Workflow Modelling of Coordinated Inter-Health-Provider Care Plans.

by

Eric Donald Browne

School of Computer and Information Science
University of South Australia
Adelaide

A thesis submitted to the
Division of Information Technology, Engineering and the Environment
in fulfilment of the requirements for the degree of
Doctor of Philosophy

January 2005

The research for this PhD was conducted under the supervision of:

Assoc. Prof. J.R. Warren
and
Prof. M. Schrefl.

Revised edition: June 2005

Author email: eric@montagesystems.com.au

ABSTRACT

Workflow in healthcare, particularly for the shared and coordinated management of chronic illnesses, is very difficult to model. It is also difficult to support via current Clinical Information Systems and current information technologies. This dissertation contributes significant enhancements to the current methodologies for designing and implementing Workflow Management Systems (WfMSs) suitable for healthcare. The contribution comprises three interrelated aspects of workflow system architecture as follows:-

- Firstly, it shifts the emphasis of workflow modelling and enactment to a focus on goals, and the monitoring and facilitation of their achievement.
- Secondly, it introduces the concept of self-modifying workflow in the context of health care planning, whereby explicit tasks in the goal-based care plan are devoted to assessing and modifying downstream workflow.
- Thirdly, this dissertation proposes methodologies for identifying and dealing with tasks which overlap, subsume or interfere with other tasks elsewhere in a given workflow.

The language and methods introduced in goal-based requirements engineering research have been carried into the domain of WfMSs and adapted by the author as a mechanism for deriving workflow models that can be communicated and enacted by health care providers contributing to the shared care of a patient. A methodology is described, whereby a hierarchical goal-based view for the management of a chronic condition or conditions can be automatically translated into a workflow schema. This workflow schema contains subworkflows corresponding to each goal, together with specific tasks dedicated to monitoring, and, if necessary, altering the downstream workflow to optimally achieve each goal target.

For self-modifying workflow, certain tasks in the workflow schema are devoted to modifying the downstream workflow on an instance by instance basis. Such self-modifying schemas provide the necessary flexibility to suit the evolving diagnostic and therapeutic processes encountered in Chronic Disease Management (CDM), particularly in complex areas requiring significant individualisation. The management of Diabetes Mellitus in a community care setting provides an example to illustrate this complexity. In order to facilitate self-modification of workflow schemas, this dissertation enunciates a set of valid operations that can be applied to downstream components of a workflow schema. These operations are primarily concerned with turning abstract subworkflows into concrete ones through completion and alteration of template primitives.

There are many situations in inter-organisational health care, where, for a given care process, activities might be undertaken in one clinic that overlap with, or repeat activities undertaken elsewhere. This dissertation proposes solutions to situations where duplicated tests and procedures are costly and can have negative health impacts on patients undergoing unnecessary tests and interventions. The approach builds on the two-tier goal/process representation of healthcare processes and describes an execution model comprising a candidate discovery phase, followed by a component crediting phase. The notions of full vs. partial crediting, and goal-level vs. activity-level crediting are introduced. The role that temporal constraints play in determining candidate components for crediting is also examined.

Aspects of a prototype Workflow Management System (called *Stream*Line) that the author has built, are described in order to illustrate how the approach of goal-based workflow schema derivation, self-modifying worklow schemas, and activity overlap identification and crediting can provide sufficient flexibility and focus to substantially improve the management of complex, chronic conditions.

The author's prototype is tested using the current local work practices for treating Non-Insulin Dependent Diabetes Mellitus involving shared care plans based on Australian guidelines.

The dissertation concludes with an assessment of the implications of goal-based, self-modifying, redundancy-reducing workflow models for developers and implementors of WfMSs as well as for implementors of future Health Information Networks employing such complex workflow solutions.

DECLARATION

I declare that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of knowledge it does not contain any materials previously published or written by another person except where due reference is made in the text.

Signed _____ . Date _____ .

Contents

List of Figures

List of Tables

Glossary

AWDGP Adelaide Western Division of General Practice

BMI Body Mass Index

BP Blood Pressure

BPEL Business Process Execution Language

CDM Chronic Disease Management

CIS Clinical Information System

COPD Chronic Obstructive Pulmonary Disease

DAML-S DARPA Agent Markup Language - for Services

DSS Decision Support System

ECA Event/Condition/Action

EDS Electronic Decision Support

EDSS Electronic Decision Support System

EHR Electronic Health Record

GP General Practioner

HbA1c Haemoglobin A1c (glycated haemoglobin)

KAOS Knowledge Acquisition in autOmated Specifications

MBS Medical Benefits Scheme

MCPOP Modelling the Clinical Processes Of Prescribing

NIDDM Non-insulin-dependent Diabetes Mellitus

OMG Object Management Group

OWL-S Web Ontololgy Language - for Services

QoS Quality of Service

RACGP Royal Australian College of General Practioners

W*f*MC The Workflow Management Coalition

W*f*MS Workflow Management System

WHO World Health Organisation

ACKNOWLEDGEMENTS

I gratefully acknowledge the support of the South Australian Department of Health (formerly Human Services) and the Australian Research Council [1], who have contributed financially to the research undertaken as part of this thesis.

My two supervisors, Associate Professor James (Jim) Warren, and Professor Michael Schrefl have provided great guidance over the years. I have benefited both from Jim's focus on ensuring that theory is always appropriately directed towards improving outcomes for patients, as well as Michael's emphasis on ensuring that ideas are pursued and analysed in order to establish if they are well-grounded and can move beyond "mere ideas".

My partner, Julie, not only supported and encouraged my research, but instigated and motivated my interest in improving healthcare from the outset. Her own struggle with cancer, and with a healthcare system that from an information management perspective is archaic and inefficient, led me to analyse and try to make some small contribution to improving the care process. Her knowledge of the Australian healthcare scene was, and is, invaluable.

And to the others that encouraged, gave insight, criticised, or suffered in silence along the way, I thank them all for their contributions.

[1] This work was supported by Australian Research Council SPIRT (Strategic Partnership with Industry Research and Training) grant C00107117 in partnership with the South Australian Department of Human Services.

CHAPTER 1

Introduction

"If I'd known I was going to live this long, I'd have taken better care of myself."

James Herbert Blake (1883–1983).

1.1 Motivation

According to the World Health Organisation (WHO), chronic non-communicable diseases currently account for some 60% of global deaths and almost half the global burden of disease. When one considers the burden of disease in terms of the total disability-adjusted life years (DALYs), one finds chronic conditions to be the major contributor, even in wealthy countries such as Australia. According to recent studies undertaken by the Austalian Institute of Health and Welfare [Mat99], twelve chronic diseases and conditions accounted for an estimated 42% of the total disability-adjusted life years lost in Australia in 1996, and all such diseases and conditions accounted for about 80% of DALYs. Much effort has been put into examining the effectiveness of different regimes for the management of these chronic conditions. Many professional health organisations and government health agencies have produced evidence-based guidelines in order to provide health care practioners with appropriate knowledge or "best practice" in chronic

disease management. However, there are still significant challenges in ensuring the utilisation, coordination and continuity of these "best practices" across organisations involved in a patient's shared care, and over the long time scales encountered with chronic illnesses.

1.2 Problem Description

Workflow Management Systems (W*f*MSs) offer the potential to help ensure that tasks are performed as and when required, and that unnecessary tests and interventions are not undertaken if they have already been performed by another clinician or clinic responsible for the patient's care. In order for W*f*MSs to successfully assist clinicians in the management of chronic conditions, a number of significant hurdles associated both with workflow modelling approaches, as well as with system implementations, need to be overcome. The specific problems that this thesis addresses are:

- The non-deterministic nature of health and healthcare requires a shift in the focus (as might pertain in other domains) from a set of prescriptive tasks whose enactment will ensure the overall objective, to a set of flexible processes aimed at achieving, assessing the achievement of, and if necessary, altering aspects of those processes in order to reach an optimal outcome for each individual.

- In many chronic disease management cases, there are a number of healthcare providers co-operating in the care of the patient. This often leads to the duplication of tests and treatments across inter-organisational boundaries. W*f*MSs need to clarify the characteristics of tasks in such a way that clinicians (and in some cases, systems) can be made aware of potential du-

plication. Furthermore, WfMSs should be able to provide mechanisms to assist in the resolution of these potential duplications.

In approaching these problems, the author noted that most of the research related to distributed workflow systems has been based on system implementation architectures, whereas most research related to adaptive and dynamic workflow systems is based on object-oriented process modelling. There has been little published research which bridges the gap or ties these two approaches together. This thesis tries to overcome this dichotomy, by presenting new approaches to both workflow modelling and implemention methodologies.

1.3 Thesis Outline

The thesis is presented in three parts, namely **I - background**; **II - approach, theory & solutions**; and **III - practice**.

Part I - chapter 2 and **chapter 3** give an overview of health informatics issues related to chronic disease management and current workflow-related technologies respectively. These chapters also address related work in the field, which has informed and guided the author towards his solutions.

Part II - chapter 4 explains the overall approach taken to improve the design and modelling of WfMSs for use in healthcare generally, and chronic disease management in particular. **Chapters 5, 6** and **7** explain in detail, the modelling and theoretical ideas of the author which address the specific problems alluded to in Part I. These chapters show how WfMSs can be reengineered and enhanced to provide a flexible goals-focused solution to meet individual patients' needs. In particular, **chapter 5** illustrates the importance of goals, and how a goal-focussed view of a care process can be used to automatically generate a nested skeleton workflow schema with goal-assess, alter and act tasks. **Chapter 6** articulates a

new paradigm for individualisation and dynamic adaption of workflow schemas which retains the goal-focussed approach discussed in chapter 5. **Chapter 7** introduces a methodology for "crediting" overlapping or redundant activities, that can commonly occur where multiple providers share in the overall care process of a patient.

Part III - chapters 8 and **9** address the practical application and evaluation of the author's approach, using a prototype workflow engine *Stream*Line, developed by the author, that incorporates many of the design solutions described in Part II. A case-study is presented in **chapter 9**, which shows how *Stream*Line can help coordinate and focus the workflow required to support the inter-health-provider managment of Non-Insulin Dependent Diabetes Mellitus(NIDDM).

Final conclusions, observations, issues and future directions are discussed in **chapter 10**.

Part I

Background

Chapter 2 - Computing in Healthcare

Chapter 3 - Current Workflow-related Technology

CHAPTER 2

Computing in Healthcare

"*Life is short, but its ills make it seem long.*"

Pubilius Syrus, Sententiae, 1st century BC.

2.1 Introduction

This chapter examines the healthcare context for applying workflow-based information systems, starting with a comparison between Clinical Guidelines, Care Plans and Workflow. This is followed by a critical analysis of the obstacles faced when trying to apply traditional Workflow Management Systems (WfMSs) to the domain of healthcare. The role of Clinical Information Systems, more generally, is discussed, in order to illustrate how workflow meshes with other components of health information systems. Finally, the relationship between Electronic Health Records and Workflow Management Systems is explained, in order to highlight the interdependence between data and processes respectively.

2.2 Clinical Guidelines, Care Plans and Workflow

Clinical Guidelines stem mainly from the goal of ensuring quality and consistency in individual patient treatment, for a specific, target illness. They do so by representing standardised treatment pathways for the illness, with decision points based on various values of a generic patient's state. They focus on diagnoses and "what" needs to be done to treat the illness. Although Clinical Guidelines were originally documented in normal, written language and disseminated on paper, they are increasingly being represented in new computer-interpretable formats and executable languages. As such, they are becoming candidates for integration into electronic patient management systems and decision support systems. Care plans, which are often based on clinical guidelines, have individualisation as a prime focus, i.e. adapting care to suit an individual's circumstances. They are somewhat akin to being an instance of a guideline implementation and often involve long timescales such as weeks, months, or years, usually involving chronic illness, and often more than one care provider. Care plans would not normally be applied in acute care situations. They focus on "when" actions need to occur.

Workflow Management Systems have historically aimed at monitoring and coordinating the activities associated with well defined business processes, according to pre-defined "process definitions". The component activities or tasks, might include decision-making, notification, synchronisation etc. and can either be launched automatically by computer, or placed on worklists to be initiated manually. Workflow Systems, rather than concentrate on the detail of tasks, provide an environment to automate and assist the management of tasks and the flow of work associated with a case from one task to the next. They encompass authorisation, authentication, roles, scheduling, monitoring, resourcing,

event processing, queue management, prioritisation, escalation, notification, task suspension/termination, auditing. In the past few years, there has been a growing belief that workflow technologies could benefit many areas of healthcare and as such, just like the Electronic Health Record (EHR), should form a foundation stone for the next generation of health information systems. Guidelines, Care Plans and Workflow are beginning to coalesce - their domains of applicability are widening, and their ranges of functionality are broadening. With the advent of better messaging systems, and better patient data repositories, we have reached a stage where a workflow-enabled management system can interpret a patient's care plan and play an active role in initiating and monitoring healthcare activities in accordance with appropriate clinical guidelines.

2.2.1 Guidelines

Clinical Practice Guidelines are "systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific clinical circumstances" [Fie90]. Guidelines are either evidence-based, or formulated by consensus from the opinions of "experts" in the field of the illness. A guideline is applicable to all patients suspected of suffering from the target illness. Clinical Guidelines have been adopted for the treatment of chronic illnesses such as diabetes and asthma, where the treatment regime is moderately complex, yet fairly well prescribed and agreed upon by the medical community. Guidelines were originally represented either in text form or as an annotated flow diagram, where each component of the diagram encapsulated a test, treatment or decision task that might need to be undertaken. The treatment for a specific patient traces a path through a subset of the guideline steps, depending upon that patient's specific symptoms and characteristics.

Some arguments that have been postulated in favour of using guidelines are:

to reduce variation in practice, thereby improving quality of care; to encapsulate current "best practice" in consistent, communicable formats; to improve resource planning and efficiencies; to provide a more rational basis for referral.

Many attempts have been made to develop computer-based systems to assist health providers in charting a given patient's course through a guideline. These attempts have resulted in computer-interpretable languages, models and tools such as Arden Syntax[Ard02], Asbru[Mik97], EON[Tu01], GEM[Shi00], GLIF[OM98], GUIDE[Qua01], Prestige[Gor97; Gor99], PRODIGY[Pur99], PROforma [Bur00] and Protégé[Sta]. Representation of guidelines in such formats allows for integration with other information management systems such as decision support, domain knowledge repositories, terminology and coding systems, online care plans, workflow, patient management systems, electronic health records, messaging, prescribing, scheduling etc. Computer Interpretable Guideline representations fall into two broad classifications [Wol02], viz. narrative and algorithmic, with the former being easier for domain experts to formulate, whilst the latter are more readily implementable by computer.

Tu *et al* [Tu03] have categorised clinical guidelines according to their process models, citing four distinct types, namely:-

1. flowcharts for capturing problem-solving processes,
2. disease-state maps that link decision points in managing patient problems over time,
3. plans that specify sequences of activities that contribute toward a goal,
4. workflow specifications that model care processes in an organization.

When using Clinical Guidelines, it must be remembered that the guideline encapsulated best evidence-based practice at the time it was developed. If

information systems are built to utilise Guidelines, then there has to be clear differentiation between the system and the domain knowledge, so that updates to Guidelines can be readily incorporated.

The widespread utilisation of Computer Interpretable Guidelines, although seen almost universally as a means of improving both quality and efficiency of healthcare, is still a long way from realisation. Some of the major questions still to be answered are:

- Is there a single optimum Guideline representation language? [Pel03]
- What level of automation is possible/desirable?
- How can we achieve a decoupling between knowledge and process that supports evolving clinical knowledge?
- How can we resolve conflicts between competing/overlapping guidelines for a patient with comorbidities[1]?
- Which components of a health information system should implement/maintain guidelines?

However, as a recent study on the use of computerised evidence-based guidelines [Ecc02] concluded:

> *"Even if the technical problems of producing a system that fully supports the management of chronic disease were solved, there remains the challenge of integrating the systems into clinical encounters where busy practioners manage patients with complex, multiple conditions."*

The vision of "automating" Clinical Guidelines received new impetus in 2002, with the initiation of the SAGE (Standards-Based Sharable Active Guideline

[1] two or more conditions exhibited by a patient at the same time.

Environment) project [SAG02], a collaboration among research groups at IDX Systems Corporation, the University of Nebraska Medical Center, Intermountain Health Care (IHC), Apelon, Inc., Stanford University, and the Mayo Clinic, which seeks to create the technology for integrating guideline-based decision support into enterprise clinical information systems. In SAGE, access to patient data is supported through "Virtual Medical Record(VMR) Services", whilst guideline-based actions are provided by "Action Services", which utilise clinical knowledge encoded in standard medical terminologies (SNOMED CT, LOINC, etc). An attempt has recently commenced [Ram04] to apply the SAGE guideline engine to the enactment of the American Diabetes Association's Diabetes Guideline[Fra02].

2.2.2 Care Plans

As a general term, "care plan" is quite broad, encompassing virtually any description of the steps required to treat the clinical condition of a single patient over an extended period of time. It is normally developed to deal with chronic conditions or conditions involving a multidisciplinary team of carers. In Australia, with the advent of the Enhanced Primary Care (EPC) initiative and corresponding Medical Benefits Scheme (MBS) remuneration scales, a narrower definition [DoH02] has become almost the de facto definition, constrained by the following rules:

- The patient's usual GP is able to develop a plan for the care of the patient with other health providers. This plan provides a documented process for long term care for a patient with a chronic or complex illness with multidisciplinary needs.
- A plan can also be developed for a patient being discharged from hospital.
- The care plan team must include a GP and at least 2 other health providers who contribute a different service.

- A chronic illness is one that is likely to be present for 6 months or more or a terminal illness.
- The GP is able to initiate the plan and include other appropriate health providers. Or, another health provider is able to initiate the plan where the GP contributes.
- The patient must consent to the care plan and receive a copy.
- The plan can only be reviewed at minimum intervals of at least 6 months.

Conditions to which care plans are often formalised in Australia, include: cardiovascular conditions; asthma and COPD[2]; chronic mental conditions; Diabetes Mellitus[3] ; cancer; rheumatic conditions, particularly arthritis.

Frequently, care plans are based on clinical guidelines, and as such, represent the tailoring of a guideline to an individual patient. Since guidelines are disease-specific, care plans can involve the merging of several guidelines, with the consequential need to resolve conflicting recommendations, such as drug interactions, dietary conflicts etc. Care plans also often involve multiple actors in the care process, sometimes referred to as shared care. The coordination of shared care for chronically ill patients places large demands on the information-processing capacity of the GP and the efficiency of communication with other care providers. Many studies [IOM99; Coi00; ACQ01] have demonstrated that communication between clinicians regarding cotreated patients is prone to be delayed, incomplete, or erroneous. Current MBS items for care plans are geared more towards remunerating the GP for initiating the plan, than for covering the costs associated with managing the plan effectively. In fact, the six-monthly limitation on

[2]Chronic Obstructive Pulmonary Disease
[3]particularly Type II Diabetes (NIDDM)

frequency of care plan review (as a means to save money) runs counter to effective patient management for some patient conditions.

