : Parsing:)
[2004-02-28 02:15:17.776] : Parsing: IF
[2004-02-28 02:15:17.866] : Parsing: IF
[2004-02-28 02:15:17.946] : Parsing: state_name%patient_state%database%tops_patient_state = confirmed_microalbuminuria%string
[2004-02-28 02:15:18.016] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:18.086] : Parsing: =
[2004-02-28 02:15:18.167] : Parsing: confirmed_microalbuminuria%string
[2004-02-28 02:15:18.237] : Parsing: state_name
[2004-02-28 02:15:18.317] : Parsing: patient_state
[2004-02-28 02:15:18.397] : Parsing: database
[2004-02-28 02:15:18.467] : Parsing: tops_patient_state
[2004-02-28 02:15:18.547] : Parsing: confirmed_microalbuminuria
[2004-02-28 02:15:18.627] : Parsing: string
[2004-02-28 02:15:18.707] : Parsing: DO
[2004-02-28 02:15:18.787] : Parsing: order_test ('SCR')
[2004-02-28 02:15:18.878] : Parsing: order_test
[2004-02-28 02:15:19.028] : Parsing: 'SCR')
[2004-02-28 02:15:19.098] : Parsing: 'SCR'
[2004-02-28 02:15:19.178] : Parsing: RULE
[2004-02-28 02:15:19.328] : Parsing: NPH1
[2004-02-28 02:15:19.398] : Parsing: DESCRIPTION
[2004-02-28 02:15:19.498] : Parsing: when a patient is referred to a specialist a patient referral note is created
[2004-02-28 02:15:19.579] : Parsing: ON

```
[2004-02-28 02:15:19.659] : Parsing:  state_change()
[2004-02-28 02:15:19.739] : Parsing: state_change
[2004-02-28 02:15:19.819] : Parsing: )
[2004-02-28 02:15:19.909] : Parsing: IF
[2004-02-28 02:15:19.979] : Parsing: IF
[2004-02-28 02:15:20.059] : Parsing:  state_name%patient_state%database%tops_patient_state =
confirmed_microalbuminuria%string
[2004-02-28 02:15:20.139] : Parsing: state_name%patient_state%database%tops_patient_state
[2004-02-28 02:15:20.219] : Parsing: =
[2004-02-28 02:15:20.3] : Parsing: confirmed_microalbuminuria%string
[2004-02-28 02:15:20.38] : Parsing: state_name
[2004-02-28 02:15:20.45] : Parsing: patient_state
[2004-02-28 02:15:20.53] : Parsing: database
[2004-02-28 02:15:20.61] : Parsing: tops_patient_state
[2004-02-28 02:15:20.69] : Parsing: confirmed_microalbuminuria
[2004-02-28 02:15:20.85] : Parsing: string
[2004-02-28 02:15:20.931] : Parsing: DO
[2004-02-28 02:15:21.021] : Parsing:  create_referral_note ('nephrologist' )
[2004-02-28 02:15:21.091] : Parsing: create_referral_note
[2004-02-28 02:15:21.171] : Parsing: 'nephrologist' )
[2004-02-28 02:15:21.251] : Parsing: 'nephrologist'
[2004-02-28 02:15:21.441] : Both schedule and protocol rule sets are present in the protocol.
[2004-02-28 02:15:21.521] : Protocol Specification after parsing
------------------------------------------------------
PROTOCOL_NAME: MAP2;
DESCRIPTION: This is a protocol for the diagnosis and management of microalbuminuria in diabetes
patients;
DATE_CREATED: 2004-02-28 02:15:21.441;
CREATOR_ID: 3;
CATEGORY_ID: 1;

BEGIN SCHEDULE_SET

BEGIN SCHEDULE;
SCHEDULE_NAME: AUS;
DESCRIPTION: This is a microalbuminuria protocol schedule called AUS for Annual dipstick Urine
Screening;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: AUS2;
DESCRIPTION: no description;
ON: result_arrival('DSU');
IF: DSU = positive;
DO: PATIENT_STATE('other_infections_screening');
RULE_NAME: AUS3;
DESCRIPTION: no description;
ON: result_arrival('DSU');
IF: DSU = NEGATIVE;
DO: PATIENT_STATE('microalbuminuria_screening');
END SCHEDULE_RULE_SET;

END SCHEDULE;

BEGIN SCHEDULE;
SCHEDULE_NAME: OIS;
DESCRIPTION: This is a microalbuminuria protocol schedule called OIS for SCREENING OTHER
INFECTIONS in the diagnosis of microalbuminuria and proteinuria;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: OIS2;
DESCRIPTION: no description;
ON: result_arrival('UTI');
IF: UTI = negative;
DO: ORDER_TEST('24CRCL_PL');
RULE_NAME: OIS3;
DESCRIPTION: no description;
ON: result_arrival('UTI');
IF: UTI = positive;
DO: PATIENT_STATE('annual_urine_screening');
RULE_NAME: OIS4;
DESCRIPTION: no description;
ON: result_arrival('24CRCL_PL');
IF: 24CRCL_PL = POSITIVE;
DO: PATIENT_STATE('nephrology_referral');
RULE_NAME: 0IS5;
DESCRIPTION: no description;
ON: result_arrival('24CRCL_PL');
IF: 24CRCL_PL = NEGATIVE;
DO: PATIENT_STATE('annual_urine_screening');
END SCHEDULE_RULE_SET;

END SCHEDULE;

BEGIN SCHEDULE;
SCHEDULE_NAME: MAS;
DESCRIPTION: This is a microalbuminuria protocol schedule called MAS for the screening of
microalbuminuria;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: MAS2;
DESCRIPTION: no description;
ON: result_arrival('ACR');
IF: ACR > 20.0;
DO:
ADD_RULE('MAS2','ADD_RULE*MAS2/STATIC/MAS2a|null|time_rule_added|0|1555200000|777
6000000|ORDER_TEST;"ACR";|rule orders ACR test during the next 6 month period/*');
RULE_NAME: MAS3;
DESCRIPTION: no description;
ON: result_arrival('ACR');
IF: ACR > 20.0;
DO: PATIENT_STATE('annual_urine_screening');
RULE_NAME: MAS4;
DESCRIPTION: no description;
ON: result_arrival('ACR');
DO: 2_OF_3_ACR_CHECK('ACR');
RULE_NAME: MAS5;
DESCRIPTION: no description;
ON: RESULT_ARRIVAL('ACR');
IF: ACR > 200.0;
DO: PATIENT_STATE('nephrology_referral');
END SCHEDULE_RULE_SET;

END SCHEDULE;

BEGIN SCHEDULE;
SCHEDULE_NAME: CMA;
DESCRIPTION: This is a microalbuminuria protocol schedule named CMA for confirmed
microalbuminuria – handles treatment and control of microalbuminuria;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: CMA5;
DESCRIPTION: no description;
ON: result_arrival('ACR');
IF: ACR < 20.0;
DO: PATIENT_STATE('annual_urine_screening');
RULE_NAME: CMA6;
DESCRIPTION: no description;
ON: result_arrival('ACR');
IF: ACR > 200.0;
DO: PATIENT_STATE('nephrology_referral');
END SCHEDULE_RULE_SET;

END SCHEDULE;

BEGIN SCHEDULE;
SCHEDULE_NAME: NPH;
DESCRIPTION: This is a microalbuminuria protocol schedule named NPH for nephrology referral –
handles preparation and transmission of the necessary documentation for the referral;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: NPH2;
DESCRIPTION: no description;
ON: new_referral_note();
DO: SEND_REFERRAL_NOTE();
END SCHEDULE_RULE_SET;

END SCHEDULE;
END SCHEDULE_SET
BEGIN PROTOCOL_RULE_SET;
RULE_NAME: OIS1;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = other_infections_screening;
DO: ORDER_TEST('UTI');
RULE_NAME: MAS1a;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = microalbuminuria_screening;
DO: ORDER_TEST('ACR');
RULE_NAME: MAS1b;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = microalbuminuria_screening;
DO: ORDER_TEST('SCR');
RULE_NAME: CMA1;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = confirmed_microalbuminuria;
DO: SUGGEST('optimisation_of_glycaemic_control');
RULE_NAME: CMA2;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = confirmed_microalbuminuria;
DO: ORDER_TEST('BP');
RULE_NAME: CMA3;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = confirmed_microalbuminuria;
DO: PRESCRIBE_MEDICATION('ACE_inhibitor');
RULE_NAME: CMA4a;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = confirmed_microalbuminuria;
DO: ORDER_TEST('ACR');
RULE_NAME: CMA4b;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = confirmed_microalbuminuria;
DO: ORDER_TEST('SCR');
RULE_NAME: NPH1;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = confirmed_microalbuminuria;
DO: CREATE_REFERRAL_NOTE('nephrologist');
END PROTOCOL_RULE_SET;
END PROTOCOL.
------------------------------------------------------
[2004-02-28 02:15:22.042] : Parsing protocol specification completed.
[2004-02-28 02:15:22.112] : No. of Schedules: 5
[2004-02-28 02:15:22.192] : No. of Protocol dynamic rules: 9
[2004-02-28 02:15:22.272] : No. of Protocol static rules: 9
[2004-02-28 02:15:23.694] : Protocol [MAP2] inserted into database.[ID: 1]
[2004-02-28 02:15:23.775] : Adding protocol schedules to database ...
[2004-02-28 02:15:23.865] : SCHEDULE: [name: AUS]
[spec: null][description: This is a microalbuminuria protocol schedule called AUS for Annual dipstick
Urine Screening] [creatorID: 3] [dateCreated: 2004-02-28 02:14:27.734]
[2004-02-28 02:15:23.945] : Adding schedule AUS to database.
[2004-02-28 02:15:24.606] : Schedule [AUS] added to database.
[2004-02-28 02:15:29.433] : ACTION [PATIENT_STATE] added to database.
[2004-02-28 02:15:32.397] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:15:33.859] : ACTION [PATIENT_STATE] already exists in database.
[2004-02-28 02:15:35.281] : SCHEDULE: [name: OIS]
[spec: null][description: This is a microalbuminuria protocol schedule called OIS for SCREENING
OTHER INFECTIONS in the diagnosis of microalbuminuria and proteinuria] [creatorID: 3]
[dateCreated: 2004-02-28 02:14:37.829]
[2004-02-28 02:15:35.351] : Adding schedule OIS to database.
[2004-02-28 02:15:35.922] : Schedule [OIS] added to database.
[2004-02-28 02:15:37.805] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:15:39.828] : ACTION [ORDER_TEST] added to database.
[2004-02-28 02:15:42.091] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:15:43.493] : ACTION [PATIENT_STATE] already exists in database.
[2004-02-28 02:15:46.167] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:15:47.569] : ACTION [PATIENT_STATE] already exists in database.
[2004-02-28 02:15:50.133] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:15:51.434] : ACTION [PATIENT_STATE] already exists in database.
[2004-02-28 02:15:52.917] : SCHEDULE: [name: MAS]
[spec: null][description: This is a microalbuminuria protocol schedule called MAS for the screening of
microalbuminuria] [creatorID: 3] [dateCreated: 2004-02-28 02:14:51.028]
[2004-02-28 02:15:52.987] : Adding schedule MAS to database.
[2004-02-28 02:15:53.457] : Schedule [MAS] added to database.
[2004-02-28 02:15:55.24] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:15:57.233] : ACTION [ADD_RULE] added to database.
[2004-02-28 02:15:59.566] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:16:00.878] : ACTION [PATIENT_STATE] already exists in database.
[2004-02-28 02:16:03.422] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:16:04.653] : ACTION [2_OF_3_ACR_CHECK] added to database.
[2004-02-28 02:16:08.208] : ACTION [PATIENT_STATE] already exists in database.
[2004-02-28 02:16:09.821] : SCHEDULE: [name: CMA]
[spec: null][description: This is a microalbuminuria protocol schedule named CMA for confirmed
microalbuminuria – handles treatment and control of microalbuminuria] [creatorID: 3] [dateCreated:
2004-02-28 02:14:56.786]
[2004-02-28 02:16:09.891] : Adding schedule CMA to database.
[2004-02-28 02:16:10.352] : Schedule [CMA] added to database.
[2004-02-28 02:16:12.204] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:16:13.586] : ACTION [PATIENT_STATE] already exists in database.
[2004-02-28 02:16:16.18] : EVENT [result_arrival] already exists in database.
[2004-02-28 02:16:17.472] : ACTION [PATIENT_STATE] already exists in database.
```

```
[2004-02-28 02:16:19.064] : SCHEDULE: [name: NPH]
[spec: null][description: This is a microalbuminuria protocol schedule named NPH for nephrology
referral – handles preparation and transmission of the necessary documentation for the referral]
[creatorID: 3] [dateCreated: 2004-02-28 02:14:59.219]
[2004-02-28 02:16:19.134] : Adding schedule NPH to database.
[2004-02-28 02:16:19.515] : Schedule [NPH] added to database.
[2004-02-28 02:16:22.659] : ACTION [SEND_REFERRAL_NOTE] added to database.
[2004-02-28 02:16:23.49] : Adding protocol rules to database ...
[2004-02-28 02:16:26.234] : ACTION [ORDER_TEST] already exists in database.
[2004-02-28 02:16:28.908] : EVENT [state_change] already exists in database.
[2004-02-28 02:16:30.37] : ACTION [ORDER_TEST] already exists in database.
[2004-02-28 02:16:33.024] : EVENT [state_change] already exists in database.
[2004-02-28 02:16:34.737] : ACTION [ORDER_TEST] already exists in database.
[2004-02-28 02:16:37.25] : EVENT [state_change] already exists in database.
[2004-02-28 02:16:39.043] : ACTION [SUGGEST] added to database.
[2004-02-28 02:16:41.266] : EVENT [state_change] already exists in database.
[2004-02-28 02:16:42.748] : ACTION [ORDER_TEST] already exists in database.
[2004-02-28 02:16:45.272] : EVENT [state_change] already exists in database.
[2004-02-28 02:16:47.124] : ACTION [PRESCRIBE_MEDICATION] added to database.
[2004-02-28 02:16:49.428] : EVENT [state_change] already exists in database.
[2004-02-28 02:16:50.72] : ACTION [ORDER_TEST] already exists in database.
[2004-02-28 02:16:53.424] : EVENT [state_change] already exists in database.
[2004-02-28 02:16:54.785] : ACTION [ORDER_TEST] already exists in database.
[2004-02-28 02:16:57.299] : EVENT [state_change] already exists in database.
[2004-02-28 02:16:59.332] : ACTION [CREATE_REFERRAL_NOTE] added to database.
[2004-02-28 02:17:00.594] : Adding protocol static rules to database ...
[2004-02-28 02:17:01.505] : ACTION [ORDER_TEST] already exists in database.
[2004-02-28 02:17:03.228] : ACTION [ORDER_TEST] already exists in database.
[2004-02-28 02:17:03.688] : Protocol [MAP2] saved to file: D:\TOPS\specs\MAP2_1077934623648
```

.protocol]

E. The MAP Specification as Stored in the TOPS Database

This appendix presents figures that illustrate how the specification for the **M**icro**A**lbuminuria **P**rotocol (MAP) is stored in the TOPS database, a relational database implemented in the Oracle9i database system. The figures present queries and the results of these queries on relational tables that hold the attributes of the protocol specification.