Good patient management information systems, network infrastructure and efficient communication systems are all prerequisites for efficient management of interprovider care plans. The Australian Coordinated Care Trials (Round 1: 1997-1999); Round 2: 2002-) were used by a number of participants to test prototype care planning methodologies. In the South Australian HealthPlus Coordinated Care Trial, an online facility for presenting care plans via browser technology was trialed. The software, Care Plan On-Line (CPOL) [War99] assisted practitioners to tailor a "base" care plan for an individual, by augmenting their own knowledge and experience with additional computer supplied knowledge from a range of sources . These sources included:-

- medical and pharmaceutical benefits data
- hospital visit records
- initial medical assessment
- current medications
- ongoing observations
- clinical guidelines

Prototype applications such as this help to illustrate the benefits that can be gained by providing as much relevant information as possible to each clinician at the point of care, and by providing common electronic care plans that can be accessed by any member of the team responsible for a patient's care.

2.2.3 Workflow Management Systems

According to the Workflow Management Coalition[Fis01], Workflow is:

> *the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant*[4] *to another for action, according to a set of procedural rules.*

The primary aim of a Workflow Management System (W*f*MS), is to utilise information technology to assist, and where appropriate automate, activities involved in a specific process. At a minimum, a W*f*MS should be able to facilitate, monitor and audit "who" has done, is doing, or is scheduled to do -"what", "when" and "why" to "whom". Thus, the W*f*MS should be able to provide context to any particular activity. They should help to *ensure* that activities that *should* be undertaken *are* undertaken.

Workflow systems are process-oriented, where a process represents a set of activities that need to occur in a prescribed sequence to achieve an outcome. In health care, activities can be diagnostic, therapeutic, administrative, or decision-making. W*f*MS are designed to be aware of the actors (patients, practitioners, administrators) and sometimes other resources (e.g. operating theatre availability, communication pathways) involved in a given care process. They also must coordinate those resources across successive cases (patients). Since they are designed to play an active role in the health care process, they also need to be highly robust, and secure. All these demands on a Workflow System result in them being highly complex pieces of software.

As can be seen from the above description, it is clear that workflow goes substantially further than implementations of care plans and clinical guidelines, yet there is substantial similarity and perhaps overlap of function.

[4] a participant in this context can be either a human resource, intelligent agent or a computer application.

Aspect	**Clinical Guidelines**	**Care Plans**	**Workflow Systems**
Clinical Knowledge	High	Moderate	Low
Condition-specific	Yes	Mainly	Not usually
Patient-specific	No	Yes	Rarely
Single case	Yes	Yes	No
Security	Low	Low	High
Active/Passive	Mainly passive	Mainly passive	Mainly active
Built-in audit/statistics	No	No	Yes
Inter-provider communication	Little	Moderate	High
Focus	Condition	Patient	Process
Notification support	No	No	Yes
Role-participant support	Low	Medium	High

Table 2.1: *Comparing Clinical Guidelines, Care Plans and Workflow Systems*

An attempt to categorise aspects of all three in such a way as to clarify their roles in the complete healthcare process is shown in Table 2.1. Although this table suggests a delineation between the three approaches that is fixed, this is by no means the case. There are certainly efforts afoot, for Clinical Guideline-based systems, to provide patient-centric care-plans and to provide notification mechanisms. Care providers, and health informaticians at the leading edge of the development of health information systems can well envision the functionality that technology can now provide, and it certainly encompasses all the functionality available from existing Clinical Guidelines, Care Planning and Workflow-based systems and more. Sometimes the term Decision Support System has been extended to cover such functionality, but this stretches the natural definition that should be ascribed to such a term [see 2.4.2]. The author views Clinical Guidelines, Care Plans and Workflow Management as complementary components, which together with Electronic Health Records, Authorisation/Authentication, Patient Identification, Terminology Services and others, will build the comprehensive Pa-

tient Management Systems of the future. The outstanding challenge is "how to design, develop and implement such complex systems in the most efficient, safe and flexible way".

One approach is to extend the functionality of existing systems - development by evolution. This is attractive to existing vendors, system integrators and hospital managers. Changes can be introduced with minimum disruption to existing services, and with reasonably predictable outcomes. A Clinical Guideline system may well then be "enhanced with Workflow capabilities". The opposite approach is a paradigm shift - engineering new systems from the ground up, with functionality that matches the forseeable health needs and resources of the community, but which are based on design principles that cater for, and support the rapidly changing knowledge-base of the health domain. With the opportunities and demands that Health*Connect*[5] and other Electronic Health Record based systems are throwing us, it would appear that a paradigm shift is both opportune and inevitable. From this perspective, it is important to propose clear definitions for these terms.

Thus, a **Clinical Guideline** can be viewed simply as *the representations of clinical best practice that can inform decisions about appropriate health care for specific clinical circumstances.* A **Care Plan** is most appropriately defined as *the set of activities and decisions that need to be undertaken at agreed times by agreed parties to provide best practice care for a specific patient.* And a **Workflow Management System** provides *the software infrastructure to assist customers and service providers in the optimal, provision of best-practice care, by providing worklists of activities, notifications of changes and priorities, and access to care related information pertaining to each patient.* Each has its role in tomorrow's healthcare systems and each can be designed, developed and optimised by appro-

[5]A national, Australian initiative to provide individuals with their own shared Electronic Health Record, which commenced design in 2002 - (http://www.healthconnect.gov.au)

priate domain experts.

Integration

With the above definitions in mind, we can turn our attention to how these components can be combined to build effective healthcare management systems. Dazzi *et al* [Daz97] were one of the first to describe a Patient Workflow Management System (PWfMS) based on Clinical Guidelines and illustrate how such a system could help manage Acute Myeloid Leukemia in children. By logging the tasks actually performed by the clinician into the patient record, analyses were able to be undertaken to determine compliance with the formal Guideline; where non-compliance was observed, the reason for non-compliance was used to inform further refinement of the Guideline. Since management of chronic conditions represents such a significant burden in the Primary Care arena, it is the potential for Workflow and Guidelines to help manage these conditions that is gaining much attention from researchers and Health Information Systems providers/managers. Management of Non-Insulin Dependent Diabetes Mellitus (NIDDM) is a typical candidate for such systems. There are established evidence-based guidelines; the treatment requires a number of repeating activities including diagnostic, decision-making and therapeutic; there are a number of actors involved in the management process (GP, ophthalmologist, endocrinologist, pathologist, dietician, podiatrist, patient). Once certain diagnostic parts of the guideline have been completed, the workflow moves into a "predictable" phase - predictable in the sense that for a given patient, a long term sequence of activities, based on the Guideline, can be articulated to form a care plan. A Workflow Management System can not only play a role in co-ordinating the "categorising" of diagnostic activities, but can play an ongoing management role, based on the care plan, by alerting the patient and providers when interventions are required. The Workflow System can poten-

tially store the care plans, as well as the actual treatment activities undertaken for all patients for whom it is responsible, and if standard data dictionaries are used to define this information, then considerable population data can be acquired to broaden and deepen the evidence base for determining best-practice. This evidence base then feeds back into future revisions of the published guidelines.

However, care plans are not static. They need to evolve and adapt to meet the changing state of the patient, to meet the changing health care environment and to match the evolving best-practice guidelines. *Dynamic evolution* with respect to W*f*MSs refers to changes occurring to a process schema over time. This can be akin to the *modification* of an existing care plan.

In Australia, the embracing of HL7 messaging, the move towards coordinated care under the Enhanced Primary Care (EPC) program and the move towards standardization of hospital discharge summaries have promoted the specification of individual patient care plans. In concrete terms, EPC has resulted in the introduction of a series of Medical Benefits Schedule (MBS) items that allow healthcare providers to make claims to the government health insurance scheme for care planning activities, including the preparation and review of the care plan, as well as their contribution to its enactment[RAC03].

There is, however, a need to bring additional rigour to bear in order to coordinate, monitor and manage the enactment of such care plans. W*f*MSs have the potential to provide this rigour. At the minimum, the workflow can help to ensure that healthcare providers have met the baseline requirements in terms of activities needed to claim the MBS care planning items.

Care Plan enactment often crosses organisational boundaries. In such cases, it is important for each organisation's contribution to the overall workflow to be articulated in terms of desired goals and outcomes for each patient's management. This is necessary in order to avoid conflicting goals and to provide a basis for

modifying specific parts of the workflow to meet the overall objectives.

2.3 Workflow in Healthcare

Workflow has its genesis in the imaging industry, from where it moved into mainstream production environments, such as manufacturing. Workflow aims to monitor and coordinate the activities associated with well defined business processes. It does this by having a Workflow Engine direct the flow of activities, under instructions from a Process Definition that ties together the component activities or tasks , including decision- making, notification, synchronisation etc. Tasks can either be launched automatically by computer, or placed on worklists to be initiated manually.

Workflow Systems, rather than concentrate on the detail of tasks, provide an environment to automate and assist the management of tasks and the flow of workitems from one task to the next. Workflow Management Systems often provide authorisation, authentication, scheduling, resource allocation, monitoring, event processing, worklist management, task prioritisation, escalation, task suspension/termination and auditing.

Collaborative workflows are focused on achieving a common goal and often require that the participants have access to a common set of documents or information and hence are often connected to document-management systems. In healthcare, this repository-based system is minimally, a Patient Identification System (sometimes referred to as a Patient Master Index or PMI) and ideally, a comprehensive patient-centric Electronic Health Record System.

In workflow systems, we normally expect to have defined a given *process*[6] and at run time, we invoke *instances* of the process. In healthcare, we refer to each

[6]Concepts are discussed more fully in 5.2

instance as a patient *case*. The *case* is treated as an object which changes state as it flows through the process. The process is composed of distinct *activities*, which at run time, become *activity instances* or *tasks*. These tasks might be performed automatically by a computer, or they may be placed in a queue for a human, or group of humans to undertake. Such a queue is termed a *worklist*, and each item on the worklist is referred to as a *workitem*. Thus, a *workitem* describes a task to be undertaken by a specific person or role on behalf of a specific patient case.

Workflow tasks in healthcare can be investigative, therapeutic, administrative, or decision-making. W*f*MSs are designed to be aware of each actor (patients, practitioners, administrators) responsible for each activity and sometimes other resources (operating theatre availability, communication pathways, etc.) involved in a given care process. They also must coordinate those resources across successive cases (patients). Since they are designed to play an active role in the healthcare process, W*f*MSs also need to be highly robust and secure.

Workflow systems have been utilised in a number of domains for many years. Productivity and quality gains have encouraged their adoption in more industries. However, before workflow functionality becomes an integral component of health information management systems, there are some domain specific hurdles that need to be cleared. These hurdles are related to various aspects of health and healthcare processes, including, but not limited to the following:-

1. **Distributed** - Traditional production workflow systems were confined to one organisation. In many healthcare situations, a number of care providers may be involved. Each of these separate actors have their own work practices and systems, and so, many workflow systems need to be distributed and inter-institutional. This implies a requirement for introspection i.e. the ability to interrogate a system at runtime for information about itself.

2. **Instance-adaptive** - workflow instances (ie. Individual patient cases in

a care process) need to adapt or even transfer to a different workflow schema. Example: someone is admitted for appendectomy. **Appendectomy Workflow** is instantiated. Patient undergoes surgery and contracts Haemolytic Streptococcal infection, which is diagnosed as malaria. Patient is placed under **Malaria treatment workflow**. Finally, the diagnosis is changed to correct the initial misdiagnosis and **Haemolytic Strep. Treatment Workflow** is invoked. The above three could be treated as separate workflows, but it is likely that some form of messaging will be required between the three instances. In the above example, the workflows are invoked sequentially. Sometimes the workflows are invoked concurrently, due to uncertainty in diagnosis. Another consequence of ad-hoc change, is the requirement for new actors to come on board.

3. **Non-deterministic** i.e. States are not necessarily related to activities or events. This means that traditional object life-cycle models are inadequate. Traditional (Petri-net based) models of object life-cycles model closed systems and so assume that a *state* results from a *transition*. In healthcare, we are not dealing with closed systems and neither are we dealing with causal systems (at least not in the macro observable level). It may be true for some activities, but not necessarily for others. E.g. Activities which are measurements are deterministic: transition.take_Xray ⇒ place.Xray_taken, but activities that invoke actions often are not: transition.lower_BP[7] may not result in place.BP_lowered. A second example is the frequent occurrence of state changes without any modelled event - particularly state changes to the patient unforeseen and unforeseeable by the modelled workflow. Often these state changes occur between activities when the patient is assumed to be in some sort of quiescent state. Patients are not passive objects like

[7]BP denotes Blood Pressure.

cars or insurance claims, but are extremely complex active objects whose intrinsic state changes must often be taken into account by each activity in the workflow.

4. **Irreversible** - not only are most health-related activities non-deterministic, they are almost always irreversible. Once a patient's leg has been amputated, it cannot be restored. Once a patient has undergone medication treatment, the medication cannot be extracted from the body, and its effects reversed. Compare this with reversible activities often encountered with simpler domains, such as claims-processing workflows, whereby authorizations can be revoked, etc. This implies that for most healthcare workflow processes, the methodologies of rollback and reversal/compensation are not suitable.

5. **Based on expectations** i.e. Care plans are about intent rather than result. Due to the non-deterministic nature of health, coupled with the fact that diagnoses are often incomplete and treatments generalised from population studies, most processes "try" to achieve certain outcomes. Post-conditions of activities sometimes need to be viewed as "expected", rather than being strictly enforced. Depending on the nature of the activity, the condition being treated, and the overall process, the way post-conditions are specified and handled needs special consideration.

6. **Complex** - in terms of:-

    - the number of activities.
    - the number of attributes.
    - uncertainty of attributes.
    - events are often analog and long-lasting.

- the patient (or some part thereof) is a complex object.
- roles can be ad hoc in nature.

7. **Properties and behaviour of actors are not defined a priori** . Actors (clinicians etc.) should probably be modelled as objects, whereas traditional workflow systems tend to ignore the properties and behaviour of participants and roles in the workflow process. Healthcare actors possess attributes and behaviour, such as current role, expertise, availability, ability to prescribe etc.

2.3.1 Specific Workflow in Healthcare Implementations

This section examines some specific implementations of workflow management systems in the domain of healthcare. For information on other implementations, prototypes, and research projects, the reader is referred to the author's website, devoted to Workflow in Healthcare [Bro04][8]

OzCare

OzCare[Lee96] was one of the earliest attempts to bring Workflow automation technology to support the description and enactment of care plans, in this case for use in hospitals. OzCare utilises Columbia University's low level decision steps called Medical Logic Modules (MLMs), developed to support clinical guideline execution in conjunction with the guideline language Arden Syntax. Template care plans are built from rules applied to *care elements*, to which appropriate MLMs are attached. These template care plans can be tailored to individual patients using *add* and *delete* operators. Functionality was later added to support event and alarm queues, using a simple Unix timer. OzCare's implementation was

[8] http://workflow.healthbase.info

based on a separation of concerns - high level control flow (what happened and when) was handled by the Oz rules language, whereas detailed clinical knowledge associated with specific tasks was embodied in the MLM's, and served from a separate machine.

MOBILE

MOBILE [Jab94] was a modular W*f*MS designed to support adaptation of workflow to meet individual needs at runtime, using a late modelling approach referred to as *descriptive modelling* (in contrast to traditional *prescriptive modelling*), whereby aspects of the workflow model, such as choosing an appropriate activity or subworkflow, cannot be determined until execution time for a given case. The descriptive modelling constructs introduced in MOBILE include *temporal constraints* (delay, deadline) and *existence constraints*.

CareFlow

CareFlow[Qua00; Qua01] is the name given to the approach developed by Stefanelli, Quaglini *et al* at the University of Pavia, Italy for the application of W*f*MSs to healthcare, based on clinical guidelines. Many of the CareFlow projects have emphasised the need to capture sufficient clinical knowledge to meet the real needs of specific conditions, adapted to specific patients, and to represent this knowledge as workflow schemas. Exception handling, as a mechanism to support safety aspects, and as a reflection of the complexity encountered in healthcare, has played an important role in these projects, which have been tested and refined for the management of patients who have suffered from ischemic strokes and more recently, for the management of chronic conditions such as diabetes. CareFlow projects are complete applications and combine W*f*MSs with clinical guidelines (as a source of process definition) and electronic health records (as a source and

sink for data).

METEOR

METEOR (**M**anaging **E**nd-**T**o-**E**nd **O**pe**R**ations) is a long running project, commenced in 1991, to support coordinated distributed processes in a number of domains, including telecommunications and healthcare. METEOR has been trialed[Mil96] at the Connecticut Healthcare Research and Education Foundation for modelling and enacting workflows for tracking childhood immunisations. METEOR's architectural contributions are focussed towards improving implementation in distributed environments and is strongly influenced by the OMG's CORBA technology.

METUFlow

METUFlow[Dog97] is a prototype implementation of a distributed W*f*MS that utilises distributed scheduling, monitoring and worklist managers, together with a distributed transaction service. Concurrency control is supported via the identification of "spheres of isolation" within a given process schema.

CPOL

Care Plan On-Line[War99] was a joint initiative between the University of South Australia, Medical Communication Associates and the South Australian Department of Human Services, aimed at providing clinicians with a flexible care planning tool, which allowed for the capturing of patients' problems and goals, capturing of data to support clinical decision making, and scheduling of tasks to support care plan enactment. Although not strictly a W*f*MS , since it provided passive functionality only, CPOL provided much of the functionality required for supporting the enactment of care plans for chronic illnesses, and was extensively tested in the South Australian Co-ordinated Care Trials, known as SA Health-

Plus. In particular, care plans could be cloned from *base care plans*, based on clinical guidelines for a specific condition, and tailored to an individual patient's needs by adding and/or removing tasks.

ADEPT

ADEPT[Rei98] is a research project aimed at developing flexible workflow management design architectures. Since its commencement in 1996, the project has produced several prototype implementations that have been used in healthcare, including being used as a basis for **Hematowork**, described below. ADEPT's key design aspects are:-

- the specification and management of temporal constraints such as "complete activity X two days before starting activity Y". Such temporal constraints are common in healthcare process protocols.
- support for exception handling and ad-hoc schema changes, by the use of formal pre- and post-conditions for change operators.
- workflow schema evolution with change propagation to running instances.
- the specification and implementation of inter-workflow dependencies in distributed environments. This is achieved through interaction expressions to support synchronisation of concurrent cooperating workflows.
- scalability and load balancing via migration of running instances and their state information between servers.

HematoWork

HematoWork is an application of a WfMS to oncology processes, developed by the University of Leipzig, and based on theoretical work of Müller *et al* [Mü99]

and the prototype W*f*MS AgentWork. The primary focus of AgentWork is an ability to support the semi-automated handling of control flow failures, using logic-based Event-Condition-Action (ECA) rules. AgentWork, and particularly its theoretical underpinnings, is extensively and impressively described in Robert Müller's PhD dissertation[Mü02].

2.4 Information Technology in Healthcare

This section looks at the broader role of information technology as it relates to health care.

Hau *et al* [Hau02] suggests "Three factors will greatly influence the further development of information processing in health care within the near future: the development of the population, medical advances, and advances in informatics." These factors led the authors to establish 30 theses for health care provision as it might be enacted in the year 2013. The theses covered areas of health care, such as its people, its information systems, and its ICT tools. Three major goals of information technology were identified:

- patient-centered recording and use of medical data for cooperative care;
- process-integrated decision support through current medical knowledge; and
- comprehensive use of patient data for research and health care reporting.

The burgeoning political impetus in some countries to drive the adoption of information technology into health care, particularly to support the first two of the above-mentioned goals, namely electronic health records and integrated decision support, is as astounding as it is unprecedented. The UK's National Health Service (NHS) has committed well in excess of $US5 billion per annum[9], for the

[9]NHS Information Authority, October 2004

foreseeable future on infrastructure for this purpose. The USA has embarked on major new projects to deliver an electronic health record for most Americans within 10 years[10].

Let us briefly examine the evolving information technology landscape of healthcare under the broad headings of "Clinical Information Systems", "Electronic Decision Support Systems" and "Electronic Health Records", and the role W*f*MSs might play in their deployment.

2.4.1 Clinical Information Systems

Sittig *et al* [Sit02] suggest that "Clinical information systems (CIS) could drive progress in health care into the 21st century". They describe a Clinical Information System as "a collection of various information technology applications that provides a centralized repository of information related to patient care across distributed locations". They further cite functionality that CISs can provide, as allowing clinicians to:

- view patient-specific information in a legible, organized, and timely manner,
- access the medical literature,
- ask clinical or administrative questions of aggregates of patient data,
- receive automatic warnings or suggestions when the patient's data satisfy certain logical rules,
- receive critiques when proposing therapies or ordering diagnostic tests,
- access guidelines for standards of care,

[10] address by President Bush, Maryland, April 2004

- analyze tradeoffs and the likelihood of alternative outcomes (decision analysis), and
- receive lists of differential diagnoses.

In Thomas Beale's Health Information Standards Manifesto[Bea01], Beale concludes that future CIS's need to overcome the following:

- Domain size and rate of change,
- Systems obsolescence,

and provide:

- Support for a life-long health record,
- True interoperability among all parties and systems used in patient care,
- Intelligent decision support,
- Support for a multi-contact healthcare system including mobile patients,
- Support for multiple medical cultures, including developing world - asian,
- Support for domain experts to have direct control over the information design and change management of their systems.

From the above, it is obvious that WfMSs of the kind proposed in this thesis, need to be an integral component of effective Clinical Information Systems. A broad perspective on the role of clinical knowledge, patient data and workflow processes being applied to improving individual patient health care is illustrated in fig.2.1

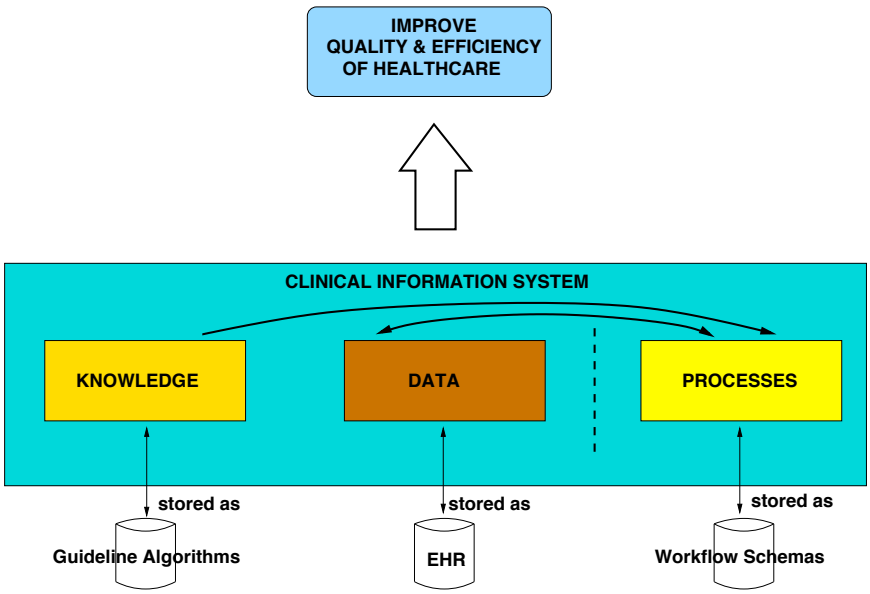


Figure 2.1: *Interplay of knowledge, data and process to improve healthcare.*

2.4.2 Electronic Decision Support and Workflow

Clinical Decision Support Systems (DSSs) are "active knowledge systems which use two or more items of patient data to generate case-specific advice" [Wya91]. Apart from their specific object of improving patient care, they are designed to do so by increasing "the uptake of research evidence in clinical practice". Clinical Decision Support is also often referred to as Electronic Decision Support (EDS)

Cesnik[Ces02] suggests that in discussing EDS it is "useful to view it as occurring at four levels:

- **Level 1** - Provides base level categorised information that requires further processing and analysis by users before a decision could be arrived at.
- **Level 2** - Presents clinicians with trends of patients' changing clinical status and alerts clinicians to out of range assessment results and intervention strategies. Clinicians are prompted to review information related to the alerts before arriving at a clinical decision.

- **Level 3** - Uses deductive inference engines to operate on some knowledge base and automatically generates diagnostic or intervention recommendations based on changing patient clinical condition, with the knowledge and inference engines stored in the knowledge base.

- **Level 4** - Uses more complex knowledge management and inference models such as case-base reasoning, neural networks, or statistical discrimination analysis to perform outcome or prognostic predictions. Such systems possess self-learning capabilities and use fuzzy set formalism and similarity measures or confidence level computation as mechanisms to deal with uncertainty intelligently and accurately.

Levels 1 and 2 EDS are data/information driven while Levels 3 and 4 systems are knowledge driven." Thus, a primary aspect of all DSSs is the inherent requirement for clinical information or knowledge to be represented and manipulated. This differs from classical WfMSs , where such knowledge is decoupled from direct representation, and where only small amounts of clinical data are supported by the software.

2.4.3 Electronic Health Records and Workflow

For inter-health-provider care plans to be successful, health providers need access to the right information at the right time, at the point of care. Shared Electronic Health Records (EHRs) can play a fundamental role in providing the correct data to support care activities. The international Health Level Seven's[HL7] initiative to base healthcare communication on the object-oriented Reference Information Model (RIM), recognised the importance of linking data with process, by incorporating a Unified Service-Action Model (USAM) into the RIM, whereby all data in electronic messages and EHRs can be associated with specific health

care activities (ACTs), and these ACTs can be related to each other through ACT-RELATIONSHIPs. Schadow *et al* [Sch01] showed how these actions and patient data could further be linked to clinical guidelines, to produce more effective decision-support systems.

Barretto *et al* [Bar03; Bar04] propose extensions to the *open*EHR [11] EHR modelling constructs INSTRUCTION *Archetypes*[Bea03], which provide a mechanism to support workflow schemas and record workflow state in individuals' EHRs.

The Danish National Board of Health has developed a conceptual model [Asp03] for EHR architectures based on a Process Model which associates and provides for collection and dissemination of data based on, and appropriate to generic process steps of *diagnosis, planning, plan execution*, and *evaluation*.

Whilst each of the above enhanced EHR architectures provides mechanisms to link data, process and knowledge, in order to ensure that the right information is available for the right purpose, at the right time, they do not directly address the issues of dynamically adapting processes to meet changing patient circumstances. This is particularly true of the HL7 approach, which views a goal as a property of an ACT, rather than as an enabling mechanism to achieve a goal.

2.5 Summary

This chapter showed the potential contribution that W*f*MSs can make in ensuring that guideline-based, individualised care plans are actually followed and coordinated across care providers. It also highlighted the difficulties that traditional W*f*MSs face in being applied in a healthcare context. The healthcare-specific characteristics of multi-provider, instance-adaption, non-determinism, irreversi-

[11] The *open*EHR Foundation is an international endeavour to produce a standardised, flexible EHR architecture - http://www.openehr.org

bility, expectation-focus, complexity and incomplete specifications lead to the need for a paradigm shift in the functionality required of WfMSs . These broad characteristics of healthcare complexities will be used in chapter 4 to derive a comprehensive table of functional requirements for an ideal WfMS .

This chapter also examined how WfMSs support, interact and in some cases depend upon other components in the information technology arsenal encountered in the health domain, particularly Clinical Information Systems; Electronic Decision Support; and Electronic Health Records. The next chapter looks more broadly than healthcare, to explore current technologies related to workflow modelling and WfMSs .

CHAPTER 3

Current Workflow Related Technology

"If you wish to advance into the infinite, explore the finite in all directions."

Goethe, Miscellaneous Epigrams, 18xx.

3.1 Introduction

This chapter examines the various approaches to modelling and designing workflow systems across a broad spectrum of application domains. It also considers related technologies and approaches that have influenced the author's approach to meeting the complex needs of coordinated health care. In particular, the contributions made from the following related research areas are discussed:

- Petri Nets
- Computer-interpretable Clinical Guideline Systems;
- Goal-oriented Requirements Engineering;
- Active Object Databases;
- Database Schema modification;

- Flexible/Dynamic Workflow Approaches.
- Web Services Composition

3.2 Workflow Management Taxonomies

A number of comparisons of Workflow Management approaches have been documented, which categorise W*f*MSs according to different aspects or characteristics.

Han *et al* [Han98] categorise workflow management systems by their support for runtime adaptability. In so doing, they identify four distinct levels, based on decreasing levels of abstraction, namely domain, process, resource, and infrastructure. The authors then go on to identify modelling principles of "Flexible Composition and Dynamic Hierarchy of Workflow Models", "Systematic Management and Dynamic Binding of Workflow Resources", and "Local Decision Making and User Involvement". These are all very useful principles that can be applied, and indeed need to be applied in the domain of health care.

Bernauer *et al* [Ber03] identify nine different perspectives from which to view interorganisational workflow specifications, namely "functional, operational, behavioural, informational, organizational, causal, historical, transactional and interactions". However, the last of these, whilst covering some coordinating and managing aspects across organisations, leaves out dealing with interactions between tasks, which can occur both for intra-, as well as inter-organisational workflows.

3.3 Petri Nets

Probably the largest contribution to research into W*f*MSs has come from the discipline of process modelling, particularly that based on Petri Nets, as discussed extensively by van der Aalst [Aal98]. There is further discussion regarding the

issue of Petri nets, and their contributions to W*f*MSs and my approach in section 4.1.7 on process flow.

3.4 Computer-interpretable Clinical Guidelines

Clinical Practice Guidelines are *"systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific clinical circumstances"* [Fie90]. Many attempts have been made to develop computer-based systems to assist health providers in charting a given patient's course through a guideline. These attempts have resulted in computer-interpretable languages, models and tools such as Arden Syntax, Asbru, Dharma, GEM, GLIF, Prestige, PRODIGY, PROforma, EON. These languages concentrate on expressing process semantics, including temporal constraints, but generally fall short of providing full workflow capabilities in areas such as notification mechanisms, actors/role support, resource constraints. They differ from classical workflow-based systems in that they incorporate considerable domain knowledge, but little or no organisational knowledge.

3.5 Goal-oriented Process Modelling

Jacobs *et al* [Jac95] discuss the separation of business goals from the tasks undertaken to achieve those goals. Such an approach has also been adopted for modelling software engineering processes [Myl99].

Gross *et al* [Gro01] utilise a hierarchical goal model in the context of evolving system architectures, particularly agent-based systems that redefine alternative implementations.

Koubarakis *et al* [Kou99] view business process modelling through a set of sub-models: organisational; objectives and goals; process; concepts; constraints.

They introduce a formal framework based on these sub-models, and expressed in the concurrent logic programming language ConGolog [Gia00]. Whilst their framework covers some of the ideas discussed in this chapter, it does not approach modelling from the perspective of expressing goals in a dynamic workflow schema.

Clinical guidelines proponents [Sha98] have incorporated intentions into the Asbru language for representing care plans. These intentions are temporal-pattern constraints (e.g., a process intention to administer regular insulin twice a day; an outcome intention to maintain fasting blood glucose within a certain range over at least 5 days a week) that are allocated individual weights signifying their relative importance.

3.6 Active Object Databases

Active Object(-oriented) Database (AOD) technology is a mechanism for capturing both behaviour and structure of objects inside persistent stores. Just as behaviour can and often is added to relational databases using stored procedures and triggers, AODs use *production rules* stored within an object database to capture behaviour. Unlike relational databases and stored procedures, however, AODs associate and restrict an object's behaviour to the object itself. This parallels the difference between object-oriented languages and procedural languages.

Production rules for AODs are normally expressed as **E**vent, **C**ondition, **A**ction (ECA) rules. Kappel *et al* [Kap00] show how such AODs, enhanced with support for roles, can be used as a framework for implementing WfMSs .

3.7 Database Schema Evolution and Versioning

Since workflow process schema are often stored in databases, and subject to changes in a similar manner and under similar influences to database schemas,

the issues concerning database schema versioning are often applied to workflow schema versioning. According to Roddick [Rod95], "within the object-oriented paradigm, a common method [for schema evolution] is to establish a set of invariants to ensure the semantic integrity of the schema and a set of rules or primitives for effecting the schema changes, while within the relational model a set of atomic operations is proposed, which result in a consistent and, as far as possible, reversible database structure."

However, databases are used for a much broader range of applications than managing the enactment of business processes, and although the literature related to database schema evolution and versioning can teach us about implementation techniques, there are still some fundamental philosophical issues specific to workflow schema adaptability that need to be addressed. It is the insights gleaned from such workflow approaches described in the next section, coupled with those related to goal-oriented requirements engineering and process modelling described above, that have most influenced the author.

3.8 Flexible/Dynamic Workflow Approaches

There have been many contributions in recent years to the literature on Dynamic Workflow, in particular starting from the work of Ellis *et al* [Ell95]. Research since has broadly covered three main areas. These areas have followed the spread of W*f*MS from highly process-oriented environments, where change occurred occasionally to cater for new business processes, through domains where change occurred more rapidly, through to some of today's areas of application, including healthcare and e-Business, which not only require the ability to support adaptation at the instance level, but even expect runtime changes to be the norm in many circumstances.

3.8.1 Workflow Evolution

Following on from the work of Ellis, research into workflow evolution centred around the issue of constructing new workflow schemas from existing ones, such that the new schema is both derived in a formal way from the old, and is, in its own right, a validly executable schema. Casati *et al* [Cas96] introduce workflow schema evolution primitives under the categories *declaration primitives* - that modify workflow variables and tasks; and *flow primitives* - that modify the flow structure of the schema. Other approaches include that of evolution based on inheritance [Aal01]; and workflow transparency models [Bic97] which utilise a two tier model separating business processes from worklow. There has also been some discussion on implementation methodologies, usually based on metamodels, and sometimes referred to as Dynamic or Active Object Models [Man99].

3.8.2 Workflow Migration

As a result of changed business processes, and resulting workflow schema evolution, the issue of migration of running workflow instances is often encountered, particularly in long-living processes. As well as being addressed in much of the literature on workflow evolution [Joe99], it is also discussed in the context of interorganisational cooperative workflows [Aal99]. The workflow migration literature discusses the circumstances under which a running instance, created from one schema, should be migrated to another schema, and asks the question - can the new schema legitimately accommodate all states of the old schema?

One can postulate scenarios in chronic disease management, where new clinical practice may supercede existing practice. For chemotherapy treatment of cancer patients, for instance, subtle, or even radical changes in the type, dose, frequency, method of administration etc. could occur during the course of a prescribed treatment regime. Patients undergoing such treatment could be considered candidates

for migration to the new treatment process. However, it is unlikely that migration strategies would be developed at the level of transitioning between two workflow schemas. A more likely solution would be that a new workflow schema would be devised for the individual patient, based on his/her prior exposure, and current condition, and then enacted.

3.8.3 Flexible Workflow

Flexible workflow can be considered as workflow based on schemas whose control flow is incompletely specified at design time, and which are completed at runtime, from information acquired at runtime on an instance by instance basis. Klingemann [Kli00] extends traditional notions of workflow schemas to support Quality of Service (QoS) goals. These QoS goals are then used at runtime to influence decisions applied to *flexible elements* of the workflow schema. According to Klingemann, flexible elements can be of three types, namely: *Alternative Activities*, *Non-vital Activities*, or a group of activities with *Optional Execution Order*. *Alternative Activities* are considered to be different from traditional OR-SPLIT/OR-JOIN constructs. With an OR-SPLIT, the successor activity of the split is determined by the evaluation (by the workflow engine) of a predicate specified at design time, whereas a choice amongst Klingemann's *Alternative Activities* can be made (by a user) based on global goals of the workflow.

Sadiq *et al* [Sad01; Sad02] have postulated a mechanism for providing runtime flexibility in process enactment by identifying "pockets of flexibility" in a workflow schema at design time, and inserting a "build" task into the schema, that can tailor and finalise those tasks in the pocket to meet instance requirements. This is a very similar technique to developed by this author, described in chapter 6 of this thesis. Sadiq *et al* place considerable emphasis on schema correctness when both identifying any pocket of flexibility and also in constraining the operations

that are permitted at runtime. They build on coloured Petri Net based models to provide these validation constraints.

The methodology espoused in this dissertation goes even further, by coupling both the "alter task" (corresponding to Sadiq's "build" task), as well as the "change region" (corresponding to Sadiq's "pocket of flexibility") to an assessment and an achievement of a given goal. Where there are multiple goals, or subgoals, there will be multiple alter tasks and change regions.

3.9 Web Services Composition

The recent drive to utilise internet technologies to harness and integrate services from otherwise distinct and heterogeneous organisations has led to a proliferation of XML-based languages and evolving solutions. Two main application areas that have driven research towards this end are "e-Business service integration" in general, and collaborative scientific research using "the Grid". For the former, several authors [Car03; Maj04b] have analysed web service characteristics that allow for classification of services. Sirin *et al* [Sir03] illustrate how automatic workflow compositions can be created and orchestrated via BPEL4WS (sometimes abbreviated to BPEL i.e. Business Process Execution Language), supported by DAML-S (now OWL-S).

In the arena of Grid Computing, Majithia *et al* [Maj04a] have applied workflow technologies to the matching and coordination of computational tasks amongst host machines. Even health care workflow has been approached from the viewpoint of web services composition, with Lee *et al* [Lee04] emphasising the importance of *syntactic, semantic* and *contextual* constraint bases of knowledge as inputs for the composition of clinical services for *service flows*.

Morrison *et al* [Mor04] have gone further with this approach, using BPEL, to-

gether with an information model based on the General Practice Data Model and Core Data Set. The resulting "Modelling the Clinical Processes of Prescribing" (MCPOP) project aims to produce a system to support "on demand" service discovery and workflow orchestration, based on emerging (health) industry standards.

Mandell *et al* [Man03] have argued for a richer level of semantic knowledge to be brought to bear for Web Services Composition through languages such as OWL-S (formerly DAML-S), which supports the representation of each service through a *Service Profile* - a description based on the service's inputs, outputs, preconditions and effects.

3.10 Summary

This chapter examined a set of different approaches to modelling and designing workflow systems in various application domains. It also considered related technologies and approaches that have influenced the author's approach to meeting the needs of coordinated health care. In particular, the contributions made from the related research areas of computer-interpretable guideline systems; goal-oriented requirements engineering; active object databases; Petri-net-based process modelling; database schema modification; workflow migration; and web services composition were discussed.

Part II

Theoretical Solutions

CHAPTER 4

Overview of Approach

"It is easier to say new things than to reconcile those which have already been said."

Vauvenargues, Reflections and Maxims, 1746.

This chapter introduces and explains a number of concepts important for the design and implementation of the author's workflow enhancements. This is followed by an analysis of the requirements for workflow-enhanced systems to be successful in the particular healthcare context of chronic disease management. An attempt to address some of these challenges - the two-tier, two-phased approach adopted by the author is then described, whereby a design phase and execution phase are used to delineate structural from behavioural aspects of WfMSs . The clinical context and associated workflow for Diabetes Management, which is used throughout this thesis in examples, is also described.

4.1 Concepts

Let us first analyse a generic workflow that represents the diagnosis and treatment of a condition as might occur in many healthcare settings. Fig. 4.1 illustrates how such a workflow is composed of a set of generic activities (examine, assess, plan,

treat) that recur until the patient is deemed "well", or, in the case of many chronic diseases, the patient dies.