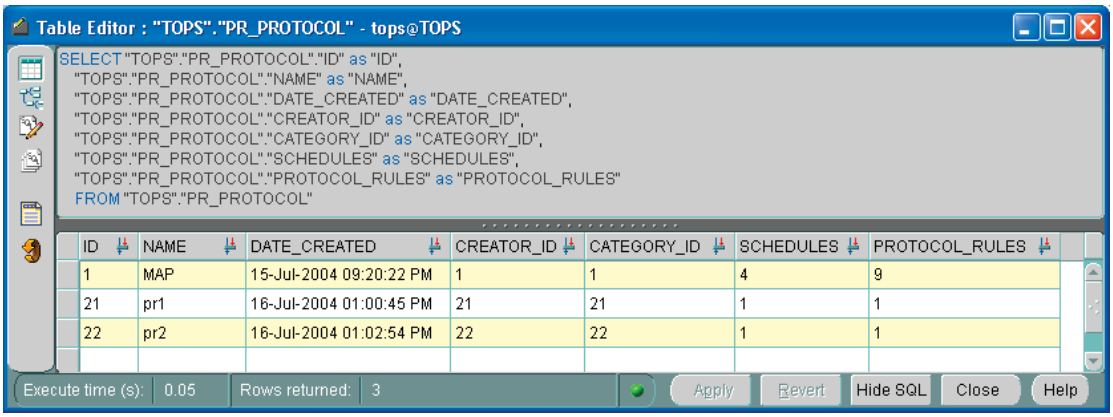


Figure 78 Attributes of protocol specificateons in the TOPS database

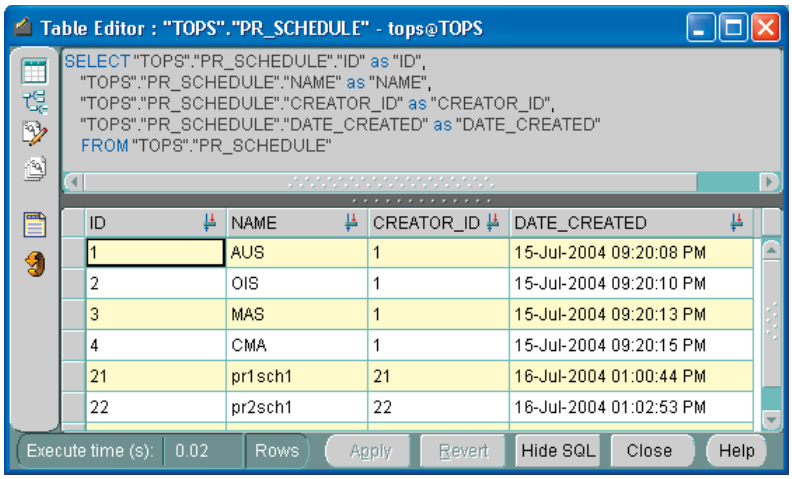


Figure 79 Schedule sspecifications in the MAP as stored in TOPS

Table Editor : "TOPS"."PR_RULE" - tops@TOPS

```
SELECT "TOPS"."PR_RULE"."ID" as "ID", "TOPS"."PR_RULE"."NAME" as "NAME", "TOPS"."PR_RULE"."RULE_T
  "TOPS"."PR_RULE"."DATE_CREATED" as "DATE_CREATED" FROM "TOPS"."PR_RULE"
```

ID	NAME	RULE_TYPE	CREATOR_ID	DATE_CREATED
1	AUS2	SCHEDULE	1	15-Jul-2004 09:20:07 PM
2	AUS3	SCHEDULE	1	15-Jul-2004 09:20:08 PM
3	OIS2	SCHEDULE	1	15-Jul-2004 09:20:09 PM
4	OIS3	SCHEDULE	1	15-Jul-2004 09:20:09 PM
5	OIS4	SCHEDULE	1	15-Jul-2004 09:20:10 PM
6	OIS5	SCHEDULE	1	15-Jul-2004 09:20:10 PM
7	MAS2	SCHEDULE	1	15-Jul-2004 09:20:12 PM
8	MAS3	SCHEDULE	1	15-Jul-2004 09:20:13 PM
9	MAS4	SCHEDULE	1	15-Jul-2004 09:20:13 PM
10	MAS5	SCHEDULE	1	15-Jul-2004 09:20:13 PM
11	CMA5	SCHEDULE	1	15-Jul-2004 09:20:15 PM
12	CMA6	SCHEDULE	1	15-Jul-2004 09:20:15 PM
13	OIS1	PROTOCOL	1	15-Jul-2004 09:20:17 PM
14	MAS1a	PROTOCOL	1	15-Jul-2004 09:20:17 PM
15	MAS1b	PROTOCOL	1	15-Jul-2004 09:20:18 PM
16	CMA1	PROTOCOL	1	15-Jul-2004 09:20:18 PM
17	CMA2	PROTOCOL	1	15-Jul-2004 09:20:19 PM
18	CMA3	PROTOCOL	1	15-Jul-2004 09:20:20 PM
19	CMA4a	PROTOCOL	1	15-Jul-2004 09:20:21 PM
20	CMA4b	PROTOCOL	1	15-Jul-2004 09:20:21 PM
21	NPH1	PROTOCOL	1	15-Jul-2004 09:20:22 PM
22	AUS1	STATIC	1	15-Jul-2004 09:20:16 PM
41	pr1sr1	STATIC	21	16-Jul-2004 01:00:44 PM
42	pr1sdr1	SCHEDULE	21	16-Jul-2004 01:00:44 PM
43	pr1dr1	PROTOCOL	21	16-Jul-2004 01:00:45 PM
44	pr2sr1	STATIC	22	16-Jul-2004 01:02:51 PM
45	pr2sdr1	SCHEDULE	22	16-Jul-2004 01:02:52 PM
46	pr2sdr2	SCHEDULE	22	16-Jul-2004 01:02:53 PM

Execute time (s): 0.041 Rows returned: 2 Apply Revert Show SQL Close Help

Figure 80 Protocol rule specifications for the MAP in the TOPS database

Table Editor : "TOPS"."PR_DYNAMIC_RULE" - tops@TOPS

```
select "TOPS"."PR_DYNAMIC_RULE"."ID", "TOPS"."PR_DYNAMIC_RULE"."EVENT_ID",
 "TOPS"."PR_DYNAMIC_RULE"."EVENT_PARAMETERS",   "TOPS"."PR_DYNAMIC_RULE"."RULE_TYPE"
  from "TOPS"."PR_DYNAMIC_RULE"
```

ID	EVENT_ID	EVENT_PARAMETERS	RULE_TYPE
1	1	'DSU'	SCHEDULE
2	1	'DSU'	SCHEDULE
3	1	'UTI'	SCHEDULE
4	1	'UTI'	SCHEDULE
5	1	'24CRCL_PL'	SCHEDULE
6	1	'24CRCL_PL'	SCHEDULE
7	1	'ACR'	SCHEDULE
8	1	'ACR'	SCHEDULE
9	1	'ACR'	SCHEDULE
10	2	'ACR'	SCHEDULE
11	1	'ACR'	SCHEDULE
12	1	'ACR'	SCHEDULE
13	3		PROTOCOL
14	3		PROTOCOL
15	3		PROTOCOL
16	3		PROTOCOL
17	3		PROTOCOL
18	3		PROTOCOL
19	3		PROTOCOL
20	3		PROTOCOL
21	3		PROTOCOL
42	1	'A'	SCHEDULE
43	1	'B'	PROTOCOL
45	1	'A'	SCHEDULE
46	1	'A'	SCHEDULE
47	1	'B'	PROTOCOL

Figure 81 The specification of MAP rules of the dynamic rule type in the TOPS database

Table Editor : "TOPS"."PR_STATIC_RULE" - tops@TOPS

```
select "TOPS"."PR_STATIC_RULE"."ID",
  "TOPS"."PR_STATIC_RULE"."ZERO_TIME_REF_TERM",
  "TOPS"."PR_STATIC_RULE"."START_TIME",
  "TOPS"."PR_STATIC_RULE"."END_TIME",
  "TOPS"."PR_STATIC_RULE"."INTERVAL"
  from "TOPS"."PR_STATIC_RULE"
```

ID	ZERO_TIME_REF_TERM	START_TIME	END_TIME	INTERVAL
22	annual_screening_start_date	0	60000	60000
41	start_of_protocol	60000	18000000	180000
44	start_of_protocol	60000	18000000	300000

Execute time (s): 0.1 | Apply | Revert | Show SQL | Close | Help

Figure 82 The specification of MAP rules of the static rule type in the TOPS database

Table Editor : "TOPS"."PR_EVENT" - tops@TOPS

```
SELECT "TOPS"."PR_EVENT"."ID" as "ID", "TOPS"."PR_EVENT"."NAME"
  as "NAME", "TOPS"."PR_EVENT"."DATE_CREATED" as "DATE_CREATED"
  FROM "TOPS"."PR_EVENT"
```

ID	NAME	DATE_CREATED
1	result_arrival	15-Jul-2004 09:20:07 PM
2	RESULT_ARRIVAL	15-Jul-2004 09:20:13 PM
3	state_change	15-Jul-2004 09:20:16 PM

Execute time (s): | Apply | Revert | Show SQL | Close | Help

Figure 83 The attributes of event specifications for MAP rules in the TOPS database

Table Editor : "TOPS"."PR_CONDITION" - tops@TOPS

```
SELECT "TOPS"."PR_CONDITION"."ID" as "ID",  "TOPS"."PR_CONDITION"."ATTRIBUTE" as "ATTRIBUTE",  "TOPS"."PR_CONDITION"."ATTRIBUTE_ENTITY" as
"ATTRIBUTE_ENTITY", "TOPS"."PR_CONDITION"."SOURCE_TYPE" as  "SOURCE_TYPE", "TOPS"."PR_CONDITION"."SOURCE_NAME" as  "SOURCE_NAME",
"TOPS"."PR_CONDITION"."RIGHT_VALUE" as  "RIGHT_VALUE", "TOPS"."PR_CONDITION"."DATA_TYPE" as "DATA_TYPE", "TOPS"."PR_CONDITION"."COMPARATOR" as
"COMPARATOR", "TOPS"."PR_CONDITION"."DATE_CREATED" as  "DATE_CREATED"
 FROM "TOPS"."PR_CONDITION"
```

ID	ATTRIBUTE	ATTRIBUTE_ENTITY	SOURCE_TYPE	SOURCE_NAME	RIGHT_VALUE	DATA_TYPE	COMPARATOR	DATE_CREATED
1	DSU	result	database	t_results	1	DOUBLE	=	15-Jul-2004 09:20:27 F
2	DSU	RESULT	DATABASE	T_RESULTS	0	DOUBLE	=	15-Jul-2004 09:20:33 F
3	UTI	result	database	t_results	0	DOUBLE	=	15-Jul-2004 09:20:37 F
4	UTI	result	database	t_results	1	DOUBLE	=	15-Jul-2004 09:20:41 F
5	24CRCL_PL	RESULT	DATABASE	T_RESULTS	1	DOUBLE	=	15-Jul-2004 09:20:44 F
6	24CRCL_PL	RESULT	DATABASE	T_RESULTS	0	DOUBLE	=	15-Jul-2004 09:20:47 F
7	ACR	RESULT	DATABASE	T_RESULTS	20	DOUBLE	>	15-Jul-2004 09:20:52 F
8	ACR	RESULT	DATABASE	T_RESULTS	20	DOUBLE	>	15-Jul-2004 09:20:55 F
9	ACR	RESULT	DATABASE	T_RESULTS	200	DOUBLE	>	15-Jul-2004 09:21:01 F
10	ACR	RESULT	DATABASE	T_RESULTS	20	DOUBLE	<	15-Jul-2004 09:21:05 F
11	ACR	RESULT	DATABASE	T_RESULTS	200	DOUBLE	>	15-Jul-2004 09:21:09 F
12	state_name	patient_state	database	tops_patient_state	other_infections_screening	STRING	=	15-Jul-2004 09:21:12 F
13	state_name	patient_state	database	tops_patient_state	microalbuminuria_screening	STRING	=	15-Jul-2004 09:21:15 F
14	state_name	patient_state	database	tops_patient_state	microalbuminuria_screening	STRING	=	15-Jul-2004 09:21:19 F
15	state_name	patient_state	database	tops_patient_state	confirmed_microalbuminuria	STRING	=	15-Jul-2004 09:21:22 F
16	state_name	patient_state	database	tops_patient_state	confirmed_microalbuminuria	STRING	=	15-Jul-2004 09:21:25 F
17	state_name	patient_state	database	tops_patient_state	confirmed_microalbuminuria	STRING	=	15-Jul-2004 09:21:28 F
18	state_name	patient_state	database	tops_patient_state	confirmed_microalbuminuria	STRING	=	15-Jul-2004 09:21:32 F
19	state_name	patient_state	database	tops_patient_state	confirmed_microalbuminuria	STRING	=	15-Jul-2004 09:21:35 F
20	state_name	patient_state	database	tops_patient_state	confirmed_microalbuminuria	STRING	=	15-Jul-2004 09:21:38 F
21	confirmed_diagnosis	patient	database	tops_patients	DIABETES	STRING	=	16-Jul-2004 01:00:47 F

Figure 84 Condition specifications for the MAP as stored in the TOPS database

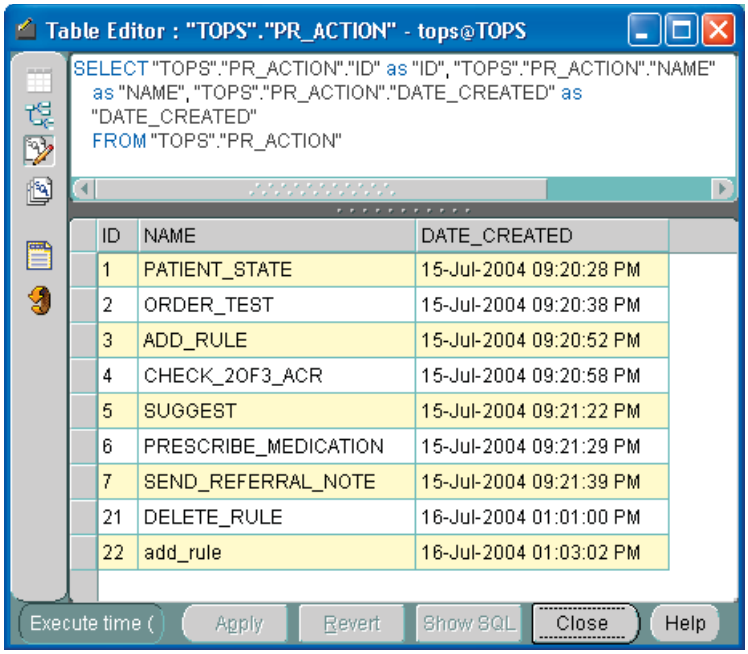


Figure 85 Core attributes of action specifications for the MAP in the TOPS database