The first point to make about this workflow is that it is hierarchical. Almost always, the *examine*, *plan*, *implement/treat* activities will be compound activities that can be broken down into more clearly defined tasks. The second point of note, is that there are distinct phases of the workflow at a high level, particularly those of "diagnosis", firstly, and "treatment" as a distinct 2nd phase. In chronic disease management, the diagnosis phase, if diagnosis is used in its strictest form as a matching of the patient's symptoms with those of a known condition, then this occurs early on and forms only a small component of the overall workflow. However, the need for ongoing assessment of a treatment effectiveness is a similar concept, that becomes a significant aspect of chronic disease management.

4.1.1 Goals and Processes

In complex chronic conditions such as diabetes, there is frequently a set of goals that are defined in Clinical Practice Guidelines for a "generic patient". These goals are then appraised by one or more actors (healthcare service providers) and modified for an individual patient. A set of tasks then needs to be instantiated to try to achieve the goal-set. For each goal, the set of tasks must include a task to assess the effectiveness of the treatment regime (i.e. has the goal been met) and one to adjust the treatment regime, or the goal(s) accordingly. The set of tasks required to achieve a goal is termed a process. Processes may be able to be decomposed into subprocesses. Several processes could be combined into a superprocess. Thus, the goal view and the process view are inextricably linked. In many domains, business goals are decomposed into tasks. These tasks when enacted, automatically result in the goal being achieved. In healthcare however, tasks very often require goal parameters to be modified dynamically, alterna-

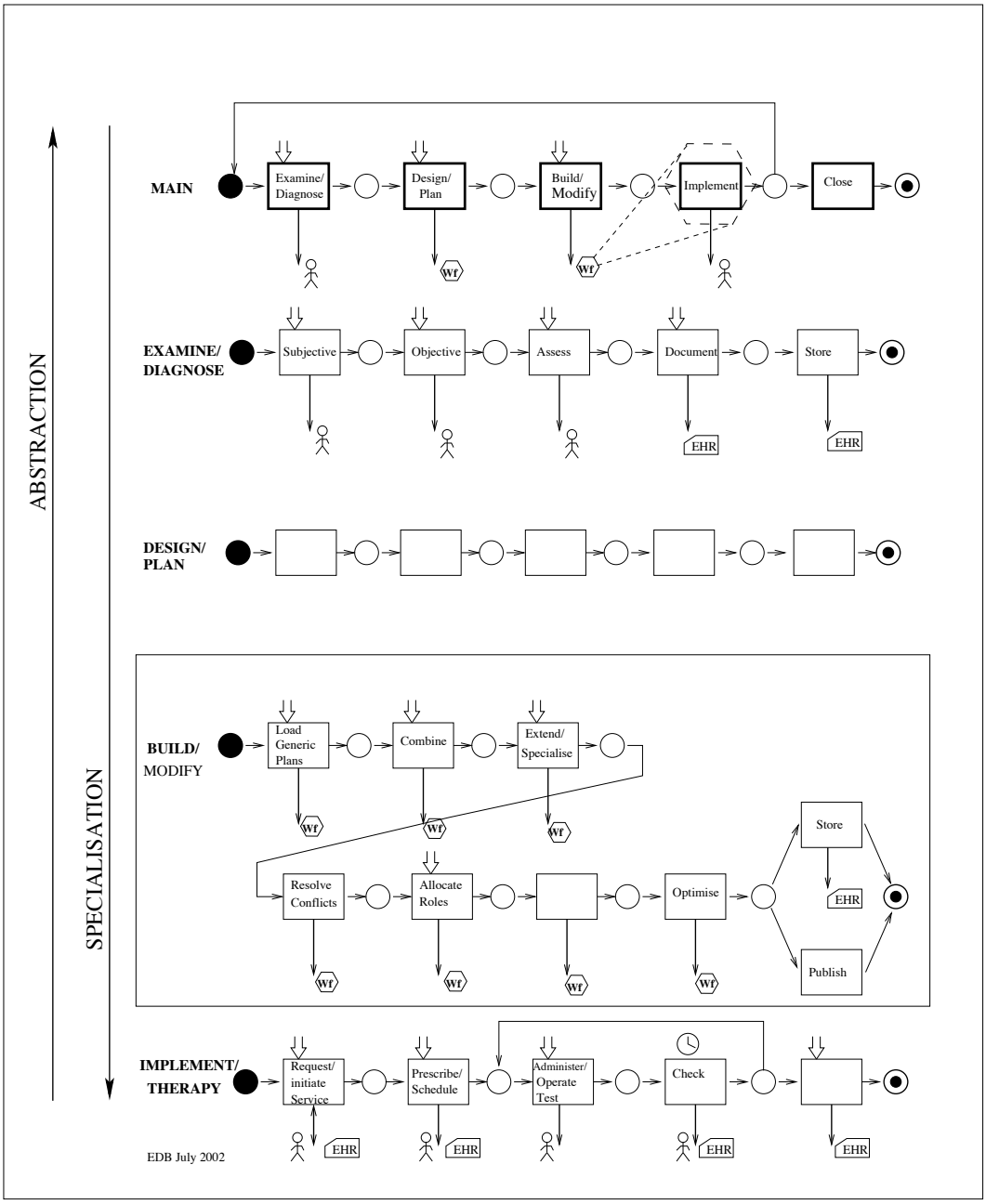


Figure 4.1: *Generic Healthcare Workflow*

tive (even competing) goals to be selected, or a runtime decision on the most appropriate method of achieving goals to be made. The principle of hierarchical decomposition, and the strong coupling between goals and their achievement through appropriate tasks at runtime, is fundamental to the approach taken by the author throughout this dissertation.

A process description or schema can be reused for many patients. Once a workflow process has commenced, it is referred to as a *process instance* or *case*.

4.1.2 The Process Metamodel

Figure 4.2 illustrates the principal concepts used to describe a generic workflow process and its enactment for a given case. This process metamodel aligns well with that used by the W*f*MC and is a simplified version of that utlised by the author's protype implementation described in chapter 8. The concept metamodel is expressed as a UML[1] class diagram. The metamodel concepts are described in the following sections.

4.1.3 Tasks and Activities

In general, there is no need to distinguish between tasks and activities and much of the literature concerning process modelling makes little or no distinction between the two concepts. There are however, some circumstances where distinctions are made. An *activity*, in some workflow literature, refers to an instantiated task, i.e. a task belonging to a specific case, being enacted (or having been enacted) by a specific resource. The Workflow Management Coalition refers almost exclusively to activities. The OMG's[2] UML provides *Activity Diagrams* [Obj03] for modelling processes, wherein the steps that constitute a process are either *Action States* or

[1] Unified Modelling Language
[2] Object Management Group

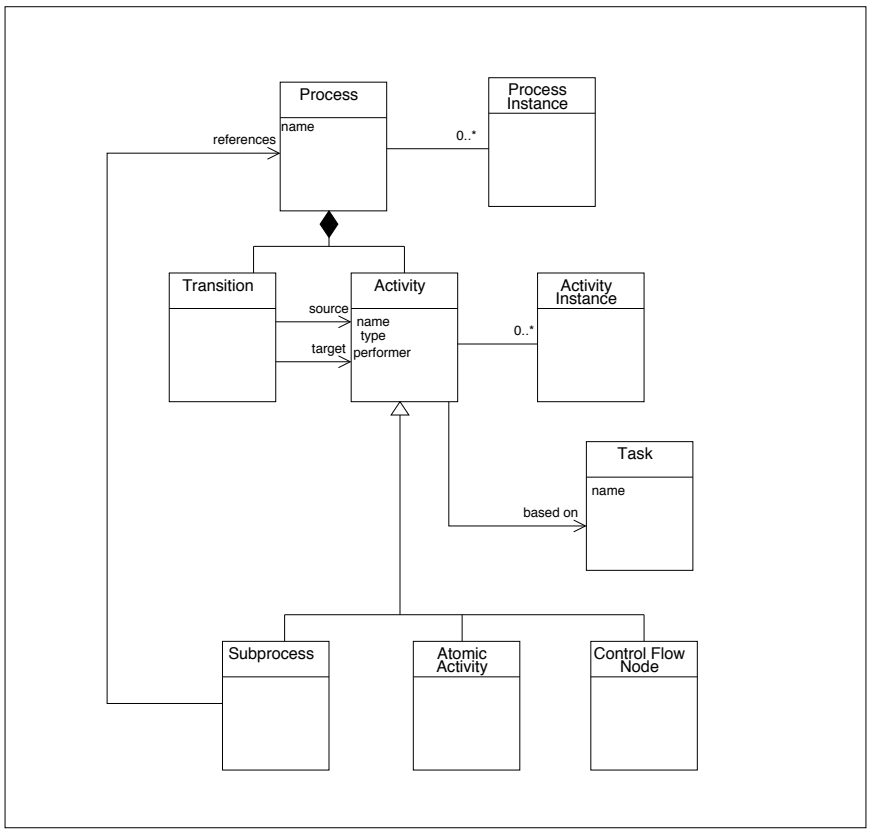


Figure 4.2: *Generic process metamodel*

SubActivity States. Their *Action State* describes an atomic process step, whereas a *SubActivity State* describes a compound process step that comprises one or more action steps. The use of the label *State* by the OMG is problematic for workflow discussions because it conflicts with the Petri Net notion of state embodied in *places* (see sections 4.1.6, 4.1.7 and 4.1.8) below.

In the implementation discussions later in this thesis, task libraries are introduced. These libraries of *tasks* can exist independent of any particular process schema. Tasks are performed on one or more objects (in the healthcare domain the primary object is a patient) and each task may be passed data as *input pa-*

rameters and may pass data to other tasks as *output parameters*. Tasks can have generic input and output parameters. When a task is chosen to be placed in a specific workflow schema, additional input and output parameters and pre- and post- conditions may be applied. In the context of a specific process schema, tasks take on the characteristics of activities, where the differentiating characteristics amongst activities are:-

- their position in the process schema
- a unique id
- the control and data flows into and out of the activity.
- the resources allocated to the activity.

At the same time, activities tend to defocus attention from the nature of the work being undertaken. It is not the traditional role of W*f*MSs to concern themselves with how an activity is performed, or with the quality of work performed.

Tasks may be atomic - i.e. not decomposable into subtasks, or they may be composite. A composite task, when instantiated, is normally instantiated as a subworkflow - i.e. a separate workflow instance exists for the duration of the composite task instance. Task types can be considered to include routing constructs such as OR-SPLITs, OR-JOINs, AND-SPLITS, and AND-JOINS; placeholder tasks such as START, STOP; and timing elements such as DELAYs. In some workflow schema meta-models, these routing constructs are considered as separate concepts, and also referred to as control-flow blocks.

4.1.4 Hierarchical Decomposition

Hierarchical decomposition offers advantages both for workflow modelling, as well as advantages at execution time. For the former, simpler high level models cor-

responding to patient and provider goals, allow for better communication and understanding between patient and provider, and when multiple providers are involved, a higher level in which to communicate major objectives of shared care. These higher level models are more stable over the long periods encountered with chronic disease management, and more uniform across patients. They can, and often are, represented by condition-specific, evidence-based guidelines. By representing the activities required to achieve each high level goal as a subworkflow in the overall workflow schema, the lower level details are encapsulated in separate subprocesses. Each of these subprocesses can then be further decomposed, taking into account current clinical practice and local organisational, environmental and patient constraints, until an appropriate overall workflow is articulated.

4.1.5 Late Binding

The detail of *how* parts of higher level models should be implemented a) are more likely to vary from patient to patient; b) are often best left to the provider involved in a specific aspect of the patient's care; and c) may be best left incompletely specified until the time that specific part of the workflow is to be undertaken. Hierarchical decomposition is often associated at implementation time, with *late binding*. Commonly adopted in flexible software architectures, *late binding* means waiting until required, in an application's execution, before finalising or chosing a particular component, process or data structure implementation. In modelling workflows, this translates to identifying high level constructs that are not likely to change, whilst employing abstract class descriptions for lower-level (*i.e.* more detailed) implementations or for areas of the workflow schema that are not required early on in the life-cycle of a given case. Late binding implementations provide a mechanism for *concreting* the abstract or incomplete components at runtime. W*f*MSs that support late binding require careful handling of activity

input/output parameters and pre- and post-conditions.

4.1.6 State

One of the major contributions of Petri Net modelling to workflow modelling, if not to the design of W*f*MSs , is the notion of state. Petri Net models are often applied to closed finite state systems, wherein the model represents a process composed of *states* (or places) and *transitions*[3] between those states. Workflow models are often represented as Coloured Petri Nets, or Place-Transitions Nets, in which coloured tokens represent a specific workflow case, residing, for the most part, in some state or other and flowing through the net from the *start* place to the *end* place. An analogy with object-oriented modelling further implies that a workflow comprised of a set of activities corresponds to a "class" with a set of "methods"; a case or an instance of a workflow corresponds to an "object"; the state of a workflow corresponds to the values of class "attributes"; the flow net corresponds to an "object life-cycle"; and the ability to individualise a workflow could be implemented through notions of "inheritance".

These representations, when applied to healthcare, can be somewhat simplistic. In simpler domains than healthcare, a case might correspond to a physical product, such as the production of an aircraft. Sometimes the case corresponds to a business service such as an insurance claim, travel booking or document processing. However, in the healthcare context, there are often a number of objects, or targets of a given activity or set of activities that need to be considered. At the highest level, we are concerned with providing a healthcare service. The particular service is often the major objective of the workflow schema. In this sense, we have two similar process targets:- the completion of the service, and the improved health of the patient that corresponds to this service case. The success

[3]see section 4.1.8 for discussion on the conflicting definitions of *transition*

of the former is often measured by the quality of service provided, independent of the patient outcome. The latter is measured by a change of state of the patient.

However, some parts of the workflow schema may act on secondary targets such as a patient specimen; an aspect of resources (*e.g.* an operating theatre); an aspect of patient data; or an aspect of the environment. An activity could be undertaken which teaches a carer how to care for a diabetic foot. Here the target might be regarded as a resource, or even part of the environment, since the carer may not be a participant in the usual sense of an initiator of a modelled activity, but merely the recipient of an activity undertaken by another service provider. Because of the flexibility required in the healthcare domain, we even introduce activities that act on the current workflow schema itself, in order to adapt the schema for changed conditions. A corresponding set of state parameters can be introduced to represent the state of these different targets at any one time. These are:-

- *workflow state*, S_W
- *patient state*, S_P
- *environment state*, S_E
- *resource state*, S_R

Workflow state S_W is further comprised of two components,

- *control state*, S_{Wc}, representing the workflow behaviour, and
- *data state*, S_{Wd}

The reason for explicitly defining *data state*, S_{Wd} is to clearly differentiate information that is a snapshot of real, continually changing, physical information from the physical information itself. For instance, a patient's blood pressure

continually changes with time, whereas recorded blood pressure is a discrete set of zero or more values taken at specific instances of time. Theoretically, we can measure a patient's blood pressure at any future time t, but we may only have access to a small set of readings from the past, and may be unable to determine what the blood pressure was at any given moment in the past. Thus, a workflow data variable can be represented by a set of tuples, where any one $s \in S_{Wd}$ can be expressed as $s = (n, d, a, v_s, v_e)$, where n is the name of the state variable, d is the data value, a is the workflow activity that acquired or set this data value, v_s is the time at which data value became valid, and v_e is the time for which the data value is no longer valid. Thus $v_e - v_s$ represents the validity duration or time-to-live of the data value. Some temporal databases [Sno85] only capture the start validity timestamp, assuming that updates to the data variable will automatically define the end of the previous validity time. In healthcare, we often have data being contributed from different sources whereby it is possible to have overlapping validity timestamps entering a shared repository. Many temporal databases also capture transaction time. In the healthcare context, we assume that all data is implicitly timestamped with its transaction time as would normally occur in patient Electronic Health Record (EHR) repositories. Some EHR-based systems may capture additional metadata regarding each data variable, such as the accuracy; the clinician's confidence in the value; a reference to the clinician who was responsible for this value of the data for a given activity.

Thus an activity target object O can be any one element of the set $\{C, P, W, E, R\}$ where C is the workflow case, P is the patient, W is the workflow schema, E is the environment, R is the set of resources involved in the case.

Healthcare activities are often categorised into *Investigations/Observations*, *Evaluations/Assessments* and *Interventions/Treatments*. From our definitions of state above, we can say that Investigations measure patient state (S_P) in order to

change workflow data state (S_{Wd}); Assessments change S_{Wd}, often by temporal, spacial and other algorithms applied to prior values from S_{Wd}; Interventions are intended to change patient state (S_P).

4.1.7 Process Flow

Another area where Petri nets complement workflow modelling is in the very notion of a "flow net", whereby the tasks to be undertaken are prescribed *a priori*, and the topology of transitions and places dictate the order and relative timing of tasks from start to end. But apart from the unsuitability of pre-ordained tasks, task sequencing and pre-ordained states to that of healthcare, there is a side-effect of Petri net modelling that skews workflow modellers and researchers towards thinking in terms of flow nets at the expense of other ways of viewing processes. Specifically, it downplays the importance of the complexity of state, described above, and gives undue weight to workflow state at the expense of environment, resource and patient state. The invocation of a given task is thought to depend, primarily, on the state of the net, *i.e.* which prerequisite tasks have been completed. In reality, there are often a set of preconditions for the invocation of a task, only some of which are that the appropriate prior tasks have been completed. Indeed, the author would argue, that for many healthcare scenarios, there are other preconditions that are of equal or greater significance.

This is an area where the contributions from the field of Active Object-oriented Databases (see section 3.6) present an interesting perspective. In AODs, there is no real notion of flow. Activities are invoked when a trigger event is received, provided the set of activity preconditions are met. A trigger event could be the completion of a prerequisite task. It could equally well be some other event, such as one related to resources, external events, or elapsed or prescribed time. If we consider a process as a set of tasks, each of which can execute subject to certain

conditions, then we obtain a model such as the one shown in fig. 4.3.

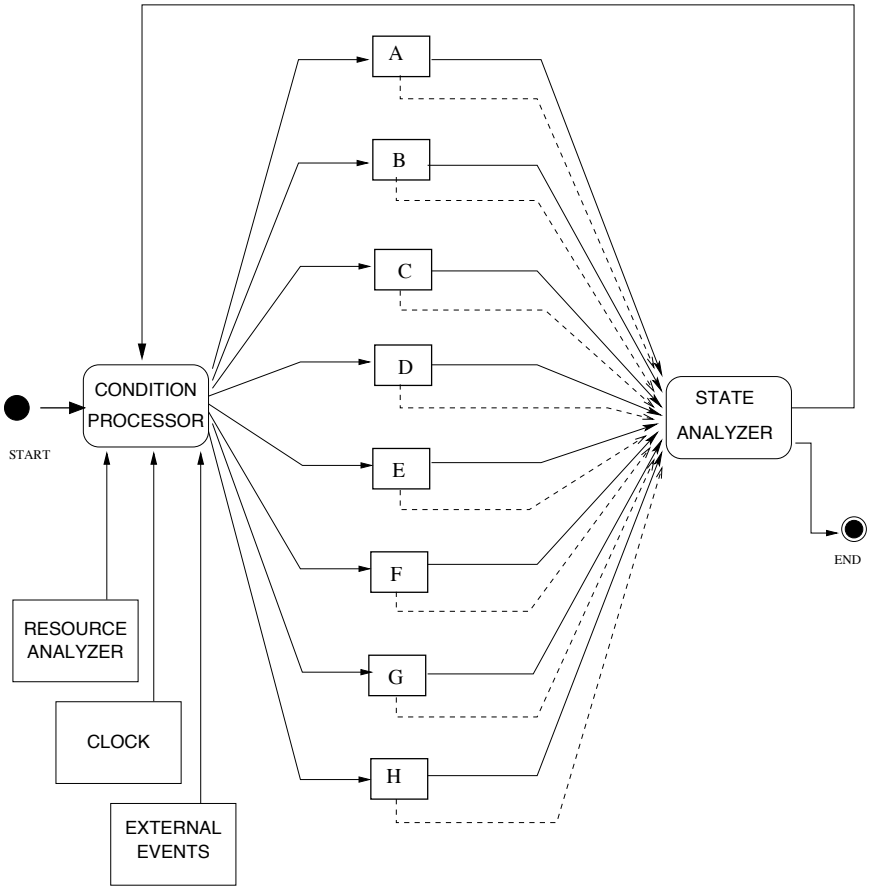


Figure 4.3: *Generic process model with 8 tasks.*

In a workflow engine, the activation of each task is under the control of a condition processor, which determines when a task may commence, be suspended/resumed, or be aborted, depending upon the state of other tasks, together with temporal, resource and external conditions. All or any of tasks A-H can execute, concurrently or sequentially, depending on prevailing conditions.

If one compares the two perspectives, that of Petri nets on the one hand, and that of AODs on the other, there are important observations that can be made.

4.1.8 Activity as Transition vs Transition between activities

In Petri Net modelling, the term *transition* refers to a transition of the 'system' from one state to another. A *transition* occurs, or is said to 'fire', when each of its input *places* has at least one token. In workflow modelling parlance, such a transition corresponds to an activity, and the input places correspond to preconditions of the activity. Simplisticly, these input places usually correspond to control-flow state, or control-flow preconditions of the activity. Some workflow literature, and indeed many W*f*MSs , differentiate between control-flow preconditions and data-flow preconditions, describing an activity as being *enabled* by control-flow preconditions being met, and the same activity being *ready* when all data preconditions have been met.

There is a competing notion of a *transition* prevalent in workflow literature and used extensively by the W*f*MC and by the developers and users of W*f*MSs , whereby a *transition* corresponds to the period between two activities. Throughout this thesis, in modelling and theoretical discussions, *transition* will usually correspond to an activity, as in most Petri Net interpretations of workflow. In implementation discussions, such as in Part III, a *transition* can generally be assumed to represent the control flow between two adjacent activities.

4.2 Functional Requirements

In approaching the problem of workflow-enabling chronic disease management, it is useful to articulate a set of functional requirements that address the issues raised above, and in chapters 2 and 3. Table 4.1 brings together a set of characteristics required of future workflow-based clinical information systems, before they can be considered, in the author's opinion, to provide adequate functionality to

improve chronic disease management.

Table 4.1: Health care workflow requirements

	Requirement	**Explanation**
1	Clinical Guideline Support	Allows for computer-interpreted clinical guidelines to form basis of workflow schema for a patient
2	Goal-focused	Goals are explicitly represented with targets, in activities (minimum).
3	Inter-organisational	Support for distributed execution of a process instance
4	Dynamic role assignment	Users can take on a role after process instance has started. ⇒ late binding of users to roles.
5	Dynamic task selection	Specific tasks can be chosen for parts of a process after the process instance has started. Implies late binding of concrete tasks to abstract placeholders.
6	Activity credits	Ability to alter schema instance to fully or partially subsume an activity into another of similar type.
7	Conflicting task resolution	Ability to alter schema instance to fully or partially resolve conflicts between tasks.

continued next page

Table 4.1: *continued*

	Requirement	**Explanation**
8	Organisational modelling	Ability to model an organisation (resource set) from orthogonal viewpoints e.g. facility/location/function.
9	Self-modifying	WfMS support operators that can change downstream workflow of running instances.
10	Self-managing	Schemas have specific tasks dedicated to support self-modifying w/fs.
11	Role-controlled change management	Schema alteration can be controlled by role-based users.
12	Process schema inheritance	New schemas can be created by inheriting from existing schemas.
13	Task Libraries	Schemas can be assembled from a library of tasks. Support for adding new tasks to the Library. Ideally, supported by an ontology of task types.
14	Schema Definition Language.	Schemas can be explicitly represented and passed between organisations using a parsable language (e.g. XML).
15	Explicit state representation	Workflow schema representations should provide for explicit workflow states between activities, as provided by P/T Net models.

continued next page

Table 4.1: *continued*

	Requirement	**Explanation**
16	Dynamic Process schema changes	Process schemas can be altered after process instantiation.
17	Subworkflows	An activity within a process instance can itself be a new process.
18	Predefined exception handling	Exception specification as part of a schema definition.
19	Ad-hoc exception handling	Exception handling on a per instance basis, under user control.
20	Role hierarchies	Roles can be defined as sub-roles of higher roles.
21	Preferred task prioritization	Tasks in an exclusive choice set, or an M of N choice set can have priorities assigned to them.
22	Instance-level user-role (re)assignment	Users can be assigned/reassigned to activities even after an activity has been started.
23	Dynamic schema evolution	New schemas can be saved as a result of a modification at the instance level.
24	Graphical Schema Representation	Tool to view schema. Tool to edit schema. Tool to modify running instances. Tool to monitor running instances.

continued next page

Table 4.1: *continued*

	Requirement	**Explanation**
25	Process state recording	State of process/activities should be recordable. Users should be able to interrogate case status at any time, to show what activities have occurred, which activities are currently active, which activities are scheduled/likely to occur.
26	Audit Trail	Support for logging of any or all instances - degree of logging should be configurable. Should include instance identifiers meaningful to users; start/end of activity execution; predefined parameters of process and individual activities.
27	Notification support	Explicit notification activities and mechanisms - *e.g.* email, fax, RPC/webservice style e-notification.
28	Priority escalation - manual and automatic	Individual instances should be able to have priority of process and/or specific activities raised. e.g. if patient state deteriorates, or delays occur. Should have the ability to set thresholds on parameters/time so that priorities can be escalated automatically.
29	Instance-settable activity pre/post-conditions	Pre-conditions and post-conditions of activities should be individually settable at the process instance level.

continued next page

Table 4.1: *continued*

	Requirement	**Explanation**
30	Break-glass emergency support	In the event of a malfunction of the system, or patient emergency, any user should be able to interrupt a running instance, undertake a change to the prescribed w/f and later inform the W*f*MS of the change.
31	Activity attribute accumulation	System should provide a stack of input/output activity parameters which can be viewed during an active process instance. This could be a cumulating view of all parameters associated with activities of this process, hithertofor completed.
32	Inter-organisational task transfer/coordination	Support for an activity to be passed from one organisation/facility/location to another for instantiation or completion. This may involve transfer between separate W*f*MS s.
33	Quality of Service goals	Support for specifying QoS parameters at the goal level. These would often be expected to translate to activity postconditions.

4.3 Design Approach

The author's design approach hinges around a two-tier model that separates, for a given business process, the high-level goal view from the lower level process view. A W*f*MS that supports modelling business goals as a goal hierarchy, can then use this goal hierarchy to generate a skeleton workflow schema, which includes the

special explicit tasks required to modify the downstream workflow at runtime. The template workflow schema can then be elaborated with specific tasks where these are sufficiently standardised to apply to all patients affected by the condition being modelled. Areas where individualisation may be required can be left as abstract subworkflows, which can be made concrete at runtime.

Other facilities incorporated into the design approach are task libraries, based on an overall health care task ontology, that characterise each task with sufficient information to support identification of potential overlap in functionality when tasks are incorporated into the overall workflow.

4.4 Execution Approach

The author's execution approach is centred around subworkflows and their accompanying *assess* and *alter* tasks, which help to ensure that, for a given patient condition and current state, those tasks that were incompletely specified in the initial schema will be fully specified at the appropriate time in the running workflow instance.

4.5 Example from Case-Study

Throughout this thesis, examples from the management of Non Insulin Dependent Diabetes Mellitus (NIDDM), commonly referred to as "adult onset diabetes", "type II diabetes", and often simply "diabetes", will be used. Clinical practice guidelines for diabetes management often include specific goal targets. The current targets specified in Australian guidelines [Dia04] compiled jointly by Diabetes Australia and the Royal Australian College of General Practioners are:-

- $\mathrm{HbA}_{1c} < 7\%$

- Blood Pressure better than 130/80 mm Hg
- Body Mass Index < 25
- Fasting Blood Glucose Level 4 to 6 mmol/litre
- LDL Cholesterol < 2.5mmol/L
- Total Cholesterol < 4.0mmol/L
- HDL Cholesterol > 1.0mmol/L
- moderate exercise of 20mins per day
- urinary albumen excretion $> 20\mu$g per collection.
- complete cessation of smoking

NIDDM was chosen for a case study, not only because of the important morbidity statistics of diabetes, but also because its treatment varies considerably from patient to patient; treatment is well documented by clinical guidelines; its management involves a number of health care provider types (GP, renal specialist, endocrinologist, podiatrist, dietician); goal targets are well established; may often be associated with other illnesses such as cardiac conditions and depression, and thus subject to considerable individual variation. However, the methodology could also be applied to the management of other forms of chronic disease such as Chronic Obstructive Pulmonary Disease (COPD), cancer, or some cardiovascular conditions.

4.6 Summary

This chapter outlined a set of requirements for WfMSs to provide effective process management capabilities for health care. The author's approach to meeting an

important subset of these requirements, namely a focus on goals; support for individualisation and runtime flexibility; and support for crediting of overlapping goals and activities was outlined. The following three chapters detail solutions to each of these requirements in turn.

CHAPTER 5

Goal-focussed Modelling

"Not every end is a goal. The end of a melody is not its goal; but nontheless, if the melody had not reached its end it would not have reached its goal either."

Nietzsche, The Wanderer and His Schadow, 1880.

5.1 Introduction

Healthcare is an area where traditional workflow management systems (W*f*MSs) are rarely found. Several reasons have contributed to this, despite the potential quality and efficiency benefits that W*f*MSs might promise. Firstly, patient state is not necessarily related to predefined tasks or events. This means that traditional object life-cycle models by themselves, are inadequate. Traditional (Petri-net based) models of object life-cycles assume that a state results from a transition (activity). As mentioned in chapter 2, in healthcare we are rarely dealing with causal systems. It may be true for some tasks, but not necessarily for others, *e.g.* activities which are measurements are deterministic: activity *take_Xray* results in state *Xray_taken* , but activities that invoke actions often are not: activity *lower_BP* (i.e. taking an action such as prescribing antihypertensive medications to lower blood pressure[BP]), may not result in state *BP_lowered*.

In order to utilise W*f*MSs to overcome this dilemma, we can abstract therapeutic goals to a higher level, and use one or more workflow models to try to achieve the higher level goals. This chapter explains how a two-tier model, based on high-level business-goals, implemented by lower-level workflow processes can achieve this. If, for a particular patient, a workflow has been enacted and found wanting, the physician might abort the workflow and instantiate a different one. This approach restricts the component tasks to deterministic ones, for which we can utilise Petri-net based models. Thus, in the previous example, *lower_BP* would be abstracted out of the workflow (treated as a goal), and replaced by specific tasks such as *administer_BP-lowering_drug* or *prescribe_exercise*, or a more complex subworkflow.

Patient care plans are about intent rather than result. Activities are undertaken in the expectation or hope that a given outcome will ensue [Mik97], and repeated assessments of the outcome will often need to be undertaken. These assessments should be built into the workflow schema.

Whilst not a panacea for all of the above challenges, self-managing workflows that link the business-goal view with the process-view, can assist in addressing those process issues where runtime changes are required. This chapter examines how such a two-tier model can help with the complex and dynamic workflows encountered in the management of a chronic medical condition such as diabetes.

5.2 Concepts

The management of complex chronic conditions is often described in terms of a set of goals that are appropriate for various examples of generic patients. When formalised in clinical practice guidelines, each goal is normally ascribed a target value, appropriate for the generic patient's current health status. For instance,

management of a Type II diabetic patient might suggest aiming for a target resting arterial blood pressure of 135/80 mm Hg, by the use of antihypertensive medication. This, and other goals specified in a clinical guideline are then appraised by one or more actors (healthcare service providers) and modified to match the state of a specific individual patient. A set of tasks then needs to be identified and enacted to try to achieve the goal-set. For each goal, the set of tasks must include a task to assess the effectiveness of the treatment regime (i.e. has the goal been met) and one to adjust the treatment regime, or the goal(s) accordingly. The set of tasks required to achieve a goal is termed a *process*.

Processes may be able to be decomposed into subprocesses. Several processes could be combined into a superprocess. Thus, the goal view and the process view are linked through tasks that **assess**, **alter** and undertake to **achieve** the corresponding goal targets. In many domains, business goals are decomposed into tasks. These tasks, when enacted, automatically result in the goal being achieved. In healthcare, tasks very often require goal parameters to be modified dynamically, or alternative goals to be selected. This section revisits the goal and process views introduced in chapter 4.

5.2.1 Two Tiers

In this chapter, we look at both the goal and process views in detail, but more importantly, examine the relationship between the two views, in order to design a workflow model that will optimize the achievement of the goal set. To this end, we introduce two separate spaces - the Goal Space **G** and the Process Space **P**. The author's view of the Goal Space is motivated by the work of Klingemann [Kli00], Jacobs *et al* [Jac95], Mylopoulos *et al* [Myl99] and Dardenne *et al* [Dar93], who, amongst others, have formalised the discipline of goal-oriented requirements engineering, leading to methodologies such as KAOS and i^*.

A somewhat similar approach is taken by the authors of the Asbru clinical guideline language, [Mik97] who couch goals in terms of plan *intentions*. However, intentions can be lower level concepts than goals. One can "intend" to administer a medication, but the administration is really an activity undertaken to achieve a higher goal.

Each activity in a workflow schema can be associated with the achievement of one or more high level goals. By decomposing goals into a goal hierarchy whereby the root goal corresponds to the overall objective of the health care for the patient, one can often identify and enunciate a range of ever-more specialised goals, culminating in goals which can be implemented by known specific best-practice activities.

5.2.2 Goals

For a given condition/treatment regime the goal set **G** will comprise the set $\{\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, ...\}$, where each $\mathcal{G}_i$ will have a relationship to one or more other $\mathcal{G}_k$. These relationships form a hierarchical directed acyclic graph, whereby children are related to parents either as *IsPartOf* or *IsA* relationships. In order to satisfy a goal, all its children of the former type must be satisfied, and hence they constitute *AND-refinements*, whereas children of the latter type constitute *OR-refinements*. There is one root goal (the overall treatment or management of the condition) $\mathcal{G}_0$ from which all other goals are descendants. We call this root goal the *Objective*. The satisfaction of all *AND-refinement* siblings causes satisfaction of the siblings' parent. The satisfaction of any *OR-refinement* sibling causes satisfaction of the sibling's parent. Thus only leaf nodes need to be considered as having associated tasks. In general, *IsPartOf* sub-goals need to be satisfied for all instances, whereas *IsA* sub-goals will usually be chosen for specific instances.

Goals have a number of properties. Firstly, goals have a set of zero or more

parameters that can express boundary conditions on goal achievement. Secondly, each goal should specify a validity time. In the care of chronic conditions, goals may take weeks, months, and sometimes even years to achieve. If several sub-goals all need to be achieved, the achievement of one sub-goal may need to be reassessed if sibling sub-goals take substantially longer to achieve.

Goals have been neglected in many quarters of health informatics. Some notable exceptions are the areas of terminologies [Bak02] and patient care data sets for shared care [Ozb97; HL7]. The HL7 patient care Technical Committee has established the following goal parameters for use in share care messages:-

1. Action Code
2. Action Date/Time
3. Goal ID
4. Goal Instance ID
5. Episode of Care ID
6. Master Goal List Number
7. Goal Established Date/Time
8. Expected Goal Achieved Date/Time
9. Goal Classification
10. Goal Management Discipline
11. Current Goal Review Status
12. Current Goal Review Date/Time
13. Next Goal Review Date/Time

14. Previous Goal Review Date/Time
15. Goal Review Interval
16. Goal Evaluation
17. Goal Evaluation Comment
18. Goal Life Cycle Status
19. Goal Life Cycle Status Date/Time
20. Goal Target Type
21. Goal Target Name

Whilst these HL7 goal parameters are not specifically designed to support any particular workflow modelling paradigm, they are indicative of an appreciation within the care plan messaging fraternity that goals should be incorporated into care plans. At least they should be capable of being articulated and communicated in a uniform manner between care providers.

5.2.3 Processes

For each goal, we can ascribe a process aimed at achieving the goal. A process is a set of tasks, that when executed in a certain sequence, by specific resources (actors, roles etc.), subject to certain constraints/conditions, should achieve the process goal. Whilst the goal view has no knowledge of the environment, the process view introduces one or more possible solutions, subject to the business rules, domain knowledge, organisational structure and environment. In traditional production domains, the process view is known as a workflow schema. In the health domain, this workflow schema must adapt not only to each patient (i.e. at

the instantiation of each case), but also must adapt during the running of each instance. Such dynamic changes are discussed in section 5.6.

Additionally, any W*f*MS used in a healthcare setting will need to provide for ad-hoc exception handling of unforeseen events. Whilst this is a critical requirement for W*f*MSs, it is becoming a standard feature of many commercial systems, and does not materially impact on the two tier modelling philosophy introduced here.

5.3 Goal View

For any health treatment regime, we may have a set of goals to achieve. These goals may be organised by decomposition into a *IsPartOf* / *IsA* hierarchy and represented graphically by an directed acyclic graph. Figure 5.1 illustrates such a goal graph for the generic management of diabetes patients. In this diagram **B.P.** denotes "Blood Pressure", **BMI** denotes "Body Mass Index" and **HbA1c** is an indirect measure of glucose levels in the patient's blood.

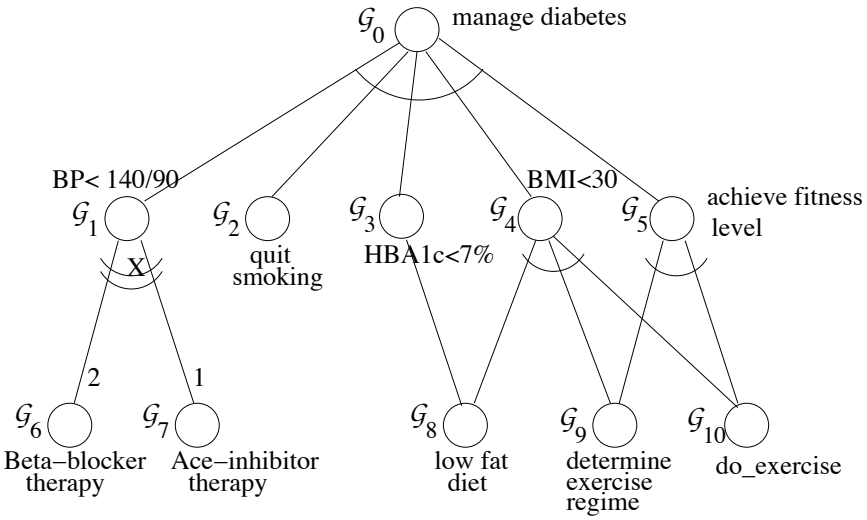


Figure 5.1: *Goal-graph for Diabetes Management (simplified).*

Sibling *IsPartOf* children form a set of goals, all of which must be achieved to satisfy the parent goal. These are depicted graphically with an arc crossing the edges connecting parent to children. Sub-goals which are *IsA* children are considered alternative choices to achieving the parent goal. These are depicted graphically with a double arc crossing the edges connecting parent to children. Alternative choice goals can either be inclusive OR (one or more could be attempted), as shown in fig. 5.2(a), or mutually exclusive, as shown in fig. 5.2(b). The latter have their double arc marked with an X. We can also nominate preferential goals in an alternative choice set, by labelling the edges in descending order as depicted in fig. 5.2(b) (1 denotes highest priority).