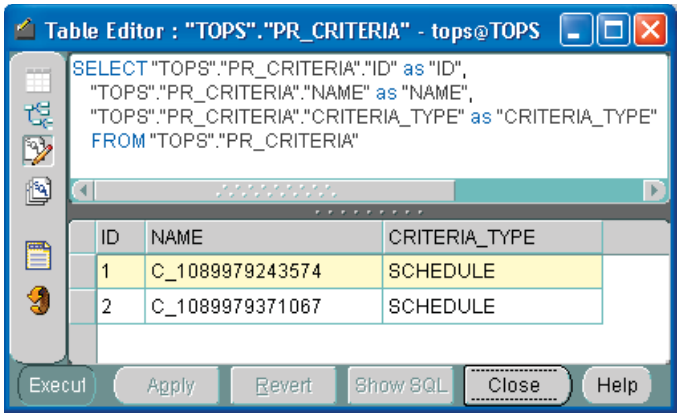


Figure 86 Entry criteria specification attributes for MAP in the TOPS database

Table Editor : "TOPS"."PR_RULE_ACTION" - tops@TOPS

```
select "TOPS"."PR_RULE_ACTION"."RULE_ID",
  "TOPS"."PR_RULE_ACTION"."ACTION_ID",
  "TOPS"."PR_RULE_ACTION"."ACTION_PARAMETERS"
  from "TOPS"."PR_RULE_ACTION"
```

RULE_ID	ACTION_ID	ACTION_PARAMETERS
1	1	'other_infections_screening'
2	1	'microalbuminuria_screening'
3	2	'24CRCL_PL'
4	1	'annual_urine_screening'
5	1	'nephrology_referral'
6	1	'annual_urine_screening'
7	3	'MAS2','ADD_RULE*MAS2/STATIC/MAS2a|
8	1	'annual_urine_screening'
9	4	
10	1	'nephrology_referral'
11	1	'annual_urine_screening'
12	1	'nephrology_referral'
13	2	'UTI'
14	2	'ACR'
15	2	'SCR'
16	5	'optimisation_of_glycaemic_control'
17	2	'BP'
18	6	'ACE_inhibitor'
19	2	'ACR'
20	2	'SCR'
21	7	'Nephrologist>Please examine this diabet
22	2	'DSU'

Figure 87 Rule-Action associations for the MAP in the TOPS database. NB: The parameters to a protocol action is an attribute of the rule-action relationship, hence why the relational table in this figure has the ACTION_PARAMETERS attribute.

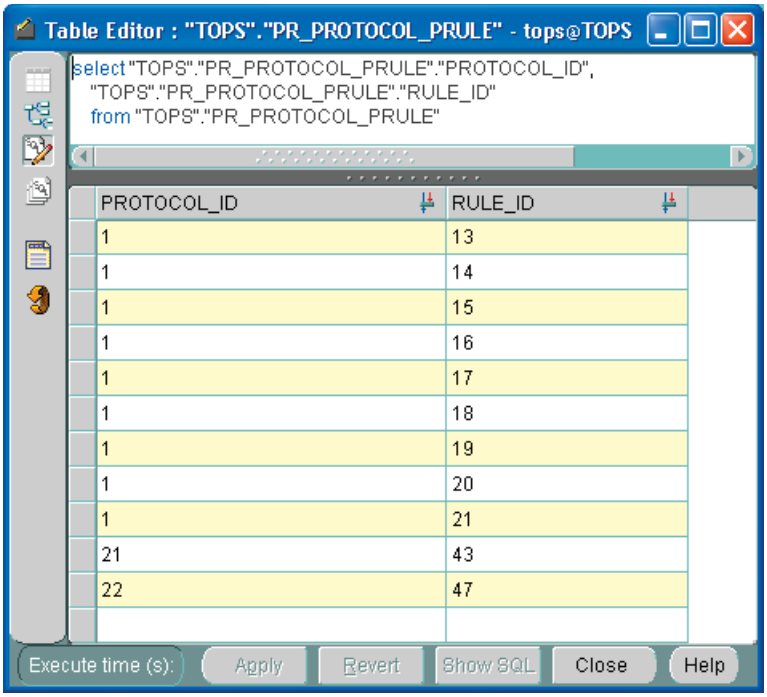


Figure 88 The Protocol-Rule relationship for the MAP

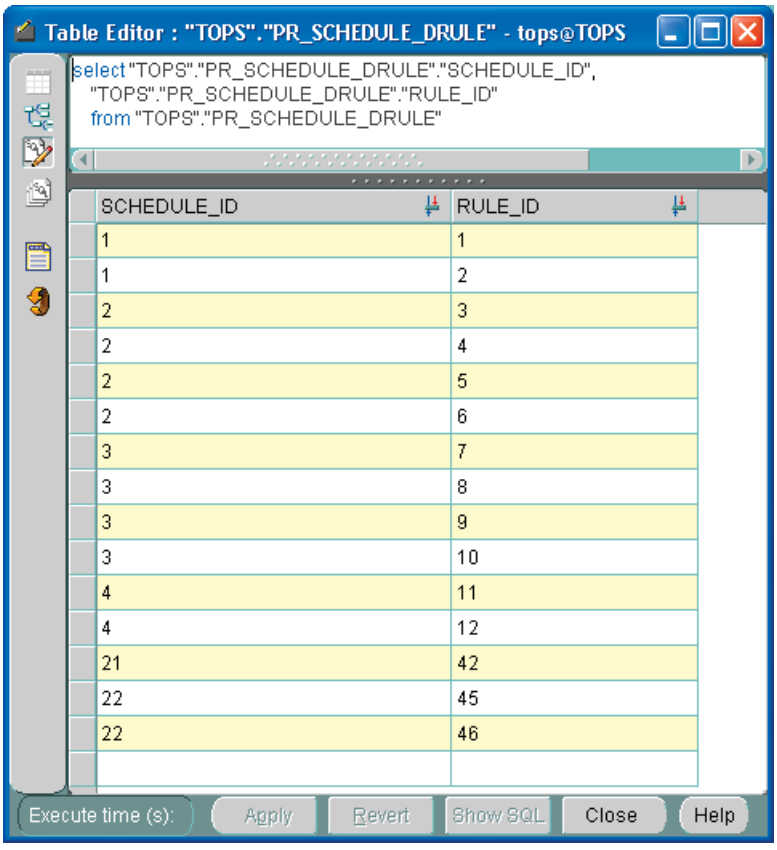


Figure 89 Schedule-Dynamic Rule relationship for the MAP

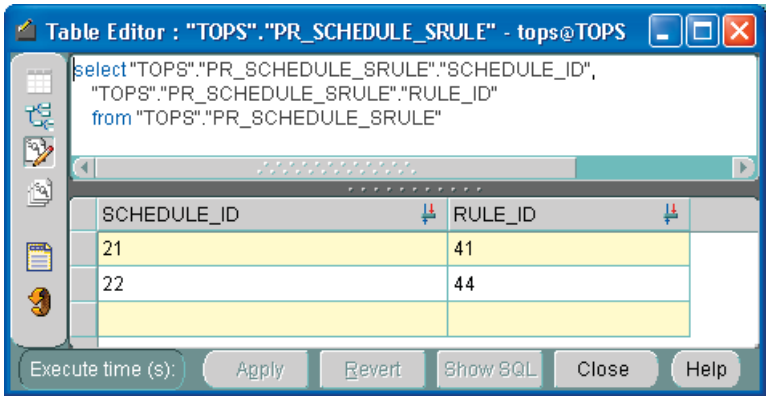


Figure 90 Schedule-Static Rule relationships for MAP in the TOPS database

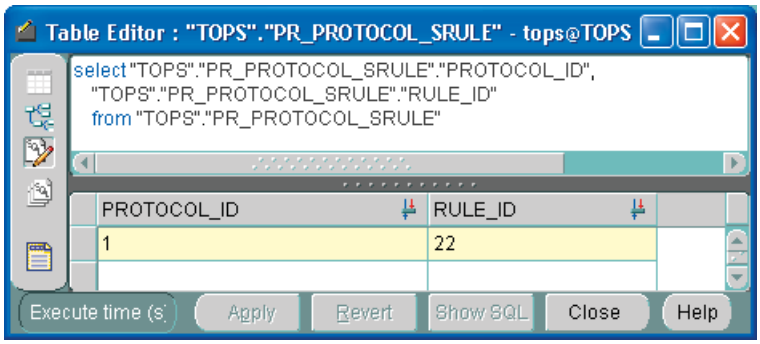


Figure 91 Protocol-Static Rule relationships for the MAP in the TOPS database

Table Editor : "TOPS"."PR_RULE_CONDITION" - tops@TOPS

```
select "TOPS"."PR_RULE_CONDITION"."RULE_ID",
  "TOPS"."PR_RULE_CONDITION"."CONDITION_ID"
  from "TOPS"."PR_RULE_CONDITION"
```

RULE_ID	CONDITION_ID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
10	9
11	10
12	11
13	12
14	13
15	14
16	15
17	16
18	17
19	18
20	19
21	20
42	22
43	23
45	25
46	26
47	27

Execute time (s: Apply Revert Show SQL Close Help

Figure 92 Rule-Condition relationships for the MAP in the TOPS database

Table Editor : "TOPS"."PR_CRITERIA_CONDITION" - tops@TOPS

```
select "TOPS"."PR_CRITERIA_CONDITION"."CRITERIA_ID",
  "TOPS"."PR_CRITERIA_CONDITION"."CONDITION_ID"
  from "TOPS"."PR_CRITERIA_CONDITION"
```