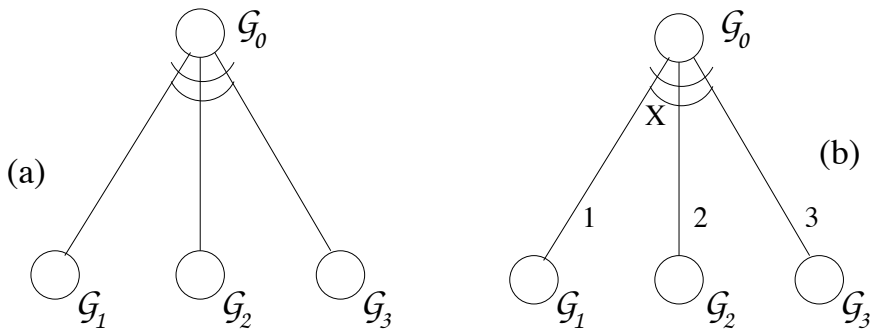


Figure 5.2: *Inclusive(a) and prioritized, mutually-exclusive(b) sub-goals.*

The simplified goal-graph for Diabetes Management, depicted in fig. 5.1, also illustrates another phenomenon that is often encountered in the health domain - that of goal-crediting. i.e. a child goal can satisfy more than one parent. This is isolated explicitly in fig. 5.3 and is discussed in much detail in chapter 7.

In this case, the satisfaction of sub-goal $\mathcal{G}_3$ is sufficient to satisfy both $\mathcal{G}_1$ and $\mathcal{G}_2$. In our diabetes example, the goals *determine_exercise_regime* and *do_exercise* contribute to both the parent goals *Body_Mass_Index*$<$ 30 and *achieve_fitness_level*, and so, there is no need to subject the patient to two programs of exercise.

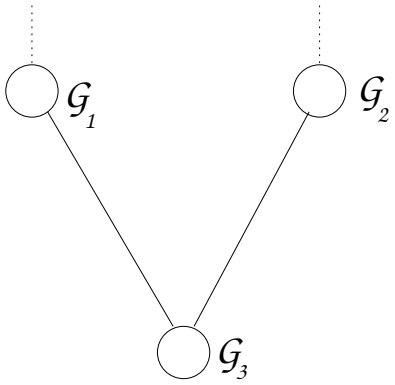


Figure 5.3: *Goal-crediting of common sub-goal.*

5.4 Goal to Task Mapping

In moving from the goal view to the process view, we are adding domain, organizational and environment knowledge in order to elucidate the explicit set of tasks, resources, conditions and task flow that needs to be applied. Every process $\mathcal{P}$, designed to achieve a goal $\mathcal{G}$, will have a set of tasks T, an associated set of resources R, achievement conditions G, and a goal validity time V. i.e. $\mathcal{P}$ is described by the 4-tuple (G, R, T, V). G expresses the boundary conditions on the goal, such as the achievement level, achievement tolerance, goal preference, goal priority, etc.

We can define three kinds of tasks:-

- T_α is a task whose function is to do or perform some action in order to *achieve* the goal,
- T_δ is a task whose function is to *assess* if the goal has been reached, and if not, to determine how different the current state is from the desired state,
- T_ν is a task whose function is to *choose* what alternative subgoals, or supplementary or altered tasks are required to achieve the goal.

T_ν and T_δ tasks together can be thought of as *supervisor* or *management* tasks. T_ν will be handed the goal's boundary conditions, G. T_δ will return the validity time, V.

We can identify two sets of goals, *leaf goals*, which are the leaf nodes of our goal graph, and *inner goals*, which represent all non-leaf goals.

5.4.1 Leaf goals

Leaf goals are goals that have no children. They map directly to processes that perform actions. The processes to which they map, are *expected* to achieve the goal, but may require some modification in order to do so. Let process $\mathcal{P}_A$ comprise the set of tasks $T_A = \{T_{A1}, T_{A2}, T_{A3}, ..T_{An}\}$, such that $\mathcal{P}_A$ is designed to achieve a leaf goal $\mathcal{G}_A$, i.e. $\mathcal{G}_A$ is some relation on $\{T_{A1}, T_{A2}, T_{A3}, ..T_{An}\}$.

We say process $\mathcal{P}_A$ *achieves* goal $\mathcal{G}_A$, denoted $\mathcal{P}_A \mapsto \mathcal{G}_A$ (see Fig. 5.4 [1]), if, for process $\mathcal{P}_A$, we can

(1) ascribe a minimum set $\mathcal{P}_{A\Psi} = \{T_{A\alpha}, T_{A\delta}, T_{A\nu}\}$,

and

(2) task $T_{A\delta}$ assesses that the current object's state meets its input goal-achievement condition, *i.e.* how well task $T_{A\alpha}$ has succeeded,

or

(3) we can identify a modified process $\mathcal{P}'_A$, resulting from the completion of task $T_{A\nu}$ which can achieve goal $\mathcal{G}_A$, i.e. $\mathcal{P}'_A \mapsto \mathcal{G}_A$. If process $\mathcal{P}_A$ can achieve goal $\mathcal{G}_A$

[1] S_{A_0}, S_{A_1}, ... represent workflow states between tasks.

without any modification by task $T_{A\nu}$, then $\mathcal{P}'_A \equiv \mathcal{P}_A$.

Only certain types of process modifications are allowed. One of the simplest and most desirable might be the selection of an alternative subprocess from a predefined set.

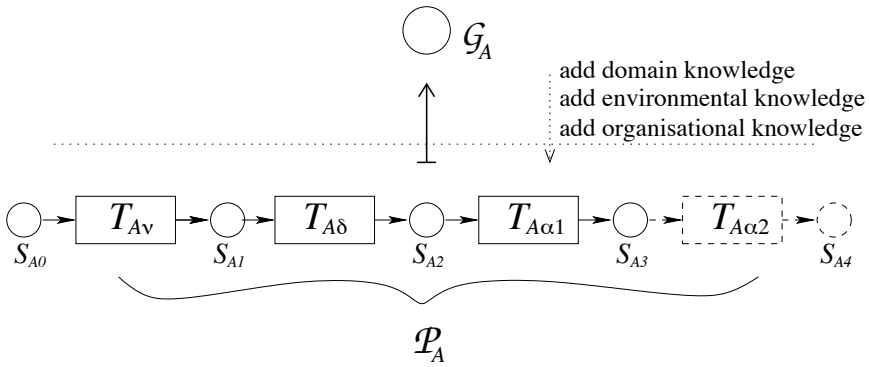


Figure 5.4: *Goal to process mapping*

In general, for any process $\mathcal{P}_\Psi$, there can be more than one T_α task, and task T_ν may be a null task if task T_δ has decided that the goal has been achieved.

5.4.2 Inner goals

Inner goals (i.e. non-leaf node) perform no actions, but do map to processes that evaluate whether the goal has been achieved, and if not, either change the goal boundary conditions, or choose an alternative goal set. Let process $\mathcal{P}_B$ be designed to achieve an inner goal $\mathcal{G}_B$.

We say process $\mathcal{P}_B$ achieves inner goal $\mathcal{G}_B$ ($\mathcal{P}_B \mapsto \mathcal{G}_B$) if we can

(1) ascribe a minimum set $\mathcal{P}_{B\Psi} = \{T_{B\delta}, T_{B\nu}\}$,
and

(2) task $T_{B\delta}$ assesses that the current object's state meets its input goal-achievement

condition

or

(3) we can identify some modified process $\mathcal{P}'_B$, resulting from the completion of task $T_{B\nu}$ can achieve goal $\mathcal{G}_B$, i.e. $\mathcal{P}'_B \mapsto \mathcal{G}_B$. If process $\mathcal{P}_B$ can achieve goal $\mathcal{G}_B$ without any modification by task $T_{B\nu}$, then $\mathcal{P}'_B \equiv \mathcal{P}_B$.

We can consider inner goals as abstract goals, in the sense that they rely on subgoals to implement actions. Thus, the processes for inner goals contain no T_α tasks, whilst those processes corresponding to leaf-goals perform real work.

5.4.3 Root goal

In order to satisfy the root goal ($\mathcal{G}_0$), we must descend each edge of the goal hierarchy to the leaf goals. At each level, we must satisfy all *IsPartOf* goals and at least one in each set of *IsA* goals. All of the leaf-node goals that need to be satisfied, will ultimately be satisfied by a set of action (T_α) tasks.

Unless there are constraints that mitigate against it, all processes should be implemented in parallel, in order to maximise the performance in achieving the top-level goal, $\mathcal{G}_0$. This provides a default mechanism for "joining" processes, and thus tasks, together to form a unified workflow schema.

5.5 Process View

In order to achieve goals, tasks may have pre-, post-, co-, inhibit-, force- and invariant-conditions, defined as:-

- *pre-condition* - a condition that must be met before a task can commence (i.e. necessary). Pre-conditions will often be expressed as conditions on input parameters of the task.

- *co-condition* - a condition which is false, prior to commencement and after completion of the task, but true for some time whilst the task is executing.

- *post-condition* - a condition which is true once the task has completed. A post-condition may typically be used to enable or inhibit another task.

- *inhibit-condition* - a condition which prevents the task from commencing. This is often used in relating one task to another, such as the second may be inhibited during some part of the execution of the first.

- *force-condition* - a condition which causes the task to commence, irrespective of any other conditions (i.e. sufficient).

- *invariant-condition* - a condition which is true before and after the task, but may be false for some or all of its execution.

Some of the above conditions may be conflicting, complementary or overlapping, and so would not all be present and utilised in a given system. They are introduced in order to illustrate how to define processes. Any specific domain or organisation may choose to utilise a subset.

Conditions can be functions on tasks, resources, external events or time. For example, a pre-condition may relate an input parameter of an assessment task to the process's associated goal, such as "Is $B.P. < 140/90$?". But in order to best illustrate sequencing of tasks, workflow schemas initially only consider those conditions that relate tasks to each other, and ignore temporal, resource and external constraints.

By considering only the conditions directly relating tasks to each other, we can formulate a representation of the task flow as a set of nodes (representing tasks) and edges (representing flow). For a given set of task-task conditions, we obtain a workflow schema, such as that represented in fig. 5.5. The workflow schema shown in fig. 5.5, corresponding to the goal-graph of fig. 5.1, illustrates the second level goals being implemented through subworkflows, indicated by rectangles with a double border. In this case, the T_α tasks to achieve, for example goal $B.P. < 140/90$, are implemented at runtime by instantiating a separate process (subworkflow), but the assessment of goal achievement is undertaken by a separate T_δ task. Task T_δ is complex, in that it resynchronises the preceding subworklows and determines if goal $\mathcal{G}_0$ has been achieved.

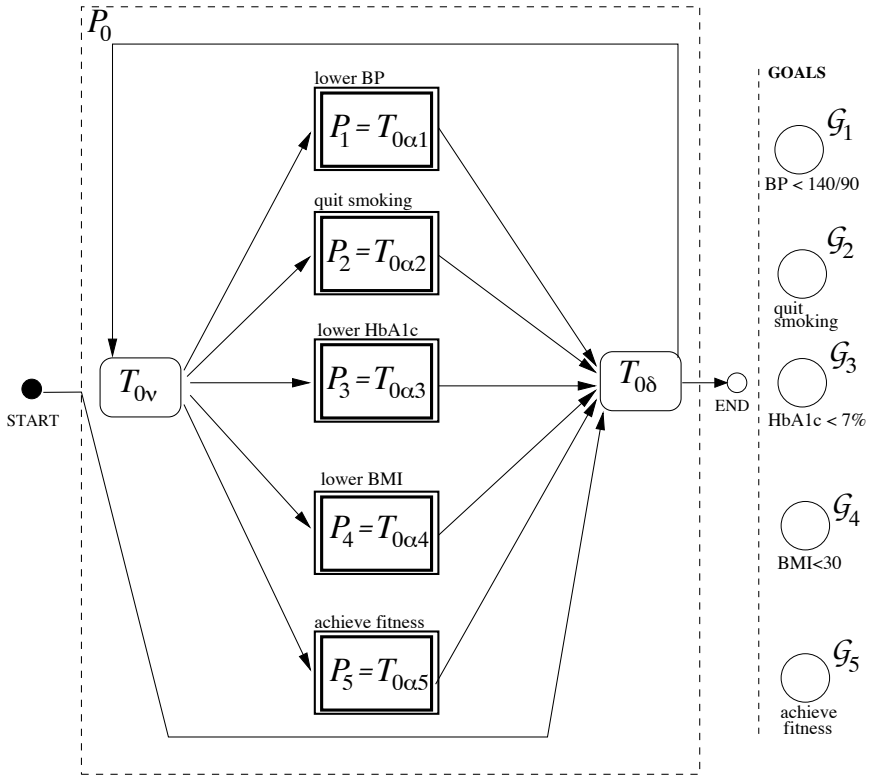


Figure 5.5: *Top-level diabetes workflow.*

Such workflow schemas can be represented by extended Petri-Net models [Aal98], such that the states between tasks are explicitly modelled, and such that additional (routing) tasks are introduced wherever the flow splits or resynchronises.

Fig. 5.6 shows the part of the workflow schema of fig. 5.5, expressed as a high-level Petri-Net, which implements process $\mathcal{P}_1$, corresponding to goal $B.P. < 140/90$. Routing task R_{11} is an OR-split, specifying which path to take, based on a pre-condition set by its precursor decision task $T_{1\nu}$ (remembering that goal $\mathcal{G}_1$ specified mutually exclusive sub-goals, with a preference for $\mathcal{G}_7$). R_{12} is an OR-join which re-merges the flow.

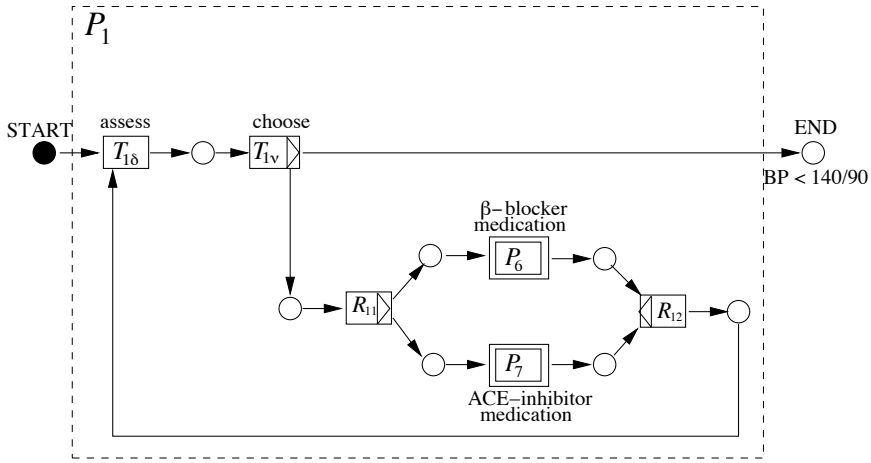


Figure 5.6: *Subworkflow for process 1 - as an extended Petri Net.*

Fig. 5.7 shows the ACE-inhibitor subworkflow ($\mathcal{P}_7$) of fig. 5.6, further decomposed into its constituent schema. Fig. 5.7 comprises soley atomic tasks. Assessment task $T_{7\delta}$ decides if and when the corresponding goal $\mathcal{G}_7$ "ACE-inhibitor therapy" has been achieved. If it has, then control is handed further up the chain to task $T_{1\delta}$ in order to determine if goal $\mathcal{G}_1$ ("is $B.P. < 140/90$") has been achieved.

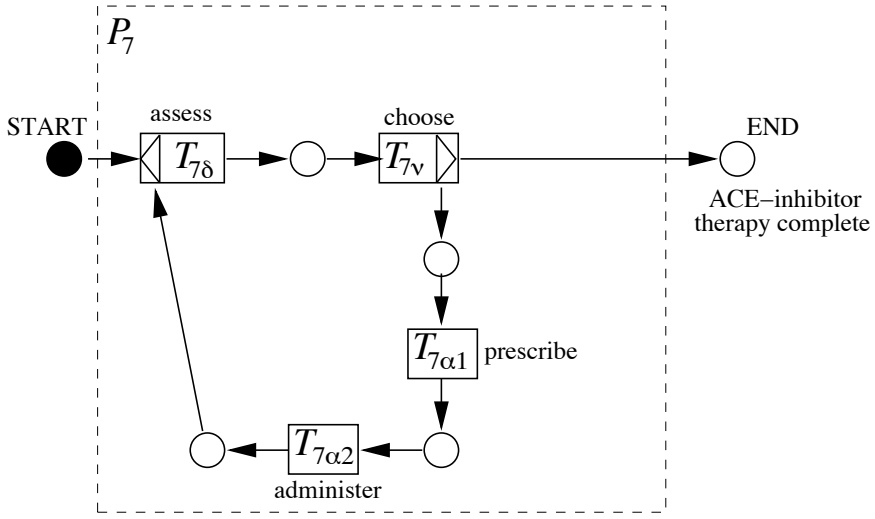


Figure 5.7: *Subworkflow schema for ACE-inhibitor process.*

Because of the tight coupling between goals and processes, we can think of the workflow engine as traversing the set of goals in the goal hierarchy, starting at the root, descending rapidly to the leaf-goals, initiating required tasks, and moving back up the hierarchy, checking for goal achievement as it goes. Similar methodologies are utilised by 3D graphics engines when rendering scene-graphs [Hes99].

At each level of the goal hierarchy, the validity time of descendent goals needs to be checked to ensure that completed goals are still valid and that their associated processes do not need to be repeated.

5.6 Issues

In order to achieve the total goal set, there may need to be changes introduced on a case by case (instance) basis. These changes can be broadly classified into three levels:-

Level 3. change workflow schema - the schema can be altered before the case is

instantiated. This in turn could comprise:

- introducing additional tasks.
- selecting from alternative tasks.
- aiming for alternative goals.

Level 2. change the workflow dynamically, whilst an individual case is executing.

Level 1. change the goal hierarchy itself

Future work is proposed which would develop models to address these levels of change in detail. The ultimate aim is to produce a comprehensive language to allow future Workflow Management System implementations to have business processes modelled at the goal level as well as the process level. This would be done through achievement constraints, separate goal assessment and choice/change tasks, and a set of rules governing the range of legitimate choice and change operations that are permited. Practical validation of these models is planned to be undertaken in conjunction with the South Australian Department of Health (formerly Human Services). Validation is intended to be undertaken, in the first instance, via the author's prototype workflow engine *Stream*Line , described in Chapter 8, to support the interorganisational requirements of chronic disease management within the Department.

5.7 Summary

In this chapter, a methodology was introduced to move from a business process view to an implementable dynamic workflow schema by encapsulating the business goal decomposition in the workflow schema itself. With this methodology, business process (re)engineering is no longer a separate process that occurs orthogonal to the business process activities themselves, but is integral with it.