CRITERIA_ID	CONDITION_ID
1	21
2	24

Execute time (s): Apply Revert Show SQL Close Help

Figure 93 Criteria-Condition relationship for the MAP in the TOPS database

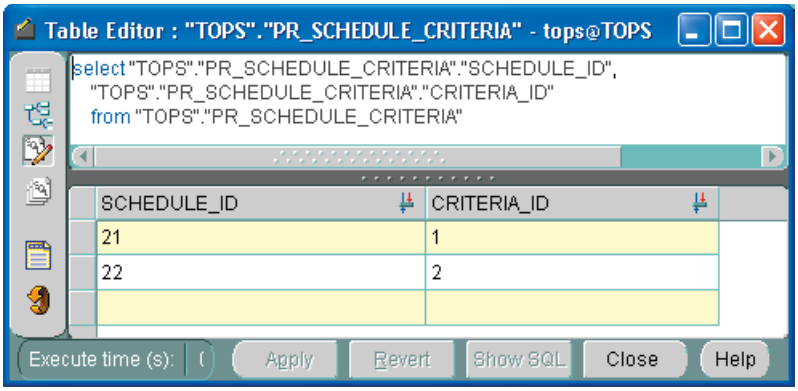


Figure 94 Schedule-Criteria relationships for the MAP in the TOPS database

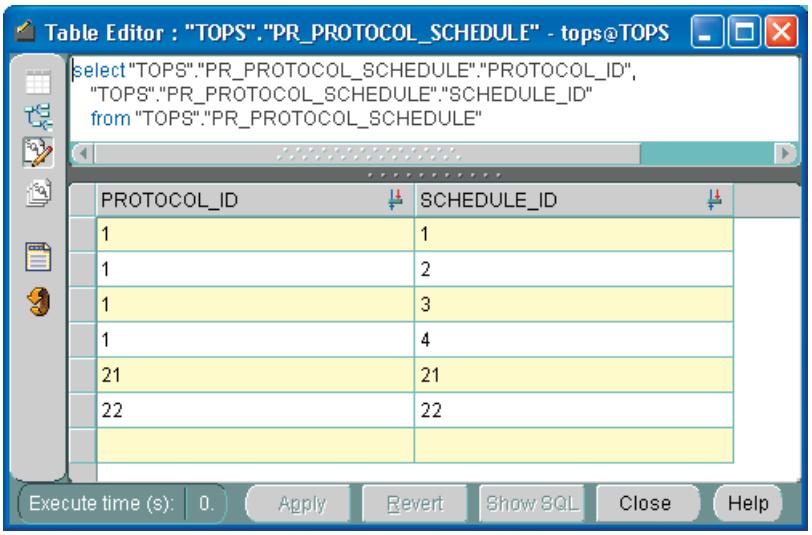


Figure 95 Protocol-Schedule relationships for the MAP in the TOPS database

F. TOPS Session for Creating a MAP Patient Plan

```
[2004-02-09 14:25:56.926] : TOPS System Execution Log: Session Starting at 2004-02-09 14:25:56.585
------------------------------------------------------------------
[2004-02-09 14:25:58.128] : Getting confirmation to create the TOPS database objects.
[2004-02-09 14:26:02.454] : TOPS rule execution listener activated ...
[2004-02-09 14:26:04.236] : Rule listener waiting ...
[2004-02-09 14:26:34.68] : Analysing command: CREATE ...
[2004-02-09 14:26:34.7] : Executing command: CREATE(PLAN)
[2004-02-09 14:27:08.469] : Retrieving the protocol specification ...
[2004-02-09 14:27:08.849] : Retrieving the schedule set ...
[2004-02-09 14:27:09.36] : Retrieving the schedule ...
[2004-02-09 14:27:09.891] : Checking if schedule [ID = 1] exists ...
[2004-02-09 14:27:10.271] : Schedule [ID = 1] exists.
[2004-02-09 14:27:10.301] : Retrieving the static rule set ...
[2004-02-09 14:27:10.582] : Retrieving the static rule ...
[2004-02-09 14:27:11.243] : Retrieving the rule action ...
[2004-02-09 14:27:11.743] : Retrieving the rule set ...
[2004-02-09 14:27:11.964] : Retrieving the rule ...
[2004-02-09 14:27:12.595] : Retrieving the rule event ...
[2004-02-09 14:27:13.216] : Retrieving the rule condition ...
[2004-02-09 14:27:13.606] : CONDITION: [ID=1][attribute=DSU][left_value=positive][type=STRING]
[2004-02-09 14:27:13.806] : Retrieving the rule action ...
[2004-02-09 14:27:14.017] : Retrieving the rule ...
[2004-02-09 14:27:14.377] : Retrieving the rule event ...
[2004-02-09 14:27:14.818] : Retrieving the rule action ...
[2004-02-09 14:27:15.038] : Retrieving the schedule ...
[2004-02-09 14:27:15.349] : Checking if schedule [ID = 2] exists ...
[2004-02-09 14:27:15.469] : Schedule [ID = 2] exists.
[2004-02-09 14:27:15.509] : Retrieving the static rule set ...
[2004-02-09 14:27:15.819] : Retrieving the static rule ...
[2004-02-09 14:27:16.01] : Retrieving the rule action ...
[2004-02-09 14:27:16.33] : Retrieving the rule set ...
[2004-02-09 14:27:16.52] : Retrieving the rule ...
[2004-02-09 14:27:16.781] : Retrieving the rule event ...
[2004-02-09 14:27:17.211] : Retrieving the rule condition ...
[2004-02-09 14:27:17.392] : CONDITION: [ID=3][attribute=UTI][left_value=negative][type=STRING]
[2004-02-09 14:27:17.552] : Retrieving the rule action ...
[2004-02-09 14:27:17.742] : Retrieving the rule ...
[2004-02-09 14:27:18.253] : Retrieving the rule event ...
[2004-02-09 14:27:18.553] : Retrieving the rule condition ...
[2004-02-09 14:27:18.713] : CONDITION: [ID=4][attribute=UTI][left_value=positive][type=STRING]
[2004-02-09 14:27:18.964] : Retrieving the rule action ...
[2004-02-09 14:27:19.745] : Retrieving the rule ...
[2004-02-09 14:27:20.306] : Retrieving the rule event ...
[2004-02-09 14:27:20.796] : Retrieving the rule condition ...
[2004-02-09 14:27:21.187] : CONDITION: [ID=5][attribute=24CRCL_PL][left_value=POSITIVE][type=STRING]
[2004-02-09 14:27:21.377] : Retrieving the rule action ...
[2004-02-09 14:27:21.548] : Retrieving the rule ...
[2004-02-09 14:27:22.148] : Retrieving the rule event ...
[2004-02-09 14:27:22.489] : Retrieving the rule condition ...
[2004-02-09 14:27:22.639] : CONDITION:
[ID=6][attribute=24CRCL_PL][left_value=NEGATIVE][type=STRING]
[2004-02-09 14:27:22.93] : Retrieving the rule action ...
[2004-02-09 14:27:23.14] : Retrieving the schedule ...
[2004-02-09 14:27:23.671] : Checking if schedule [ID = 3] exists ...
[2004-02-09 14:27:23.811] : Schedule [ID = 3] exists.
[2004-02-09 14:27:23.861] : Retrieving the static rule set ...
[2004-02-09 14:27:24.021] : Retrieving the static rule ...
[2004-02-09 14:27:24.241] : Retrieving the rule action ...
[2004-02-09 14:27:24.542] : Retrieving the rule set ...
[2004-02-09 14:27:24.972] : Retrieving the rule ...
[2004-02-09 14:27:25.323] : Retrieving the rule event ...
[2004-02-09 14:27:25.724] : Retrieving the rule condition ...
[2004-02-09 14:27:25.984] : CONDITION: [ID=7][attribute=ACR][left_value=20][type=DOUBLE]
[2004-02-09 14:27:26.184] : Retrieving the rule action ...
[2004-02-09 14:27:26.435] : Retrieving the rule ...
[2004-02-09 14:27:26.765] : Retrieving the rule event ...
[2004-02-09 14:27:27.266] : Retrieving the rule action ...
[2004-02-09 14:27:27.486] : Retrieving the rule ...
[2004-02-09 14:27:27.756] : Retrieving the rule event ...
[2004-02-09 14:27:28.508] : Retrieving the rule action ...
[2004-02-09 14:27:28.758] : Retrieving the rule ...
[2004-02-09 14:27:29.028] : Retrieving the rule event ...
[2004-02-09 14:27:29.429] : Retrieving the rule condition ...
[2004-02-09 14:27:29.809] : CONDITION: [ID=10][attribute=ACR][left_value=200][type=DOUBLE]
[2004-02-09 14:27:29.99] : Retrieving the rule action ...
[2004-02-09 14:27:30.17] : Retrieving the schedule ...
[2004-02-09 14:27:30.41] : Checking if schedule [ID = 4] exists ...
[2004-02-09 14:27:30.861] : Schedule [ID = 4] exists.
[2004-02-09 14:27:30.921] : Retrieving the static rule set ...
[2004-02-09 14:27:31.101] : Retrieving the static rule ...
[2004-02-09 14:27:31.342] : Retrieving the rule action ...
[2004-02-09 14:27:31.552] : Retrieving the static rule ...
[2004-02-09 14:27:31.832] : Retrieving the rule action ...
[2004-02-09 14:27:32.183] : Retrieving the static rule ...
[2004-02-09 14:27:32.393] : Retrieving the rule action ...
[2004-02-09 14:27:32.623] : Retrieving the static rule ...
[2004-02-09 14:27:33.084] : Retrieving the rule action ...
[2004-02-09 14:27:33.365] : Retrieving the rule set ...
[2004-02-09 14:27:33.495] : Retrieving the rule ...
[2004-02-09 14:27:33.755] : Retrieving the rule event ...
[2004-02-09 14:27:34.186] : Retrieving the rule action ...
[2004-02-09 14:27:34.376] : Retrieving the rule ...
[2004-02-09 14:27:34.807] : Retrieving the rule event ...
[2004-02-09 14:27:35.257] : Retrieving the rule action ...
[2004-02-09 14:27:35.428] : Retrieving the schedule ...
[2004-02-09 14:27:35.968] : Checking if schedule [ID = 5] exists ...
[2004-02-09 14:27:36.088] : Schedule [ID = 5] exists.
[2004-02-09 14:27:36.189] : Retrieving the static rule set ...
[2004-02-09 14:27:36.339] : Retrieving the static rule ...
[2004-02-09 14:27:36.529] : Retrieving the rule action ...
[2004-02-09 14:27:36.769] : Retrieving the rule set ...
[2004-02-09 14:27:37.06] : Retrieving the rule ...
[2004-02-09 14:27:37.36] : Retrieving the rule event ...
[2004-02-09 14:27:37.781] : Retrieving the rule action ...
[2004-02-09 14:27:38.091] : Retrieving the rule set ...
[2004-02-09 14:27:38.352] : Number of Schedules: 5
[2004-02-09 14:27:38.382] : Number of Protocol Rules: 0
[2004-02-09 14:27:38.943] : Creating plan for Patient Name: fn95857 sn25209 Patient
ID: 2]
[2004-02-09 14:27:53.243] : [start_time: 2004-02-09 14:27:38.973, end_time: 2005-02-
08 14:27:38.973, interal: 31536000]
[2004-02-09 14:28:03.248] : [Rule: AUS1; State changed to: READY]
[2004-02-09 14:28:08.775] : [start_time: 2004-02-09 14:28:03.508, end_time: 2004-02-
09 14:29:03.508, interal: 60]
[2004-02-09 14:28:16.406] : [Rule: OIS1; State changed to: READY]
[2004-02-09 14:28:20.392] : [start_time: 2004-02-09 14:28:16.547, end_time: 2004-02-
09 14:29:16.547, interal: 60]
[2004-02-09 14:28:23.867] : [Rule: MAS1; State changed to: READY]
[2004-02-09 14:28:31.849] : [start_time: 2004-02-09 14:28:24.027, end_time: 2004-02-
09 14:29:24.027, interal: 0]
[2004-02-09 14:28:33.912] : [Rule: CMA1; State changed to: READY]
[2004-02-09 14:28:37.417] : [start_time: 2004-02-09 14:28:34.042, end_time: 2004-02-
09 14:29:34.042, interal: 60]
[2004-02-09 14:28:39.44] : [Rule: CMA2; State changed to: READY]
[2004-02-09 14:28:42.674] : [start_time: 2004-02-09 14:29:39.63, end_time: 2004-02-09
14:29:39.63, interal: 60]
[2004-02-09 14:28:44.637] : [Rule: CMA3; State changed to: READY]
[2004-02-09 14:28:47.691] : [start_time: 2004-02-09 14:28:44.767, end_time: 2004-02-
09 14:28:44.767, interal: 2592000]
[2004-02-09 14:28:51.367] : [Rule: CMA4; State changed to: READY]
[2004-02-09 14:28:56.304] : [start_time: 2004-02-09 14:28:51.507, end_time: 2004-02-
09 14:29:51.507, interal: 60]
[2004-02-09 14:28:59.118] : [Rule: NPH1; State changed to: READY]
[2004-02-09 14:29:54.918] : Adding the plan's schedule to database.
[2004-02-09 14:29:56.721] : [Rule: P$21$1$AUS$AUS1; State changed to: READY]
[2004-02-09 14:29:59.815] : [Rule: P$21$1$AUS$AUS1; State changed to: ACTIVE]
[2004-02-09 14:29:59.835] : Adding the plan's schedule to database.
[2004-02-09 14:30:01.888] : [Rule: P$21$1$OIS$OIS1; State changed to: READY]
[2004-02-09 14:30:02.649] : [Rule: P$21$1$AUS$AUS1; State changed to: INACTIVE]
[2004-02-09 14:30:03.5] : [Rule: P$21$1$AUS$AUS1; State changed to: FINISHED]
[2004-02-09 14:30:06.345] : [Rule: P$21$1$OIS$OIS1; State changed to: ACTIVE]
[2004-02-09 14:30:06.365] : Adding the plan's schedule to database.
[2004-02-09 14:30:07.596] : [Rule: P$21$1$MAS$MAS1; State changed to: READY]
[2004-02-09 14:30:10.721] : [Rule: P$21$1$MAS$MAS1; State changed to: ACTIVE]
[2004-02-09 14:30:10.771] : Adding the plan's schedule to database.
[2004-02-09 14:30:11.802] : [Rule: P$21$1$CMA$CMA1; State changed to: READY]
[2004-02-09 14:30:15.127] : [Rule: P$21$1$CMA$CMA2; State changed to: READY]
[2004-02-09 14:30:18.202] : [Rule: P$21$1$CMA$CMA3; State changed to: READY]
[2004-02-09 14:30:22.538] : [Rule: P$21$1$CMA$CMA4; State changed to: READY]
[2004-02-09 14:30:45.26] : Rule listener active ...
[2004-02-09 14:30:45.361] : Rule listener waiting ...
[2004-02-09 14:30:45.391] : Rule listener receiving data ...
```