This provides a flexible model that can adjust to changing goals, and can even do so on a case-by-case basis. In chronic disease management, this provides the potential for care management and care plan management to be brought under the umbrella of a workflow management system, and thereby reap the benefits of improved quality of care and increased efficiencies that such systems can provide.

CHAPTER 6

Workflow Schema Individualisation

"Care more for the individual patient than for the special features of the disease."

William Osler.

6.1 Introduction

This chapter examines aspects of modifications of a workflow schema required to meet the changing needs of an individual patient. The changes described are not merely limited to the specification of a static schema, individualised for a patient, and then instantiated. It also includes those changes required to support a running workflow instance for the patient, as the patient's circumstances, the environment, medical knowledge and the patient's health care organisational structures evolve.

These requirements for change, some of which were originally discussed in a general (non-health-specific) context by Ellis *et al* [Ell95], are listed below:-

1. extraneous state changes, both of the patient, and of the environment
2. new medical knowledge and practices
3. new environment and organisational settings

4. when workflow has complex alternative branches, it may not be worthwhile specifying the details of all branches prior to the case commencing

5. Workflows are long-lived (often for the remainder of a patient's life in the case of chronic conditions), and so circumstances are highly likely to change through the workflow,

6. whilst some areas of a workflow schema may be well defined at the case's onset, other areas may depend on too many variables, whose values are unknown at the outset.

7. there can be substantial overheads in attempting to model all potential exceptions for a complex workflow to manage chronic illness.

6.1.1 Contributions

The contributions of this chapter include:-

- the articulation of the requirement for specific workflow tasks to alter downstream workflow.
- the articulation of the types of workflow modification tasks, i.e. *build/modify* operators.
- the specification of downstream *change regions* that correspond to identified healthcare goals.
- the articulation of a set of primitives that constitute such change regions, and upon which the modification tasks operate.
- the illustration of an approach for the representation of chronic disease management care plans.

These contributions are aimed at improving the way W*f*MSs can enhance quality of care, particularly for the cooperative management of chronic illnesses. A side benefit of a goal-based perspective, is an enhanced ability to assess the effectiveness of specific workflows, since such assessment is built into the workflow process design itself. The rest of this chapter is organised under headings of related work; goal-based workflow; self-modifying workflows; controlling workflow changes; a healthcare case study; and finally, discussion and further work.

6.2 Goal-based Workflow

Fig 6.1 illustrates the context in which we can place W*f*MS with respect to goals and clinical guidelines. Starting with the top left quadrant, we have a simple statement of the goals to be achieved to manage a given condition, compared to the bottom right corner, where we have a workflow tailored to an individual patient to achieve the stated goals within a specific organisational setting. In moving from a generic set of static goals to an instance of a specific workflow, we are both shifting the emphasis from the static to the dynamic, and moving from a view dominated by clinical knowledge, to a view rich in organisational and patient-specific constraints. In so doing, however, it is important not to lose sight of the goals. In other words, the individualisation and runtime alteration of workflow schemas should remain linked to the goals.

The importance of goal-based workflow in healthcare has been discussed in Chapter 5 which described a methodology for deriving template workflow schemas from a goal hierarchy. Such a goal hierarchy can be used to produce a workflow schema comprising a set of nested subworkflows, where each subworkflow is aimed at achieving a corresponding goal in the goal hierarchy.

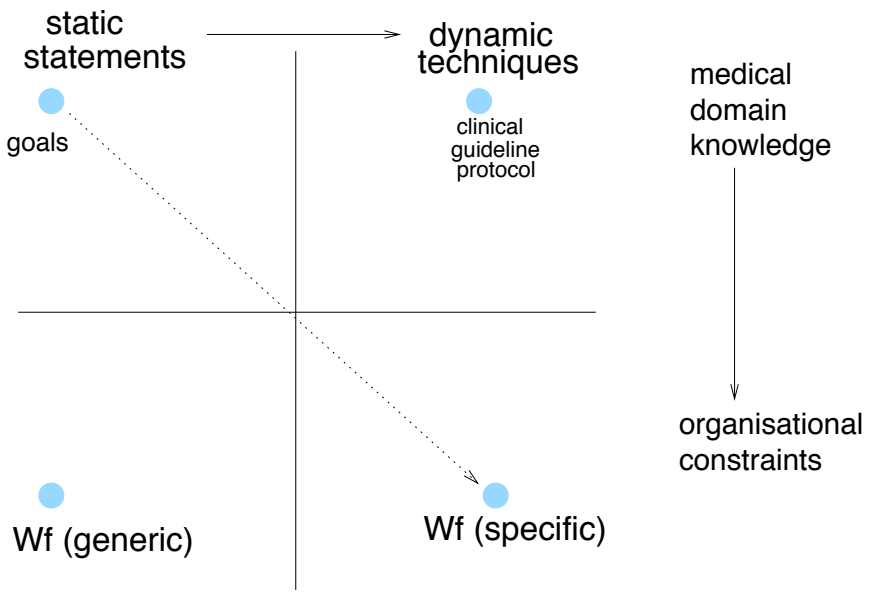


Figure 6.1: *Workflow in the healthcare context*

6.3 Approach

In order to assist the reader to understand the concepts introduced in this chapter, visual representations of the concepts are also introduced. The aim of these diagrams is primarily to assist the reader, rather than being intended as the basis of any formal specification of the concepts. However, these visual representations, wherever practicable, are in accord with and/or extend representations used elsewhere in workflow-related literature.

6.4 Self-Modifying Workflows

The issue of dynamic workflow schema evolution is often approached from the perspective of production workflows, wherein business rule changes dictate a new workflow schema, and whereby, consequentially, there is a class of running cases that might need "migration" to the new schema. Although such evolution oc-

curs sometimes in healthcare, such as when a new treatment regime is developed, there are other pressures to support schema changes. In any case, in healthcare, modifications to a workflow schema translate to modifications to downstream subworkflow, given the irreversible nature of most health care activities. Hence the emphasis is on modification operators, rather than migration issues. Moreover, in healthcare, there is often the requirement for workflow schemas to obey global, domain-specific constraints, which are independent of the flow constructs. Examples of these constraints include medication interactions and irreversible therapies.

The author's position is that any W*f*MS employed to manage chronic disease needs to provide explicit tasks which are dedicated to altering a set of downstream tasks. Self-modifying does not imply "automatic" self-modification. Instead, self-modifying refers to the inclusion of one or more modification tasks in the workflow schema, so that at run time, the modification task is guaranteed to be executed. Fig.6.2 illustrates the top level of a generic healthcare workflow. Each task at this level constitutes a process which is an abstraction of a set of tasks that are undertaken to fulfil the top level task. The fundamental processes that are illustrated here are common to the majority of chronic disease management workflows. These fundamental tasks/processes of patient assessment, evaluation and planning, building a care plan, and implementing a care plan are each implemented through subworkflows.

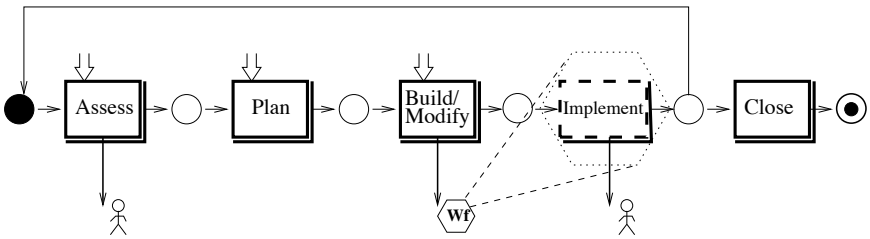


Figure 6.2: *Generic healthcare w/f utilising subworkflows*

The vertical arrows leaving each process in Fig. 6.2 denote the objects upon which each process operates. Thus the *Assess* process operates on the patient, whereas the *Build/Modify* process operates on the current workflow instance, and builds or modifies the downstream *Implementation* subworkflow. This constitutes the notion of self-modification.

It is the *Build/Modify* process which we concentrate on in this chapter. Such a process comprises a set of tasks, each of which takes as its input the following:-

a. the target task or *change region*.

b. a schema modification operator

c. task parameters

Each *Build/Modify* task is thus constrained to operate on a specific part or change region of the downstream workflow and may be further limited in the types of changes that it can undertake.

This chapter introduces a number of workflow schema modification operators, for which we provide an accompanying graphical representation. In order to achieve this we must first declare the schema primitives upon which our operators operate.

6.4.1 Workflow Schema primitives

van der Aalst *et al* [Aal00] describe a set of workflow patterns that they have identified when applying W*f*MSs to real world problems. Their patterns are grounded in an extended Place/Transition Nets (P/T nets) formalism. Here we introduce a number of primitives to support patterns often required in healthcare and we illustrate their use in a healthcare case study (Section 6.6). In most cases, these primitives are introduced to simplify the visual representation of otherwise more

complex P/T net representations. Unless otherwise stated, the models presented here are similarly based on extended P/T nets, whereby a task corresponds to a transition, and a workflow state corresponds to a place. A task becomes an activity when it is instantiated for a given case and resource/role. Each place is represented by a circle. Each task is represented by a rectangle. A task can be atomic or compound. There are two types of compound task - *subworkflow* and *aggregate*. Subworkflows result in the launching of a new instance and are represented by a rectangle with double border, as in Fig. 6.3.

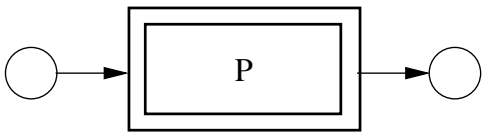


Figure 6.3: *Subworkflow*

A subworkflow inherits all the input parameters of the parent instance and returns all its output parameters to the parent on completion. Aggregate tasks are constructs representing a set of two or more tasks, all obeying the same flow rule, such that the aggregate task appears to other tasks as atomic, with defined entry and exit points. Aggregate tasks represent a convenient grouping of tasks to form a higher level of abstraction, and are akin to macro constructs in programming languages. Candidates for aggregate tasks are:-

- *Choice* - task that creates alternate execution paths, one of which is taken at runtime, based on the result of activity undertaken for this task. (Fig. 6.4)
- *Concurrent* - tasks are performed in parallel. Such tasks are otherwise represented as being preceded by an AND-split and resynchronised by an AND-join.
- *Exclusive* - a choice amongst tasks. Such tasks are otherwise represented as

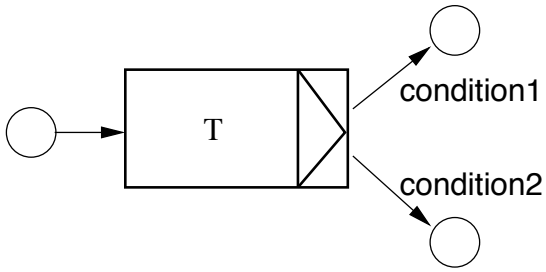


Figure 6.4: *Choice Task primitive*

being preceded by an XOR-split and remerged by an XOR-join.

- *Sequential* - a set of tasks to proceed in a strict sequential order.
- *Sequential Order-independent* - a set of tasks, which must execute sequentially, but whose order of execution is not specified.
- *M of N Concurrent* - a set of N tasks, of which any M must execute concurrently. When M have completed, the aggregate task is deemed to have completed.
- *Iterate* - repeat the task N times. When N have completed, the aggregate task is deemed to have completed. N can be replaced by an expression precondition.

Tasks can also be declared *abstract*. This means that one or more operators need to be applied to the abstract task to produce a concrete workflow schema. Tasks which are abstract, and therefore required to be replaced, are denoted as dotted rectangles, as in Fig. 6.5. Such tasks could represent either specific actions, or alternatively they could represent goal achievement tasks. Since the *Build/Modify* task of Fig. 6.2 requires such tasks to be concreted, any goal targets will, of necessity, be replaced by specific actions tasks.

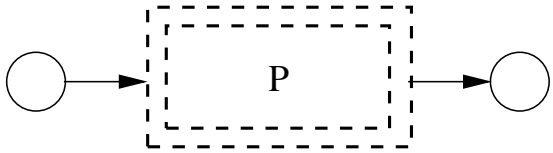


Figure 6.5: *Abstract Task - (in this case a Subworkflow)*

Pre and post conditions can be imposed upon tasks. This is normally the way users, roles or other resources are associated with a specific task. Other conditions can include workflow expressions, temporal or external events. These are depicted as symbols atop the task, as in Fig. 6.6. Task condition modifiers can also be declared abstract, and so will require concretion by appropriate operators. Abstract conditions are depicted as dotted symbols in lieu of solid ones.

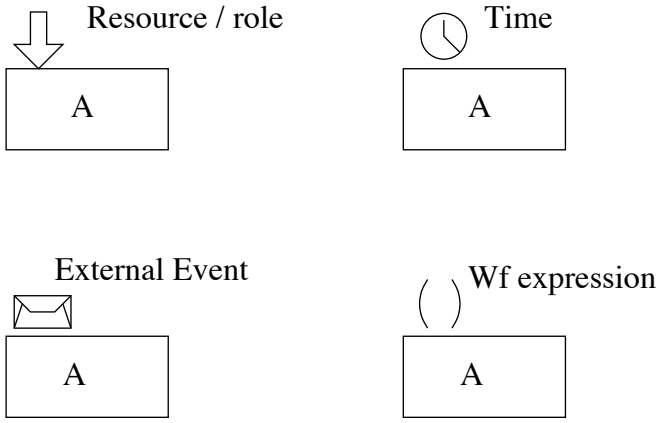


Figure 6.6: *Task Condition modifiers*

6.4.2 Schema Modification Operators

This section describes a set of operators for modifying an existing workflow schema. The operators are limited to dealing with tasks and their associated flow. In the majority of workflow implementations, there is a set of conditions, often expressed over input variables to a task, which qualify if and/or when a

task is undertaken. These constraints can be task, resource, role or event based. Currently, there are no agreed semantics to express these conditions for W*f*MSs. The reader should be aware that the set of operators introduced here is aimed at self-modifying workflows, and differs from the set of operators described elsewhere [Kra98; Joe99] for schema evolution and instance migration.

The principles upon which valid modification operations are based, firstly stem from the notion of deriving concrete processes from abstract ones. An abstract process schema for the *Implementation* subworkflow of Fig. 6.2 will normally be based on a Clinical Guideline or Care Plan template. The *Modify/Build* process will apply operators to this process to form a concrete subworkflow, which the workflow engine will subsequently enact. Parts of the Care Plan template may require *completion* - e.g. by the addition of one or more tasks specific to the patient's current needs. To support schema completion, we introduce the following operators:-

- task insertion operators
  - *add_alternative_task*: the action of adding an alternative task to an existing task or task set. This will also involve adding or modifying an associated OR-split and OR-join.
  - *add_parallel_task*: the action of adding a concurrent task to an existing task or task set. This will also necessitate adding or removing an associated AND-split and AND-join.
  - *insert_sequential_task*: the action of adding an additional task to an existing task sequence. The new task could become the first, last or some intermediate task in the sequence.
- task replacement operators

  - *substitute_task*: the action of replacing a task with a specific abstract or concrete task, such that the new task requires no changes to the input or output conditions. e.g. "administer atenolol" (a specific β-blocking agent) instead of "administer β-blocking agent".
  - *refine_task*: the action of replacing a task with a specific abstract or concrete task, such that the new task refines the input parameters to the task, or changes the implementation to improve the output conditions. Examples are replacing a generic "appendectomy" by a specialised "appendectomy for children" (refined input conditions) or replacing a "generic appendectomy" by a "laparoscopic appendectomy" (refined output conditions).
  - *extend_task*: the action of replacing a task with either a specific abstract or concrete task, such that extra output parameters are generated. e.g. replacing a "colonoscopy" task by "colonoscopy with biopsy".

- flow control operators
  - *limit_choice*: remove one or more tasks from an exclusive choice aggregate task. At least one component task from the set must remain.
  - *limit_concurrent*: remove one or more tasks from a concurrent aggregate task. At least one component task from the set must remain.
  - *impose_order*: specify one or more order constraints on a Sequential Order-independent aggregate task.

- flow constraining operators
  - *iterate_task*: apply an iteration to a task. The task will be repeated a specified number of times, or until a workflow condition is met. A

workflow condition is added to a task using the *add_expression* operator.

- *add_task*: add or insert a task into an existing workflow. The new task could be abstract or concrete.

- task constraining operators
  - *add_expression*: add a generalised workflow expression as an input condition to a task. Expressions can be based on workflow state, external events, external object state, temporal conditions, or conditions based on resources allocated to the task. Expressions can be declared sufficient (providing the task has been enabled by the flow, it will be started), or necessary (the task will be started only if all other conditions are also satisfied.) The following three operators are specific classes of this.
  - *add_resource*: add a specific resource/role to a task. i.e. declare that this task is to be undertaken by a specific class of person.
  - *add_event*: add a trigger event to a task. A trigger event could be a notification from another task, person, etc.
  - *add_time*: add an input timer to a task. i.e. the task can commence at a specific or relative time to when it was enabled.
  - *change_expression*: Modification operator. See *add_expression* above.
  - *change_resource*: Modification operator. See *add_resource* above.
  - *change_event*: Modification operator. See *add_event* above.
  - *change_time*: Modification operator. See *add_time* above.

6.4.3 Operator Specifics

The section discusses some of the above operations and associated issues in more detail. Only some of the above operators can be discussed within the bounds of this chapter, and are done so in order to give the reader better understanding of the scope of the modification operators.

Task Ordering

When the object of a workflow schema is an individual, such as a patient, it is common for tasks to be undertaken sequentially. Sometimes such a set of sequential tasks need to be undertaken in a specific order; other sets may be order-independent. Concretion operators need to be able to specify both of these scenarios. We show (Fig. 6.7) the corresponding operator *impose_order* concreting a Sequential, Order-independent task when it is declared abstract, and an order needs to be specified.

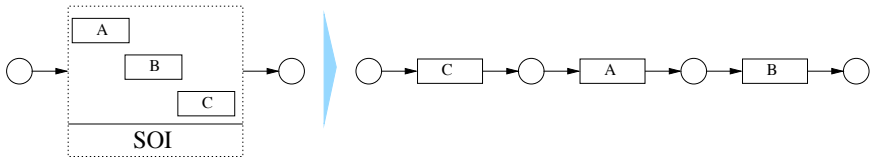


Figure 6.7: *Impose Order*

To help the reader better understand this operator, we illustrate in Figs. 6.8 and 6.9 how this might be represented using traditional P/T net notation. In Fig. 6.8 we have a 3 step Sequential Order-independent task.

In Fig. 6.9, we impose the constraint that tasks be executed in the strict order C then A, then B. Removal of redundant places reduces this net to the simpler one on the right hand side of Fig. 6.7.