G. TOPS Session for Executing the MAP Patient Plan

```
[2004-07-19 12:23:53.565] : TOPS System Execution Log: Session Starting at 2004-07-19 12:23:52.243
[2004-07-19 12:23:53.675] : ------------------------------------------------------------------------
[2004-07-19 12:24:19.683] : Getting confirmation to create the TOPS database objects.
[2004-07-19 12:24:22.927] : Listener active ...
[2004-07-19 12:24:24.7] : Waiting ...
[2004-07-19 12:27:26.922] : Analysing command: create ...
[2004-07-19 12:27:26.932] : Executing command: create(plan)
[2004-07-19 12:28:49.731] : Retrieving the protocol spec ...
[2004-07-19 12:29:21.777] : Creating plan for patient [ Alex Ferguson, ID: 81 ]
[2004-07-19 12:29:25.082] : [start_time: 2004-07-19 12:29:21.887, end_time: 2004-07-19 12:30:21.887, interal: 60]
[2004-07-19 12:29:33.234] : [Rule: AUS1; State changed to: READY]
[2004-07-19 12:30:42.163] : Adding the plan's schedule to database.
[2004-07-19 12:30:43.435] : Adding the plan's schedule to database.
[2004-07-19 12:30:44.316] : Adding the plan's schedule to database.
[2004-07-19 12:30:45.067] : Adding the plan's schedule to database.
[2004-07-19 12:30:45.958] : Adding the plan's schedule to database.
[2004-07-19 12:30:46.94] : [Rule: PL$81$1$main$AUS1; State changed to: READY]
[2004-07-19 12:30:49.644] : [Rule: PL$81$1$main$AUS1; State changed to: ACTIVE]
[2004-07-19 12:31:12.877] : Activated ...
[2004-07-19 12:31:12.937] : Waiting ...
[2004-07-19 12:31:12.977] : Receiving data ...
[2004-07-19 12:31:13.157] : Received |-->[PL$81$1$main$AUS1%TEST_ORDER*81|DSU,|*%]
[2004-07-19 12:31:13.338] : [PL$81$1$main$AUS1] executing ...TEST_ORDER (81|DSU,|)
[2004-07-19 12:31:14.569] : [Rule: PL$81$1$main$AUS1; State changed to: FINISHED]
[2004-07-19 12:31:25.705] : Activated ...
[2004-07-19 12:31:25.705] : Waiting ...
[2004-07-19 12:31:25.755] : Receiving data ...
[2004-07-19 12:31:25.816] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|81|8|2004-07-19 12:31:23.0|*%]
[2004-07-19 12:31:25.996] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|81|8|2004-07-19 12:31:23.0|)
[2004-07-19 12:31:26.136] : [LabSimulator Started ...]
[2004-07-19 12:31:27.428] : Activated ...
[2004-07-19 12:31:27.498] : Waiting ...
[2004-07-19 12:31:27.598] : Receiving data ...
[2004-07-19 12:31:27.638] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|81|8|2004-07-19 12:31:23.0|*%]
[2004-07-19 12:31:28.68] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|81|8|2004-07-19 12:31:23.0|)
[2004-07-19 12:31:28.75] : [LabSimulator Started ...]
[2004-07-19 12:31:32.025] : [Connection closed]
[2004-07-19 12:31:32.055] : [Connection closed]
[2004-07-19 12:31:33.266] : Activated ...
[2004-07-19 12:31:33.276] : Waiting ...
[2004-07-19 12:31:33.346] : Receiving data ...
[2004-07-19 12:31:33.396] : Received |-->[PL$81$1$OIS1%TEST_ORDER*81|UTI,|*%]
[2004-07-19 12:31:33.487] : Activated ...
[2004-07-19 12:31:33.487] : Waiting ...
[2004-07-19 12:31:33.617] : Activated ...
[2004-07-19 12:31:33.647] : Waiting ...
[2004-07-19 12:31:33.707] : Receiving data ...
[2004-07-19 12:31:33.747] : Received |-->[PL$61$1$OIS1%TEST_ORDER*61|UTI,|*%]
[2004-07-19 12:31:33.827] : Activated ...
[2004-07-19 12:31:33.827] : Waiting ...
[2004-07-19 12:31:33.907] : Receiving data ...
[2004-07-19 12:31:33.927] : Received |-->[PL$81$1$AUS2%PATIENT_STATE*81,other_infections_screening,*%]
[2004-07-19 12:31:33.967] : Receiving data ...
[2004-07-19 12:31:34.007] : Received |-->[PL$81$1$OIS1%TEST_ORDER*81|UTI,|*%]
[2004-07-19 12:31:34.128] : Activated ...
[2004-07-19 12:31:34.148] : Waiting ...
[2004-07-19 12:31:34.268] : Activated ...
[2004-07-19 12:31:34.308] : Waiting ...
[2004-07-19 12:31:34.428] : Receiving data ...
[2004-07-19 12:31:34.468] : Received |-->[PL$61$1$OIS1%TEST_ORDER*61|UTI,|*%]
[2004-07-19 12:31:34.518] : Receiving data ...
[2004-07-19 12:31:34.598] : Activated ...
[2004-07-19 12:31:34.678] : Received |-->[PL$81$1$AUS2%PATIENT_STATE*81,other_infections_screening,*%]
[2004-07-19 12:31:34.768] : Waiting ...
[2004-07-19 12:31:34.869] : Activated ...
[2004-07-19 12:31:34.989] : Receiving data ...
[2004-07-19 12:31:35.029] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*81|10|81|1.0|2004-07-19 12:31:26.0|*%]
[2004-07-19 12:31:35.129] : Waiting ...
[2004-07-19 12:31:35.189] : Receiving data ...
[2004-07-19 12:31:35.239] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*82|10|81|1.0|2004-07-19 12:31:28.0|*%]
[2004-07-19 12:31:35.379] : [PL$81$1$OIS1] executing ...TEST_ORDER (81|UTI,|)
[2004-07-19 12:31:35.55] : [PL$61$1$OIS1] executing ...TEST_ORDER (61|UTI,|)
[2004-07-19 12:31:35.72] : [PL$81$1$AUS2] executing ...PATIENT_STATE (81,other_infections_screening,)
[2004-07-19 12:31:35.89] : [PL$81$1$OIS1] executing ...TEST_ORDER (81|UTI,|)
[2004-07-19 12:31:36.16] : [PL$61$1$OIS1] executing ...TEST_ORDER (61|UTI,|)
[2004-07-19 12:31:36.331] : [PL$81$1$AUS2] executing ...PATIENT_STATE (81,other_infections_screening,)
[2004-07-19 12:31:36.501] : [NEW_RESULT_ARRIVAL] is executing RESULT (81|10|81|1.0|2004-07-19 12:31:26.0|)
[2004-07-19 12:31:36.801] : [NEW_RESULT_ARRIVAL] is executing RESULT (82|10|81|1.0|2004-07-19 12:31:28.0|)
[2004-07-19 12:31:41.618] : [Connection closed]
[2004-07-19 12:31:41.819] : [Connection closed]
[2004-07-19 12:31:43.03] : [Connection closed]
[2004-07-19 12:32:26.523] : Activated ...
[2004-07-19 12:32:26.523] : Waiting ...
[2004-07-19 12:32:26.633] : Receiving data ...
[2004-07-19 12:32:26.683] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|82|9|2004-07-19 12:32:22.0|*%]
[2004-07-19 12:32:26.833] : Activated ...
[2004-07-19 12:32:26.833] : Waiting ...
[2004-07-19 12:32:26.984] : Receiving data ...
[2004-07-19 12:32:27.024] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|82|9|2004-07-19 12:32:22.0|*%]
[2004-07-19 12:32:27.654] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|82|9|2004-07-19 12:32:22.0|)
[2004-07-19 12:32:27.705] : [LabSimulator Started ...]
[2004-07-19 12:32:28.095] : [Connection closed]
[2004-07-19 12:32:28.215] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|82|9|2004-07-19 12:32:22.0|)
[2004-07-19 12:32:28.265] : [LabSimulator Started ...]
[2004-07-19 12:32:28.476] : [Connection closed]
[2004-07-19 12:32:29.097] : Activated ...
[2004-07-19 12:32:29.107] : Waiting ...
[2004-07-19 12:32:29.197] : Receiving data ...
[2004-07-19 12:32:29.287] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*61|83|9|2004-07-19 12:32:23.0|*%]
[2004-07-19 12:32:29.407] : Activated ...
[2004-07-19 12:32:29.447] : Waiting ...
[2004-07-19 12:32:29.507] : Receiving data ...
[2004-07-19 12:32:29.547] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*61|83|9|2004-07-19 12:32:23.0|*%]
[2004-07-19 12:32:30.158] : Activated ...
[2004-07-19 12:32:30.168] : Waiting ...
[2004-07-19 12:32:30.248] : Receiving data ...
[2004-07-19 12:32:30.368] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*61|84|9|2004-07-19 12:32:23.0|*%]
[2004-07-19 12:32:30.428] : Activated ...
[2004-07-19 12:32:30.439] : Waiting ...
[2004-07-19 12:32:30.549] : Receiving data ...
[2004-07-19 12:32:30.589] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*61|84|9|2004-07-19 12:32:23.0|*%]
[2004-07-19 12:32:31.059] : Activated ...
[2004-07-19 12:32:31.099] : Waiting ...
[2004-07-19 12:32:31.16] : Receiving data ...
[2004-07-19 12:32:31.21] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|85|9|2004-07-19 12:32:23.0|*%]
[2004-07-19 12:32:31.29] : Activated ...
[2004-07-19 12:32:31.32] : Receiving data ...
[2004-07-19 12:32:31.32] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|85|9|2004-07-19 12:32:23.0|*%]
[2004-07-19 12:32:31.43] : Waiting ...
[2004-07-19 12:32:32.121] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (61|83|9|2004-07-19 12:32:23.0|)
[2004-07-19 12:32:32.171] : [LabSimulator Started ...]
[2004-07-19 12:32:32.341] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (61|83|9|2004-07-19 12:32:23.0|)
[2004-07-19 12:32:32.401] : [LabSimulator Started ...]
[2004-07-19 12:32:32.461] : [Connection closed]
[2004-07-19 12:32:32.642] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (61|84|9|2004-07-19 12:32:23.0|)
[2004-07-19 12:32:32.682] : [Connection closed]
[2004-07-19 12:32:32.782] : [LabSimulator Started ...]
[2004-07-19 12:32:32.952] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (61|84|9|2004-07-19 12:32:23.0|)
[2004-07-19 12:32:33.002] : [LabSimulator Started ...]
[2004-07-19 12:32:33.062] : [Connection closed]
[2004-07-19 12:32:33.303] : [Connection closed]
[2004-07-19 12:32:33.573] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|85|9|2004-07-19 12:32:23.0|)
[2004-07-19 12:32:33.633] : [LabSimulator Started ...]
[2004-07-19 12:32:33.863] : [Connection closed]
[2004-07-19 12:32:33.984] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|85|9|2004-07-19 12:32:23.0|)
[2004-07-19 12:32:34.044] : [LabSimulator Started ...]
[2004-07-19 12:32:34.264] : [Connection closed]
[2004-07-19 12:32:39.181] : [Connection closed]
[2004-07-19 12:32:39.922] : Activated ...
[2004-07-19 12:32:39.922] : Waiting ...
[2004-07-19 12:32:40.012] : Receiving data ...
[2004-07-19 12:32:40.052] : Received |-->[PL$81$1$OIS3%PATIENT_STATE*81,annual_urine_screening,*%]
[2004-07-19 12:32:40.132] : Activated ...
[2004-07-19 12:32:40.132] : Waiting ...
[2004-07-19 12:32:40.223] : Receiving data ...
[2004-07-19 12:32:40.263] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*83|11|82|1.0|2004-07-19 12:32:27.0|*%]
[2004-07-19 12:32:40.573] : [Connection closed]
[2004-07-19 12:32:41.324] : Activated ...
[2004-07-19 12:32:41.364] : Waiting ...
[2004-07-19 12:32:41.765] : Activated ...
[2004-07-19 12:32:41.775] : Waiting ...
[2004-07-19 12:32:41.775] : Receiving data ...
[2004-07-19 12:32:41.775] : Received |-->[PL$81$1$OIS2%TEST_ORDER*81|24CRCL_PL,|*%]
[2004-07-19 12:32:41.955] : Receiving data ...
[2004-07-19 12:32:42.005] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*84|11|82|0.0|2004-07-19 12:32:28.0|*%]
[2004-07-19 12:32:43.277] : [PL$81$1$OIS3] executing ...PATIENT_STATE (81,annual_urine_screening,)
[2004-07-19 12:32:43.437] : [NEW_RESULT_ARRIVAL] is executing RESULT (83|11|82|1.0|2004-07-19 12:32:27.0|)
[2004-07-19 12:32:43.898] : [NEW_RESULT_ARRIVAL] is executing RESULT (84|11|82|0.0|2004-07-19 12:32:28.0|)
[2004-07-19 12:32:44.779] : Activated ...
[2004-07-19 12:32:44.849] : Waiting ...
[2004-07-19 12:32:44.939] : Receiving data ...
[2004-07-19 12:32:44.979] : Received |-->[PL$61$1$OIS2%TEST_ORDER*61|24CRCL_PL,|*%]
[2004-07-19 12:32:45.039] : Activated ...
[2004-07-19 12:32:45.039] : Waiting ...
[2004-07-19 12:32:45.18] : Receiving data ...
[2004-07-19 12:32:45.18] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*85|11|83|0.0|2004-07-19 12:32:32.0|*%]
[2004-07-19 12:32:45.58] : Activated ...
[2004-07-19 12:32:45.58] : Waiting ...
[2004-07-19 12:32:45.65] : Receiving data ...
[2004-07-19 12:32:45.71] : Received |-->[PL$61$1$OIS3%PATIENT_STATE*61,annual_urine_screening,*%]
[2004-07-19 12:32:45.791] : Activated ...
[2004-07-19 12:32:45.841] : Waiting ...
```

[2004-07-19 12:32:45.881] : Receiving data ...
[2004-07-19 12:32:45.891] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*86|11|83|1.0|2004-07-19 12:32:32.0|*%]
[2004-07-19 12:32:46.792] : Activated ...
[2004-07-19 12:32:46.792] : Waiting ...
[2004-07-19 12:32:46.882] : Receiving data ...
[2004-07-19 12:32:46.932] : Received |-->[PL$61$1$OIS3%PATIENT_STATE*61,annual_urine_screening,*%]
[2004-07-19 12:32:47.022] : Activated ...
[2004-07-19 12:32:47.032] : Waiting ...
[2004-07-19 12:32:47.143] : Receiving data ...
[2004-07-19 12:32:47.183] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*87|11|84|1.0|2004-07-19 12:32:32.0|*%]
[2004-07-19 12:32:47.473] : [PL$81$1$OIS2] executing ...TEST_ORDER (81|24CRCL_PL,|)
[2004-07-19 12:32:47.673] : Activated ...
[2004-07-19 12:32:47.713] : Waiting ...
[2004-07-19 12:32:47.763] : Receiving data ...
[2004-07-19 12:32:47.803] : Received |-->[PL$81$1$OIS3%PATIENT_STATE*81,annual_urine_screening,*%]
[2004-07-19 12:32:47.894] : Activated ...
[2004-07-19 12:32:47.894] : Waiting ...
[2004-07-19 12:32:48.014] : Receiving data ...
[2004-07-19 12:32:48.064] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*88|11|85|1.0|2004-07-19 12:32:33.0|*%]
[2004-07-19 12:32:49.155] : Activated ...
[2004-07-19 12:32:49.155] : Waiting ...
[2004-07-19 12:32:49.246] : Receiving data ...
[2004-07-19 12:32:49.286] : Received |-->[PL$61$1$OIS3%PATIENT_STATE*61,annual_urine_screening,*%]
[2004-07-19 12:32:49.416] : Activated ...
[2004-07-19 12:32:49.416] : Waiting ...
[2004-07-19 12:32:49.516] : Receiving data ...
[2004-07-19 12:32:49.546] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*89|11|84|1.0|2004-07-19 12:32:32.0|*%]
[2004-07-19 12:32:49.816] : Activated ...
[2004-07-19 12:32:49.856] : Waiting ...
[2004-07-19 12:32:50.017] : Receiving data ...
[2004-07-19 12:32:50.077] : Received |-->[PL$81$1$OIS2%TEST_ORDER*81|24CRCL_PL,|*%]
[2004-07-19 12:32:50.187] : Activated ...
[2004-07-19 12:32:50.187] : Waiting ...
[2004-07-19 12:32:50.267] : Receiving data ...
[2004-07-19 12:32:50.317] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*90|11|85|0.0|2004-07-19 12:32:34.0|*%]
[2004-07-19 12:32:51.168] : [PL$61$1$OIS2] executing ...TEST_ORDER (61|24CRCL_PL,|)
[2004-07-19 12:32:51.349] : [PL$61$1$OIS3] executing ...PATIENT_STATE (61,annual_urine_screening,)
[2004-07-19 12:32:51.639] : [NEW_RESULT_ARRIVAL] is executing RESULT (85|11|83|0.0|2004-07-19 12:32:32.0|)
[2004-07-19 12:32:51.809] : [PL$61$1$OIS3] executing ...PATIENT_STATE (61,annual_urine_screening,)
[2004-07-19 12:32:51.979] : [NEW_RESULT_ARRIVAL] is executing RESULT (87|11|84|1.0|2004-07-19 12:32:32.0|)
[2004-07-19 12:32:52.16] : [NEW_RESULT_ARRIVAL] is executing RESULT (86|11|83|1.0|2004-07-19 12:32:32.0|)
[2004-07-19 12:32:52.34] : [PL$81$1$OIS3] executing ...PATIENT_STATE (81,annual_urine_screening,)
[2004-07-19 12:32:52.67] : [NEW_RESULT_ARRIVAL] is executing RESULT (88|11|85|1.0|2004-07-19 12:32:33.0|)
[2004-07-19 12:32:52.981] : [PL$61$1$OIS3] executing ...PATIENT_STATE (61,annual_urine_screening,)
[2004-07-19 12:32:53.161] : [NEW_RESULT_ARRIVAL] is executing RESULT (89|11|84|1.0|2004-07-19 12:32:32.0|)
[2004-07-19 12:32:53.351] : [PL$81$1$OIS2] executing ...TEST_ORDER (81|24CRCL_PL,|)
[2004-07-19 12:32:53.682] : [NEW_RESULT_ARRIVAL] is executing RESULT (90|11|85|0.0|2004-07-19 12:32:34.0|)
[2004-07-19 12:33:00.892] : [Connection closed]
[2004-07-19 12:33:01.093] : [Connection closed]
[2004-07-19 12:33:04.107] : [Connection closed]
[2004-07-19 12:33:04.497] : [Connection closed]
[2004-07-19 12:33:04.698] : [Connection closed]
[2004-07-19 12:33:05.128] : [Connection closed]
[2004-07-19 12:33:05.509] : [Connection closed]
[2004-07-19 12:33:05.709] : [Connection closed]
[2004-07-19 12:33:05.98] : [Connection closed]
[2004-07-19 12:33:07.221] : [Connection closed]
[2004-07-19 12:33:08.243] : [Connection closed]
[2004-07-19 12:33:08.363] : [Connection closed]
[2004-07-19 12:34:04.704] : Activated ...
[2004-07-19 12:34:04.724] : Waiting ...
[2004-07-19 12:34:04.824] : Receiving data ...
[2004-07-19 12:34:04.894] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|86|10|2004-07-19 12:33:59.0|*%]
[2004-07-19 12:34:04.974] : Activated ...
[2004-07-19 12:34:04.984] : Waiting ...
[2004-07-19 12:34:05.065] : Receiving data ...
[2004-07-19 12:34:05.175] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|86|10|2004-07-19 12:33:59.0|*%]
[2004-07-19 12:34:05.876] : Activated ...
[2004-07-19 12:34:05.886] : Waiting ...
[2004-07-19 12:34:06.016] : Receiving data ...
[2004-07-19 12:34:06.076] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*61|87|10|2004-07-19 12:33:59.0|*%]
[2004-07-19 12:34:06.166] : Activated ...
[2004-07-19 12:34:06.186] : Waiting ...
[2004-07-19 12:34:06.316] : Receiving data ...
[2004-07-19 12:34:06.356] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*61|87|10|2004-07-19 12:33:59.0|*%]

[2004-07-19 12:34:07.188] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|86|10|2004-07-19 12:33:59.0|)
[2004-07-19 12:34:07.248] : [LabSimulator Started ...]
[2004-07-19 12:34:07.538] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|86|10|2004-07-19 12:33:59.0|)
[2004-07-19 12:34:07.558] : [Connection closed]
[2004-07-19 12:34:07.658] : [LabSimulator Started ...]
[2004-07-19 12:34:07.919] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (61|87|10|2004-07-19 12:33:59.0|)
[2004-07-19 12:34:07.959] : [Connection closed]
[2004-07-19 12:34:08.069] : [LabSimulator Started ...]
[2004-07-19 12:34:08.229] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (61|87|10|2004-07-19 12:33:59.0|)
[2004-07-19 12:34:08.289] : [LabSimulator Started ...]
[2004-07-19 12:34:08.429] : [Connection closed]
[2004-07-19 12:34:08.6] : [Connection closed]
[2004-07-19 12:34:09.551] : Activated ...
[2004-07-19 12:34:09.561] : Waiting ...
[2004-07-19 12:34:09.651] : Receiving data ...
[2004-07-19 12:34:09.701] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|88|10|2004-07-19 12:34:00.0|*%]
[2004-07-19 12:34:09.811] : Activated ...
[2004-07-19 12:34:09.821] : Waiting ...
[2004-07-19 12:34:09.972] : Receiving data ...
[2004-07-19 12:34:10.022] : Received |-->[ORDER_EVENT_TRIGGER%TEST_ORDER_EVENT*81|88|10|2004-07-19 12:34:00.0|*%]
[2004-07-19 12:34:11.684] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|88|10|2004-07-19 12:34:00.0|)
[2004-07-19 12:34:11.744] : [LabSimulator Started ...]
[2004-07-19 12:34:11.985] : [Connection closed]
[2004-07-19 12:34:12.155] : [ORDER_EVENT_TRIGGER] is executing TEST_ORDER_EVENT (81|88|10|2004-07-19 12:34:00.0|)
[2004-07-19 12:34:12.225] : [LabSimulator Started ...]
[2004-07-19 12:34:12.375] : [Connection closed]
[2004-07-19 12:34:13.497] : [Connection closed]
[2004-07-19 12:34:14.328] : [Connection closed]
[2004-07-19 12:34:15.179] : [Connection closed]
[2004-07-19 12:34:16.351] : [Connection closed]
[2004-07-19 12:34:16.741] : [Connection closed]
[2004-07-19 12:34:18.334] : Activated ...
[2004-07-19 12:34:18.344] : Waiting ...
[2004-07-19 12:34:18.474] : Receiving data ...
[2004-07-19 12:34:18.524] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*91|8|86|23.12858|2004-07-19 12:34:07.0|*%]
[2004-07-19 12:34:18.714] : Activated ...
[2004-07-19 12:34:18.724] : Waiting ...
[2004-07-19 12:34:18.794] : Receiving data ...
[2004-07-19 12:34:18.864] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*92|8|86|-0.8134979999999999|2004-07-19 12:34:07.0|*%]
[2004-07-19 12:34:19.515] : Activated ...
[2004-07-19 12:34:19.535] : Waiting ...
[2004-07-19 12:34:19.615] : Receiving data ...
[2004-07-19 12:34:19.686] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*93|8|87|18.850575|2004-07-19 12:34:08.0|*%]
[2004-07-19 12:34:20.427] : Activated ...
[2004-07-19 12:34:20.447] : Waiting ...
[2004-07-19 12:34:20.557] : Receiving data ...
[2004-07-19 12:34:20.617] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*94|8|87|12.073729|2004-07-19 12:34:08.0|*%]
[2004-07-19 12:34:21.408] : [NEW_RESULT_ARRIVAL] is executing RESULT (91|8|86|23.12858|2004-07-19 12:34:07.0|)
[2004-07-19 12:34:21.568] : [NEW_RESULT_ARRIVAL] is executing RESULT (92|8|86|-0.8134979999999999|2004-07-19 12:34:07.0|)
[2004-07-19 12:34:21.749] : [NEW_RESULT_ARRIVAL] is executing RESULT (93|8|87|18.850575|2004-07-19 12:34:08.0|)
[2004-07-19 12:34:21.909] : [NEW_RESULT_ARRIVAL] is executing RESULT (94|8|87|12.073729|2004-07-19 12:34:08.0|)
[2004-07-19 12:34:22.72] : Activated ...
[2004-07-19 12:34:22.74] : Waiting ...
[2004-07-19 12:34:22.79] : Receiving data ...
[2004-07-19 12:34:22.92] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*95|8|88|10.242782|2004-07-19 12:34:11.0|*%]
[2004-07-19 12:34:23.681] : Activated ...
[2004-07-19 12:34:23.701] : Waiting ...
[2004-07-19 12:34:23.751] : Receiving data ...
[2004-07-19 12:34:23.872] : Received |-->[NEW_RESULT_ARRIVAL%RESULT*96|8|88|0.7299559999999999|2004-07-19 12:34:12.0|*%]
[2004-07-19 12:34:24.803] : [NEW_RESULT_ARRIVAL] is executing RESULT (95|8|88|10.242782|2004-07-19 12:34:11.0|)

H. The BNF Syntax of TOPSQL

H.1. The BNF Syntax of Manipulation Operations in TOPSQL

<TOPSQL> ::= <CREATEcmd> | <ADDcmd> | <DELETEcmd> | <EDITcmd> | <ACTIVATEcmd> |
<DEACTIVATEcmd> | <STOPcmd> | <DISPLAYcmd> | <LISTcmd> | <TOPSQL_query>
<CREATEcmd> ::= CREATE [OR REPLACE] <tops-object-type> [FOR <patientDef>] AS
"("<PLANdef>")"
<INSERTcmd> ::= INSERT <tops-object-type> <tops-object-name> <tops-object-type> "("
<PLANdef>")" | <tops-object-name>
<PLANdef> ::= <eventDef> | <actionDef> | <conditionDef> | <ruleDef> | <scheduleDef> |
<protocolDef> | <categoryDef> | <patientDef>
<eventDef> ::= <*PLAN* ***event***>
<actionDef> ::= <*PLAN* ***action***>
<conditionDef> ::= <*PLAN* ***condition***>
<ruleDef> ::= <*PLAN* ***rule***>
<scheduleDef> ::= <*PLAN* ***schedule***>
<protocolDef> ::= <*PLAN* ***protocol***>
<categoryDef> ::= <*TOPS* ***category***>
<patientDef> ::= <patient-id> | <patient-specification>
<patient-id> ::= <name> | <mrn>
<patient-specification> ::= <*TOPS specification* ***patient***>
<DELETEcmd> ::= DELETE <tops-object-type> <tops-object-name>
<EDITcmd> ::= EDIT <tops-object-type> <tops-object-name>
<ACTIVATEcmd> ::= ACTIVATE <tops-object-type> <tops-object-name>
<DEACTIVATEcmd> ::= DEACTIVATE <tops-object-type> <tops-object-name>
<STOPcmd> ::= STOP <tops-object-type> <tops-object-name>
<DISPLAYcmd> ::= DISPLAY <tops-object-type> <tops-object-name>
<LISTcmd> ::= LIST <tops-object-type>
(*see Appendix H.2 for the expansion of* <TOPSQL_query>*)*

H.2. The BNF Syntax of Queries in TOPSQL

<TOPSQL-query> ::= SELECT <select-item> [SPEC] {FOR | FROM | IN} <reference-item> WHERE
[TARGET: <condition-spec>; SOURCE:] <condition-spec>
<select-item> ::= {<target-obj-type> | <domain-dependent-obj-type>}
<reference-item> :: = {<source-ref-obj-type> | <domain-dependent-ref-obj-type>}
<target-obj-type> ::= EVENT | CONDITION | ACTION | RULE | SCHEDULE | PLAN | PROTOCOL|
CATEGORY
<domain-depenedent-ref-obj-type> ::= TEST| RESULT | TEST-ORDER | PATIENT | …
<source-ref-obj-type> ::= RULE | SCHEDULE | PLAN | PROTOCOL| | CATEGORY
<condition-spec> ::=<condition>|<time-interval>
<condition> ::= <SQL-condition>
<time-interval>::=<timestamp>,<timestamp>
<timestamp>::=<year>-<month>-dayOfMonth><blankspace><hour>:<minute>:<second>

I. TOPSQL Queries on the MAP in TOPS

I.1. Existing categories

Query	*List all categories in TOPS*
TOPSQL Statement	LIST CATEGORY

```
TOPS:\>list category
Executing command: list(category)
command code: 4

LIST of CATEGORIES
---------
MA#1
cat1#21
cat2#22
------
end LIST.

TOPS:\>
```

The listing above shows that there are currently three categories defined in TOPS: the MA (microalbuminuria) category whose ID is 1, which is appended to the category name after the hash (#) character, the *cat1* and *cat2* categories, which are sample categories created for testing. The MA category was created in the case study for the microalbuminuria protocol.

I.2. Displaying the MAP Specification in TOPSQL

Description	Displays the complete protocol for the category, MA (the microalbuminuria category)
TOPSSQL statement	**DISPLAY** protocol MA

```
TOPS starting ...
Initialising ...
Creating log files ...
Setting TOPS system attributes ...
host name or IP address --->ibmt20
client name or IP address --->ibmt20
database --->tops
database url: jdbc:oracle:thin:@ibmt20:1521:TOPS
Initialisation complete.

TOPS Command Line Facility, version 1.0.
Copyright(C) 2000-2003, K. Dube
Computer Science Department, School of Computing, DIT, Ireland.
Started at: Fri Jul 16 12:09:06 BST 2004
TOPS:\>display protocol MA
Executing command: display(protocol,MA)
command code: 9
Retrieving the protocol spec ...
Protocol spec retrieval complete.
schedules: 4
protocol dynamic rules: 9
protocol static rules: 1

PROTOCOL_NAME: MAP;
DESCRIPTION: This is a protocol for the diagnosis and management of microalbumin
uria in diabetes patients;
DATE_CREATED: 2004-07-15 21:20:22.0;
CREATOR_ID: 1;
CATEGORY_ID: 1;
BEGIN SCHEDULE_SET
BEGIN SCHEDULE;
SCHEDULE_NAME: AUS;
DESCRIPTION: This is a microalbuminuria protocol schedule called AUS for Annual
dipstick Urine Screening;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: AUS2;
DESCRIPTION: no description;
ON: result_arrival('DSU');
IF: DSU = 1.0;
DO: PATIENT_STATE('other_infections_screening');
RULE_NAME: AUS3;
DESCRIPTION: no description;
ON: result_arrival('DSU');
IF: DSU = 0.0;
DO: PATIENT_STATE('microalbuminuria_screening');
END SCHEDULE_RULE_SET;
END SCHEDULE;
BEGIN SCHEDULE;
SCHEDULE_NAME: OIS;
DESCRIPTION: This is a microalbuminuria protocol schedule called OIS for SCREENI
NG OTHER INFECTIONS in the diagnosis of microalbuminuria and proteinuria;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: OIS2;
DESCRIPTION: no description;
ON: result_arrival('UTI');
IF: UTI = 0.0;
DO: ORDER_TEST('24CRCL_PL');
RULE_NAME: OIS3;
DESCRIPTION: no description;
ON: result_arrival('UTI');
IF: UTI = 1.0;
DO: PATIENT_STATE('annual_urine_screening');
RULE_NAME: OIS4;
DESCRIPTION: no description;
ON: result_arrival('24CRCL_PL');
IF: 24CRCL_PL = 1.0;
DO: PATIENT_STATE('nephrology_referral');
RULE_NAME: 0IS5;
DESCRIPTION: no description;
ON: result_arrival('24CRCL_PL');
IF: 24CRCL_PL = 0.0;
DO: PATIENT_STATE('annual_urine_screening');
END SCHEDULE_RULE_SET;
END SCHEDULE;
BEGIN SCHEDULE;
SCHEDULE_NAME: MAS;
DESCRIPTION: This is a microalbuminuria protocol schedule called MAS for the scr
eening of microalbuminuria;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: MAS2;
DESCRIPTION: no description;
ON: result_arrival('ACR');
IF: ACR > 20.0;
DO: ADD_RULE('MAS2','ADD_RULE*MAS2/STATIC/MAS2alnulltime_rule_addedl0l155520000
00l7776000000lORDER_TEST;"ACR";lrule orders ACR test during the next 6 month p
eriod/*');
RULE_NAME: MAS3;
DESCRIPTION: no description;
ON: result_arrival('ACR');
IF: ACR > 20.0;
DO: PATIENT_STATE('annual_urine_screening');
RULE_NAME: MAS4;
DESCRIPTION: no description;
ON: result_arrival('ACR');
DO: CHECK_2OF3_ACR(null);
RULE_NAME: MAS5;
DESCRIPTION: no description;
ON: RESULT_ARRIVAL('ACR');
IF: ACR > 200.0;
DO: PATIENT_STATE('nephrology_referral');
END SCHEDULE_RULE_SET;
END SCHEDULE;
BEGIN SCHEDULE;
SCHEDULE_NAME: CMA;
DESCRIPTION: This is a microalbuminuria protocol schedule named CMA for confirme
d microalbuminuria û handles treatment and control of microalbuminuria;
BEGIN SCHEDULE_RULE_SET;
RULE_NAME: CMA5;
DESCRIPTION: no description;
ON: result_arrival('ACR');
IF: ACR < 20.0;
DO: PATIENT_STATE('annual_urine_screening');
RULE_NAME: CMA6;
DESCRIPTION: no description;
ON: result_arrival('ACR');
IF: ACR > 200.0;
DO: PATIENT_STATE('nephrology_referral');
END SCHEDULE_RULE_SET;
END SCHEDULE;
END SCHEDULE_SET
BEGIN PROTOCOL_RULE_SETBEGIN STATIC_RULE_SET;
RULE_NAME: AUS1;
DESCRIPTION: no description;
ZERO_TIME_POINT: annual_screening_start_date;
START_DATE: 0;
END_TIME: 60000;
INTERVAL: 60000;
DO: ORDER_TEST('DSU');
END STATIC_RULE_SET;
BEGIN DYNAMIC_RULE_SET;
RULE_NAME: OIS1;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'other_infections_screening';
DO: ORDER_TEST('UTI');
RULE_NAME: MAS1a;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'microalbuminuria_screening';
DO: ORDER_TEST('ACR');
RULE_NAME: MAS1b;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'microalbuminuria_screening';
DO: ORDER_TEST('SCR');
RULE_NAME: CMA1;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'confirmed_microalbuminuria';
DO: SUGGEST('optimisation_of_glycaemic_control');
RULE_NAME: CMA2;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'confirmed_microalbuminuria';
DO: ORDER_TEST('BP');
RULE_NAME: CMA3;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'confirmed_microalbuminuria';
DO: PRESCRIBE_MEDICATION('ACE_inhibitor');
RULE_NAME: CMA4a;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'confirmed_microalbuminuria';
DO: ORDER_TEST('ACR');
RULE_NAME: CMA4b;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'confirmed_microalbuminuria';
DO: ORDER_TEST('SCR');
RULE_NAME: NPH1;
DESCRIPTION: no description;
ON: state_change();
IF: STATE_NAME = 'confirmed_microalbuminuria';
DO: SEND_REFERRAL_NOTE('Nephrologist>Please examine this diabetic patient for
pr
oteinuria.');
END DYNAMIC_RULE_SET;
END PROTOCOL_RULE_SET
END PROTOCOL.

TOPS:\>
```

I.3. Existing patients in TOPS

Query	*List all patients in TOPS*
TOPSQL Statement	LIST PATIENT(S)

```
TOPS:\>list patients
Executing command: list(patients)
command code: 4

PATIENTS in TOPS
------
<<surname, firstname, patient_id, category_id>>
Dube, Kuda, 1, 1
Dube, Ano, 2, 1
Nhakwi, Sando, 21, 1
Moyo, Jabu, 41, 1
Banks, Frank, 42, 1
Doe, Mary, 61, 1
Ferguson, Alfred, 81, 1

-----
END PATIENT LIST

TOPS:\>
```

The above query lists all patients who are currently in TOPS disregarding the status of their plans. The query results also include the patient's TOPS ID and the ID of the category to which the patient currently belongs.

I.4. Existing plans in TOPS

Query	*List existing plans in TOPS*
TOPSQL Statement	LIST PLAN(S)

```
TOPS:\>list plans
Executing command: list(plans)
command code: 4

LIST of existing PLANS
------

PL$1$1$, id: 1, status: STOPPED
PL$2$1$, id: 2, status: STOPPED
PL$21$1$, id: 21, status: STOPPED
PL$41$1$, id: 41, status: STOPPED
PL$42$1$, id: 42, status: STOPPED
PL$61$1$, id: 61, status: ACTIVE
PL$81$1$, id: 81, status: ACTIVE

-------
end PLAN list

TOPS:\>
```

In TOPS, patient plan names are automatically generated and have the general form: *PL$<patient-id>$<category-id>$.* For example, in the above listing, PL$21$1$ is the name of a patient plan belonging to the patient whose TOPS ID is 21 and is in the category whose ID is 1.

I.5. The composition of a plan for a given patient

Query	*For a given patient, show the current plan*
TOPSQL Statement	SELECT plan FOR patient WHERE patient.id = k

```
TOPS:\>query
Executing command: query()
command code: 13
QUERY:\>  --->select plan for patient where patient_id=41
processing query ...
launching specialised query handler ...
Source cond: PATIENT_ID=41
Target cond:
Executing PLAN query ...
Plan ID: 41

PLAN [ PL$41$1$ ] for PATIENT id [ 41 ]
-------------
[ Rule nn ---> id, name, type  ]
Rule 1 ---> 81, PL$41$1$AUS2, DYNAMIC
Rule 2 ---> 82, PL$41$1$AUS3, DYNAMIC
Rule 3 ---> 83, PL$41$1$OIS2, DYNAMIC
Rule 4 ---> 84, PL$41$1$OIS3, DYNAMIC
Rule 5 ---> 85, PL$41$1$OIS4, DYNAMIC
Rule 6 ---> 86, PL$41$1$0IS5, DYNAMIC
Rule 7 ---> 87, PL$41$1$MAS2, DYNAMIC
Rule 8 ---> 88, PL$41$1$MAS3, DYNAMIC
Rule 9 ---> 89, PL$41$1$MAS4, DYNAMIC
Rule 10 ---> 90, PL$41$1$MAS5, DYNAMIC
Rule 11 ---> 91, PL$41$1$CMA5, DYNAMIC
Rule 12 ---> 92, PL$41$1$CMA6, DYNAMIC
Rule 13 ---> 93, PL$41$1$OIS1, DYNAMIC
Rule 14 ---> 94, PL$41$1$MAS1a, DYNAMIC
Rule 15 ---> 95, PL$41$1$MAS1b, DYNAMIC
Rule 16 ---> 96, PL$41$1$CMA1, DYNAMIC
Rule 17 ---> 97, PL$41$1$CMA2, DYNAMIC
Rule 18 ---> 98, PL$41$1$CMA3, DYNAMIC
Rule 19 ---> 99, PL$41$1$CMA4a, DYNAMIC
Rule 20 ---> 100, PL$41$1$CMA4b, DYNAMIC
Rule 21 ---> 101, PL$41$1$NPH1, DYNAMIC
Rule 22 ---> 102, PL$41$1$main$AUS1, STATIC
--------------
END PLAN [ PL$41$1$ ]

QUERY:\>  --->exit
TOPS:\>
```

The query in the above listing provides only a minimum amount of information about rules in a patient plan. The patient plan rules exist in the database as database triggers. It is possible to modify the implementation of this query to provide the SQL specifications of each plan rule but the query result is not easy to read using the current TOPS command line interface. NB: In TOPS, rule names are automatically generated and have the general form: {<plan-name>|<schedule-name>}<name-or-rule-in-protocol-spec>. Since the plan name is unique (because its composed from patient ID and category ID), the rule name is guaranteed to be unique also.

I.6. The patient plan at a given time or interval

Query	*For a given patient, what was the plan at a given time point t or interval [t1, t2]?*
TOPSQL Statement: (where t, t1 and t2 are time-stamps and k is the patient's id number)	SELECT plan FOR patient WHERE TARGET: **t** ; SOURCE: patient.id = **k**
	SELECT plan FROM snapshot WHERE TARGET: **t1**, **t2**; SOURCE: patient_id=**k**

```
TOPS:\> query
QUERY:\>  --->SELECT PLAN FROM SNAPSHOT WHERE TARGET:2004-7-19
01:55:02,2004-7-19 01:55:58; SOURCE:PATIENT_ID=25
processing query ...
launching specialised query handler ...
Source cond: PATIENT_ID=25
Target cond: 2004-7-19 01:55:02,2004-7-19 01:55:58
Executing PLAN query ...
Getting plan snapshot ...
Plan ID: 23

Plan snapshots exist for times:
No plan snapshot in given interval.
Providing current snapshot, instead:

PLAN [ PL$25$1$ ] SNAPSHOT @[2004-07-19 22:30:26.91]
[rule 1]--->[ 72, PL$25$1$AUS2, DYNAMIC, READY ]
[rule 2]--->[ 73, PL$25$1$AUS3, DYNAMIC, READY ]
[rule 3]--->[ 74, PL$25$1$OIS2, DYNAMIC, READY ]
[rule 4]--->[ 75, PL$25$1$OIS3, DYNAMIC, READY ]
[rule 5]--->[ 76, PL$25$1$OIS4, DYNAMIC, READY ]
[rule 6]--->[ 77, PL$25$1$0IS5, DYNAMIC, READY ]
[rule 7]--->[ 78, PL$25$1$MAS2, DYNAMIC, READY ]
[rule 8]--->[ 79, PL$25$1$MAS3, DYNAMIC, READY ]
[rule 9]--->[ 80, PL$25$1$MAS4, DYNAMIC, READY ]
[rule 10]--->[ 81, PL$25$1$MAS5, DYNAMIC, READY ]
[rule 11]--->[ 82, PL$25$1$CMA5, DYNAMIC, READY ]
[rule 12]--->[ 83, PL$25$1$CMA6, DYNAMIC, READY ]
[rule 13]--->[ 84, PL$25$1$OIS1, DYNAMIC, READY ]
[rule 14]--->[ 85, PL$25$1$MAS1a, DYNAMIC, READY ]
[rule 15]--->[ 86, PL$25$1$MAS1b, DYNAMIC, READY ]
[rule 16]--->[ 87, PL$25$1$CMA1, DYNAMIC, READY ]
[rule 17]--->[ 88, PL$25$1$CMA2, DYNAMIC, READY ]
[rule 18]--->[ 89, PL$25$1$CMA3, DYNAMIC, READY ]
[rule 19]--->[ 90, PL$25$1$CMA4a, DYNAMIC, READY ]
[rule 20]--->[ 91, PL$25$1$CMA4b, DYNAMIC, READY ]
[rule 21]--->[ 92, PL$25$1$NPH1, DYNAMIC, READY ]
[rule 22]--->[ 93, PL$25$1$main$AUS1, STATIC, EXECUTING ]
END SNAPSHOT FOR PLAN PL$25$1$.

QUERY:\>  --->exit

TOPS:\>
```

In the above query, a patient plan at a given time or interval refers the plan's composition and status of its rules at that time or interval. In executing the above query, TOPS first determines if at least one patient plan snapshot exists within the interval specified in the query. If the patient plan snapshot does not exist, the query returns the plan's snapshot at the time this query is being processed.

I.7. Test orders recommended by TOPS for a Given Patient on MAP

Query Description	*For a given patient, what test orders were made during the interval [t1, t2]?*
TOPSQL Statement	SELECT order FOR patient WHERE TARGET: t1, t2; SOURCE: patient_id=k

```
TOPS started.
Initialising ...
Creating log files ...
Provide network_name or ip_address for:
server --->IBMT20
client(this computer) --->IBMT20
TOPS database name --->TOPS
Initialisation complete.

TOPS Command Line Facility, version 1.0.
Copyright(C) 2000-2003, K. Dube
Computer Science Department, School of Computing, DIT, Ireland.
Started at: Wed Jul 21 05:28:13 BST 2004
TOPS:\>QUERY
Executing command: QUERY()
command code: 13
QUERY:\>  --->SELECT ORDER FOR PATIENT WHERE TARGET:2004-7-16 17:48:30,2004-7-16
17:51:25; SOURCE:PATIENT_ID=61
processing query ...
launching specialised query handler ...
processing ORDER query [ 2004-7-16 17:48:30,2004-7-16 17:51:25 ] for [ PATIENT ] ...

Tests ordered for [PATIENT_ID=61] during time interval [2004-7-16 17:48:30,2004-7-16 17:51:25]
Dip_stick_urine Profile, DSU, 2004-07-16 17:49:28.0
Urinary_Tract_Infection Profile, UTI, 2004-07-16 17:50:06.0
Urinary_Tract_Infection Profile, UTI, 2004-07-16 17:50:06.0
-------------------
End test listing.

QUERY:\>  --->EXIT
TOPS:\>
```

The TOPSQL query illustrated in the above TOPS session provides information on what tests where ordered with respect to the specified patient during the given time interval. The query *target* is the *order* while the *source* is the *patient*. The target condition is a time interval, which means that the orders of interest must first belong to the patient with ID 61 and must fall within this time interval, [2004-7-16 17:48:30, 2004-7-16 17:51:25]. The term *order* in the query can be generalised to *rule-action* so that one can obtain information on rule actions that have been performed during the specified time interval.

I.8. The rule responsible for a given test order resulting from MAP plan

Query Description	*Which rule originated a suggestion for and order whose ID in TOPS is xxx?*
TOPSQL Statement	SELECT rule FOR *test_order* WHERE test_order.id = xxx

```
TOPS started.
Initialising ...
Creating log files ...

Provide network_name or ip_address for:
server --->IBMT20
client(this computer) --->IBMT20

TOPS database name --->TOPS
Initialisation complete.

TOPS Command Line Facility, version 1.0.
Copyright(C) 2000-2003, K. Dube
Computer Science Department, School of Computing, DIT, Ireland.
Started at: Wed Jul 21 21:12:56 BST 2004
TOPS:\>QUERY
Executing command: QUERY()
command code: 13

QUERY:\>  --->SELECT RULE FOR ORDER WHERE SOURCE:ORDER_ID=50
processing query ...
[2004-07-21 21:21:49.343] Generic query handler started.
[2004-07-21 21:21:49.353] Specialised RULE query handler started ...

Order [ORDER_ID=50] was suggested by the following rule:
-------------------------------------------------------------------------
rule_name: PL$41$1$MAS1a,
rule_id: 94
order_execution_date: 2004-07-16 04:20:46.0
-----------------------------------------------------------

QUERY:\>  --->exit
TOPS:\>
```

This query returns the *ID, name* and *execution date* for a rule whose execution resulted in the suggestion for an (test) order. In this example, the TOPS *id* for the order is 50.

J. The TOPS Mechanism for Translating ECA Rules to Oracle Database Triggers

Figure 96 illustrates the specification of the ECA rule, MAS5, as it appeared in the PLAN specification of the protocol, MAP.

```
RULE MAS5,
DESCRIPTION: if ACR > 200 mg/l then refer patient to nephrologist for possible proteinuria,
ON: RESULT_ARRIVAL('ACR'),
IF: ACR%RESULT%DATABASE%T_RESULTS  > 200%DOUBLE,
DO: PATIENT_STATE('nephrology_referral' );
```

Figure 96 The rule MAS5 from the MAP specification

The specification of the rule MAS5 after parsing the MAP specification. The attributes of the rule at this stage are held in a Java object. The rule specification is returned by the toString() method of the PDRule() class.

RULE_NAME: MAS5;
DESCRIPTION:
if ACR > 200 mg/l then refer patient to nephrologist for possible proteinuria;
ON: RESULT_ARRIVAL('ACR');
IF: ACR > 200.0;
DO: PATIENT_STATE('nephrology_referral');

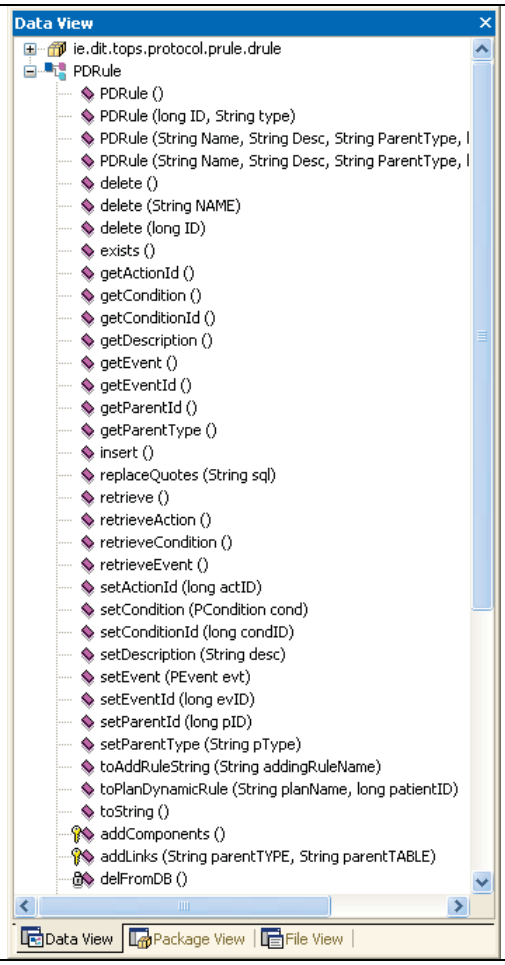


Figure 97 The rule MAS5 after processing by the TOPS protocol specification parser together with the Java class whose instance is an output of the parser

Figure 98 illustrates the Oracle database trigger SQL code for the rule, MAS5, generated by TOPS during the creation of a patient plan. This translation of MAS5 to a database trigger is done by TOPS and may involve prompting for user input. The trigger has a number of customisations. As illustrated in Figure 98 the rule name has been translated from just MAS5 to PL$81$1$MAS5 where 81 is the patient's ID and 1 is the category ID. This ensures that the rule name is unique within the database, which is a requirement imposed by the Oracle DBMS and useful for the management of patient plans in TOPS.

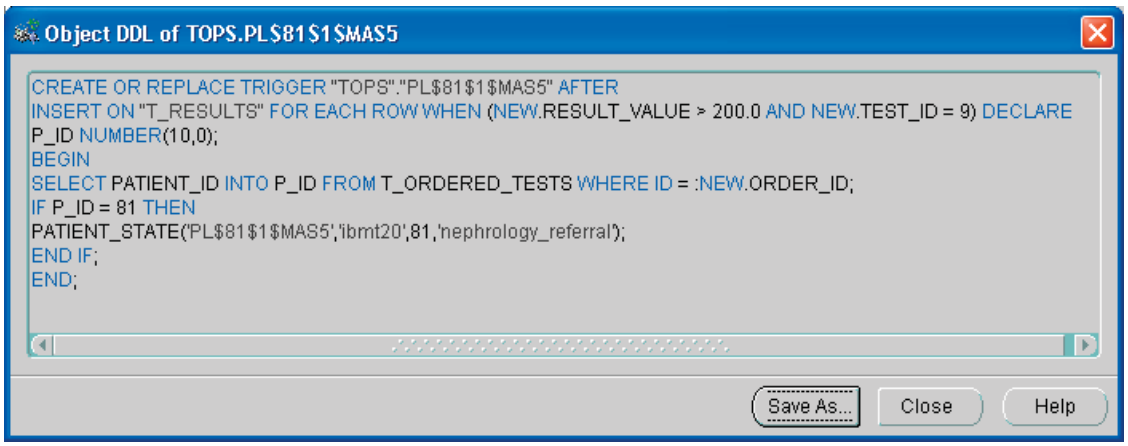


Figure 98 The rule MAS5 translated to the Oracle database trigger, PL$81$1$MAS5

The event result_arrival('ACR') has been translated into two parts: the first part is the database triggering event INSERT ON T_RESULTS and the second part is the condition, NEW.TEST_ID=9, where 9 is the TOPS ID for the ACR test. Thus the rule is now able to monitor the arrival of ACR results. A further customisation has been done to ensure that the rule performs a change in the state of the specific patient to whom the result belongs. The rule is now more specific than what it was in Figure J.2 The MAS5 rule action invokes an Oracle stored procedure, PATIENT_STATE, An Oracle Java call specification, which is an interface to a Java stored procedure within the DBMS. Figure 99 illustrates the Oracle SQL code for the Java call specification for the PatientState() java stored procedure (JSP).

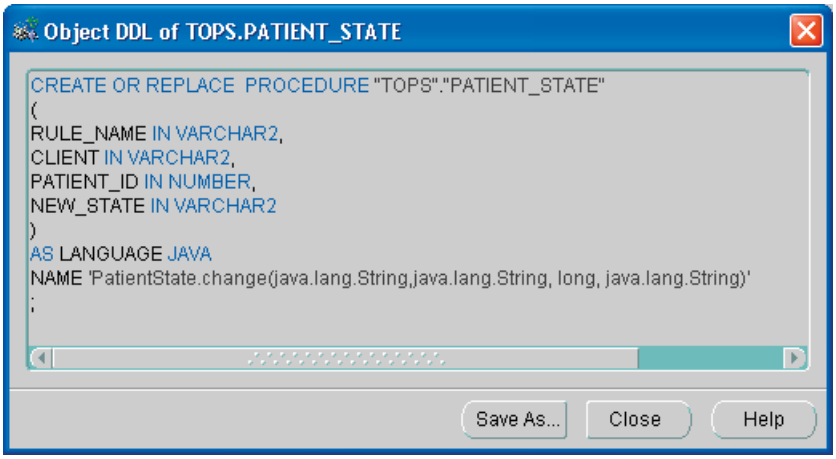


Figure 99 The MAS5 rule action, PATIENT_STATE, in the form of the Oracle Java Call Specification

The Java stored procedure has the method, change(), which updates the patient state in the database and sends a message to an external TOPS module, an instance of the Listener() class, through another Java stored procedure, the Notifier().

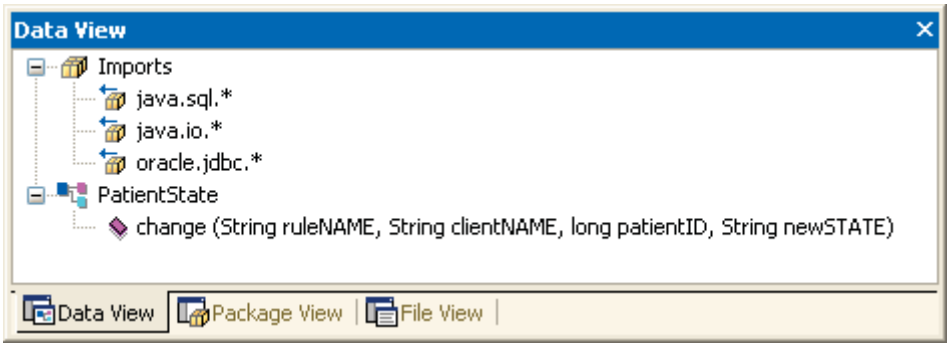


Figure 100 PatientState() Oracle Java stored procedure effecting changes to patient state during protocol execution in TOPS

Figure 101 illustrattes the TOPS the Notifier-Listener mechanism, which allows database triggers to communicate with external modules. Such communication is not achievable by using the JDBC, which is unidirectional in terms of initiating communication. JDBC allows communication to be initiated only from outside the DBMS, thus rendering the database passive. The Notifier class sends text messages to the Listener() through an HTTP connection via a point that is being constantly monitored by the Listener(). (NB: The is nothing to prevent XML messages to be exchanged between the Listener and the Notifier and beyond.)

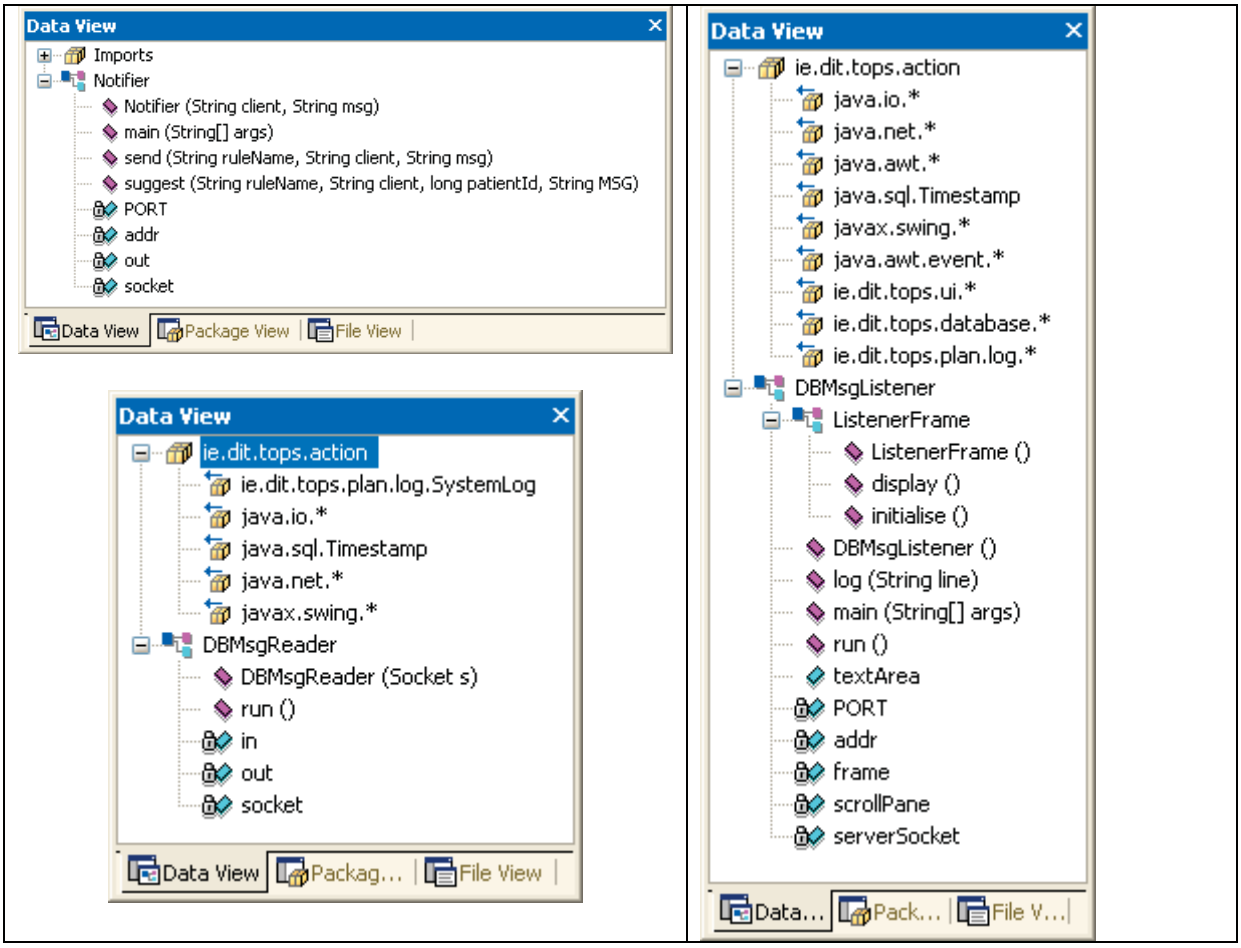


Figure 101 The TOPS mechanism for database trigger communication with TOPS modules outside the Oracle DBMS

The Listener() is implemented as an HTTP server and runs outside the Oracle DBMS. The Notifier is an HTTP client that connects to the Listener() only when a rule executes and invokes it to send a message to modules of TOPS running outside the Oracle DBMS. On accepting a connection, the Listener() invokes a DBMsgReader(), which accepts the data that is being sent by the Notifier(). After accepting the message from a database trigger, the DBMsgReader(), passes the message to a message processor, the DBMsgProcessor(), whose function is to parse the message and determine the agent to which the message needs to be delivered. These agents constitute TOPS's mechanism for extending the database trigger mechanism.

K. The TOPS Command Line Interface

Currently, TOPS uses a command line interface, illustrated in Figure 102, that brings the three planes in the SpEM framework (see Chapters 3 and 5) together with the aim of making them accessible through the manipulation language, TOPSQL. TOPS is currently run by executing the Java class, myprojects.tops.TOPS.class with the root classpath <drive>\TOPS\classes. The Oracle JDBC driver will need to be added to the class path. During initialisation, TOPS will solicit for network names for the Oracle database server and the client machines as well as the user name and password for accessing the database. TOPS needs to have a database account with a password and rights to create, modify, delete and enable/disable database triggers and to make external socket connections.

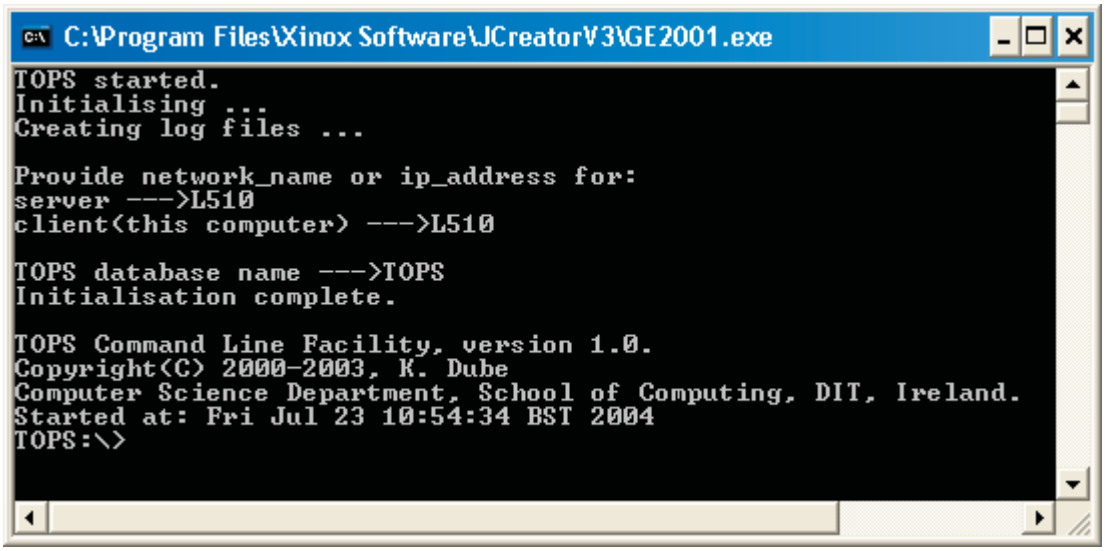


Figure 102 The TOPS command line utility

L. TOPS System Packages

Figure 103 illustrates the constituent packages for TOPS. It can be seen that TOPS is a complex system that consists of forty separate Java packages, the ie.dit.tops.*, myprojects.tops.* and ie.tcd.cs.* packages. The ie.tcd.cs.kdeg.medilink.* and ie.dit.tops.medilink.* packages contain modules for integrating TOPS to other healthcare system within the MediLink Project. The other TOPS packages can be grouped into the three planes of the SpEM framework presented in the book. The *specification plane* is supported by the ie.dit.tops.protocol.* packages. The *execution plane* is supported by the ie.dit.tops.plan.* packages. The *manipulation plane* is supported by the ie.dit.tops.topsql.* packages. Work on the GUI-based user interface for TOPS was initiated but remains incomplete and can be found in the ie.dit.tops.ui.* packages. The utility package, ie.dit.tops.util.*, contain modules for supporting tasks such as printing and sending e-mails to clinicians.

Figure 103 TOPS system packages for supporting the SpEM framework