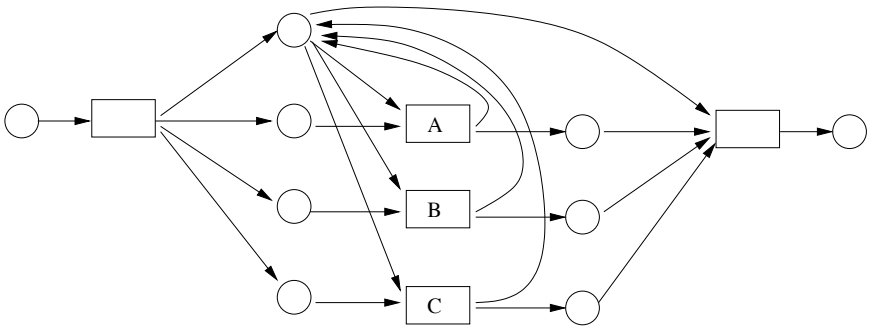


Figure 6.8: *3 Sequential Order-independent tasks (P/T net representation)*

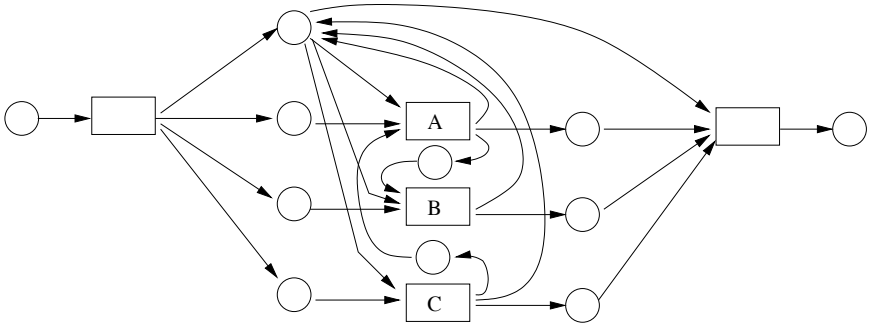


Figure 6.9: *Impose Order - C must precede A, which must precede B.*

Iteration

Another common schema tailoring operation is the replacement of a single invocation of a task with an iteration on the task, as shown in Fig. 6.10. A decision step is introduced to control under what conditions Task A will be undertaken, and how many iterations will be applied. Decision conditions could be determined a priori (i.e. at the time of subworkflow concretion), or the decision step could be assigned a role and so evaluated manually at runtime.

Prerequisites, Corequisites, Postrequisites

Another feature frequently encountered in healthcare workflow is the requirement to specify prerequisite, corequisite or postrequisite activity constraints. Often

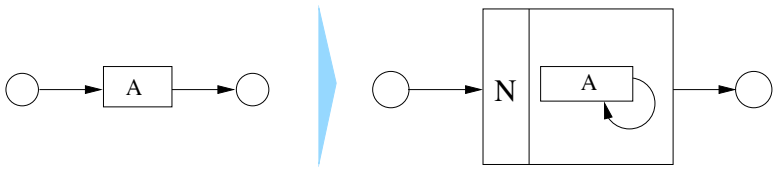


Figure 6.10: *Iterate a task*

these are global constraints, independent of the specifics of the flow. Given a task A, there may also need to be a task B, such that task B occurs somewhere in the workflow. If there is no requirement for temporal constraint between A and B, then A and B are termed corequisites. If A must precede B, then A is a prerequisite of B. If A must succeed B, then A is a postrequisite of B. Such constraints can be applied as task input conditions using the *add_expression* operator.

Resource assignment

Tasks requiring human intervention are commonplace in healthcare. An abstract workflow schema would be expected to contain a set of tasks which need roles to be assigned, in order for them to be considered complete. We denote such tasks by a rectangle with a wide dotted arrow atop, as in Fig. 6.11, and we can declare a completion operator *assign_resource* to be a required operator for such a task. If the workflow schema is to be later modified, then we would apply a *change_resource* operator. Sometimes a task could be assigned to an organisation, organisational unit or subsystem, rather than a person.

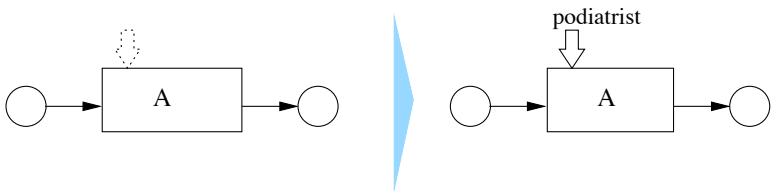


Figure 6.11: *Assign a role to a task*

Such resource assignments differ from traditional W*f*MSs whereby resources are normally assigned statically, as part of the workflow schema design process.

6.5 Controlling the Changes

In this section, we discuss the ways in which we can introduce greater control over the extent and scope of changes that can be made to a given workflow instance. Before so doing, it is worth reminding the reader of the importance of subworkflows.

6.5.1 The Use of SubWorkflows

Figure 6.12 shows a workflow schema illustrating how subworkflows are expanded to lower levels of abstraction (higher levels of specialisation).

By utilising a hierarchical structure (nesting) of workflows, we can obtain the following advantages:-

- Levels of abstraction to assist modelling the care process.
- Levels of specialisation that correspond to resources and roles.
- Processes (subworkflows) whose details can be altered dynamically at run-time.
- High level abstract tasks that correspond to goals, to be replaced by concrete tasks that achieve those goals.
- Mapping of abstract tasks to sets of preclassified low-level tasks.

The notion of hierarchical abstraction has been used in the Clinical Guideline modelling language, Arden Syntax [Ard02] to encapsulate clinical tests and actions into components called Medical Logic Modules (MLMs). MLMs can

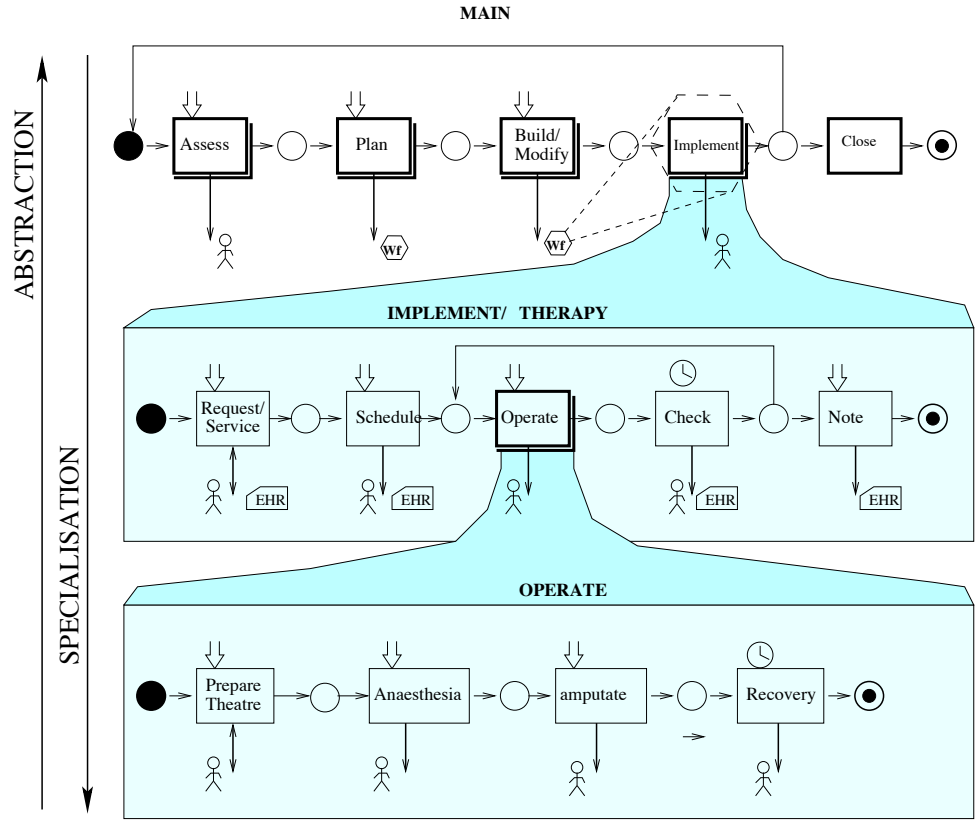


Figure 6.12: *Expanded Healthcare w/f utilising subworkflows*

then be combined using "if-then-else" rules to form more complex (workflow-like) computer-interpretable protocols.

6.5.2 Classes of Change Region Restrictions

We can categorise the types of dynamic changes that can be applied to a template care plan according to the types of restrictions imposed. These restrictions can be both at a schema level (i.e. before the case is instantiated), as well as at runtime.

1. A single Build/Modify task consisting of a single schema modification operator, operating on a process consisting of a single target task, where the task type is unconstrained. Restrictions can then be imposed at schema

level which can:

(a) restrict the target task to a subset of available tasks, and/or

(b) restrict one or more task parameters.

At runtime, the W*f*MS can either

(a) select/bind to specific tasks, or

(b) specify the value of task parameters

2. An unrestricted set of modification operators, operating on a restricted set of tasks.

3. A Build/Modify process operating on a skeleton target process. The Build/Modify process will constrain the target task parameters.

4. A Build/Modify process corresponding to an unconstrained target process.

The category of change restrictions imposed for a given circumstance, will depend on the complexity and nature of the base care plan, as well as the nature and business processes of the organisation or service provider.

6.6 A Healthcare Case Study

The need for goal-focused, self-modifying workflow is well illustrated by the problem of diabetes management in the community. Diabetes has a high prevalence in the Australian population, and the rate is rising in concert with obesity [Zim01]. There is a great deal the family physician (known in Australia as the General Practitioner, or GP) can do to coordinate the care of the diabetic patient [Mat00][Div03][Phi02]. Each element of a GP's care plan has the potential to

provide greatly improved outcomes for the patient. For instance, just considering the activity of foot examination and possible podiatry referral, the evidence indicates that diabetic foot complications are responsible for over 85% of non-traumatic lower limb amputations annually in Australia [Pay00]. Early identification of complications associated with diabetes has been shown to reduce the risk of future development of ulceration and gangrene - recognised as the precursors to amputation [Arm97]. The objective of workflow management in this domain is to ensure that all relevant care activities are followed through.

Table 6.1 lists the set of activities that must be documented for the Australian GP to make the maximum reimbursement claim allowable under the Enhanced Primary Care scheme[Div03]. In some cases, these activities concern the performance of specific acts to satisfy implied goals. Abstracting some of the tasks from Table 6.1, and combining with other commonly accepted diabetes management goals[Mat00][Phi02], we derive a goal hierarchy from which, in turn, we can arrive at a first-cut workflow for implementation of a diabetes management. As shown in Fig.6.13, this goal-derived workflow schema serves as a template for the tasks a GP may include in the care plan of a diabetic patient in order to achieve the high-level management goals. Referring back to Fig.6.2, this is the starting point for the subworkflow of the *Implementation* task, and in turn, consists of a set of parallel tasks.

Tailoring this workflow to a specific patient is achieved during the execution of the *Build/Modify* task, by applying the schema modification operators, we have described above, to a specific change region. In real world situations, the *Build/Modify* task may operate on more than one change region, and there may often be more than one *Build/Modify task* required to achieve a set of goals.

Each of the tasks of Fig.6.13 is initially abstract. Either of two schema operators can be used to move from abstract to concrete, executable tasks: *substitu-*

task	qualification
Assess diabetes control by measuring HbA1c	At least once per year
Ensure that a comprehensive eye examination is carried out	At least once every two years
Measure weight and height and calculate BMI	At least every six months
Measure Blood Pressure	At least every six months
Examine feet	At least every six months
Measure total cholesterol, triglycerides and HDL cholesterol	At least once every year
Test for microalbuminuria	At least once per year
Provide self-care education	Patient education regarding diabetes management
Review diet	Reinforce information about appropriate dietary choices
Review levels of physical activity	Reinforce information about appropriate levels of physical activity
Check smoking status	Encourage cessation of smoking (if relevant)
Review of Medication	Medication Review (at GPs discretion)

Table 6.1: *Completion of an annual cycle of care for patients with Diabetes Mellitus (from [Div03])*

te_task or *refine_task*. With *substitute_task* we can build the concrete task either by directly substituting in a concrete task or by substituting in a task primitive, such as the Exclusive Task (choice) primitive. Fig.6.14 illustrates this option as applied to Self-Management Education wherein the GP may undertake the education him/herself or refer the patient to a hospital-aligned Diabetes Centre for training through specialized diabetes nurse educators.

To make a task concrete through refinement can exploit the abstraction of subworkflow, for instance, via using a library of available workflow pattern templates based on clinical guidelines. The remaining diagrams illustrate detail pertaining to the Manage Blood Pressure(BP) subworkflow for patients with above-normal

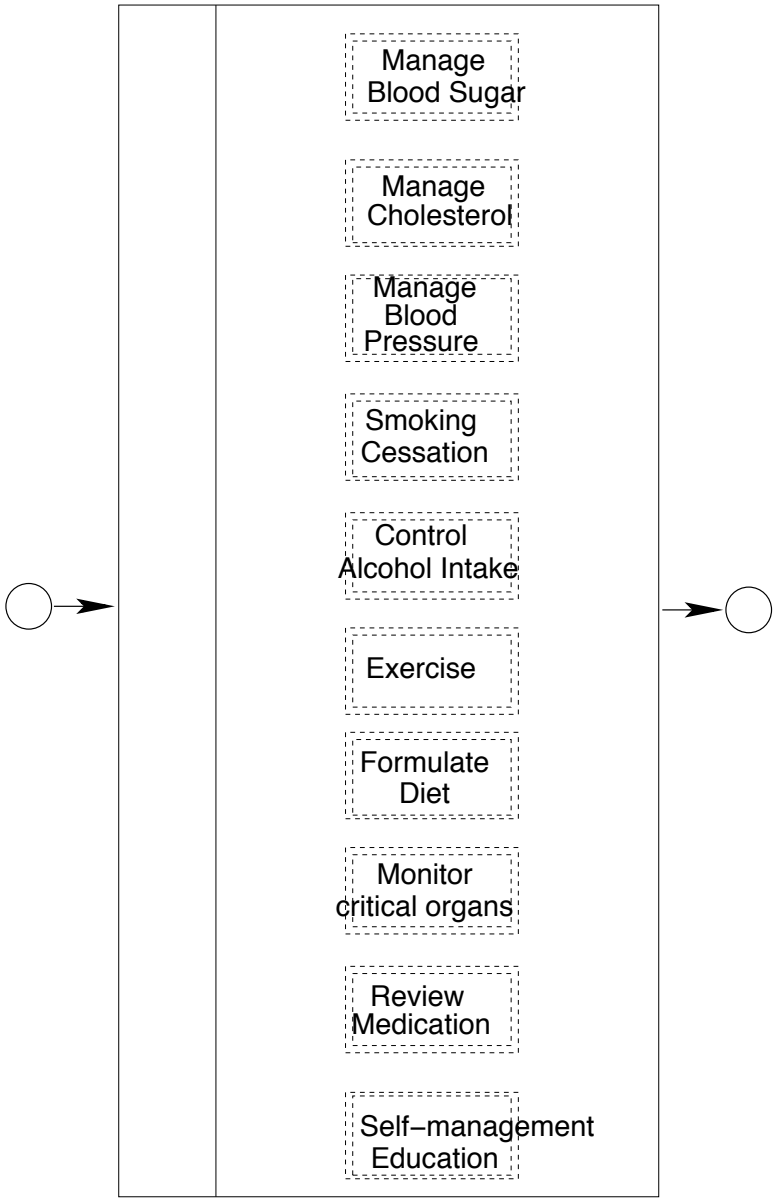


Figure 6.13: *Top-level tasks for Diabetes Management*

BP readings. These diagrams have been simplified from P/T Net representations for clarity, by not explicitly depicting intermediate states. Fig.6.15 illustrates a GP-based hypertension management workflow based loosely on a hypertension algorithm for diabetes mellitus in adults[Tex00]. This hypertension subworkflow

is, in turn, composed of a number of subworkflows, one of which (**progressive_therapy**) is expanded and illustrated in Fig.6.16.

Further application of schema modification operators is often necessary due to individual variation. Often this variation may be due to exceptions that are foreseen in the relevant guideline but are too detailed to explicitly represent in the workflow template. For example, the Texas Diabetes Council's hypertension algorithm [Tex00] has eight explicit footnotes, a sidebar table of supplementary considerations and many additional subtleties that are implied in word choices such as "preferred" versus "strongly recommended" options. Among these footnotes is one that indicates that early combination therapy "may be warranted" if systolic blood pressure is greater than or equal to 145mmHg or diastolic blood

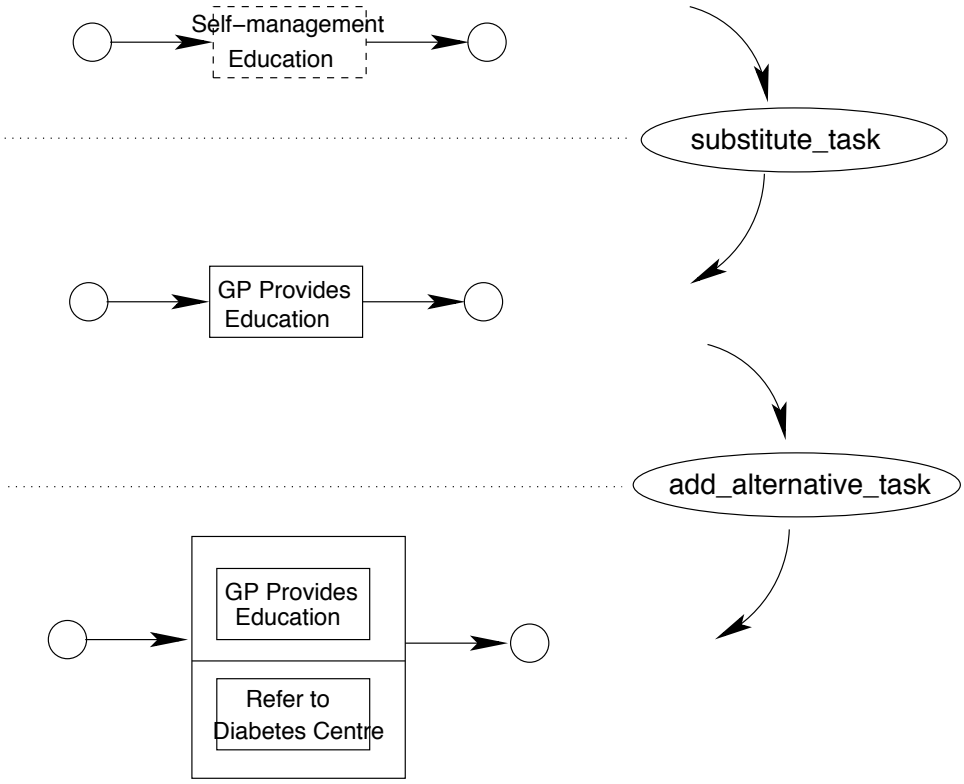


Figure 6.14: *Tailoring of Self-Management Education subworkflow*

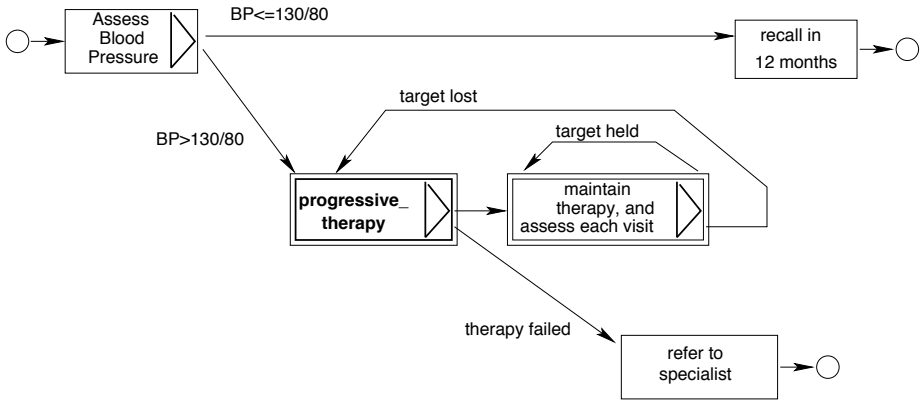


Figure 6.15: *Hypertension Management Wf Schema*

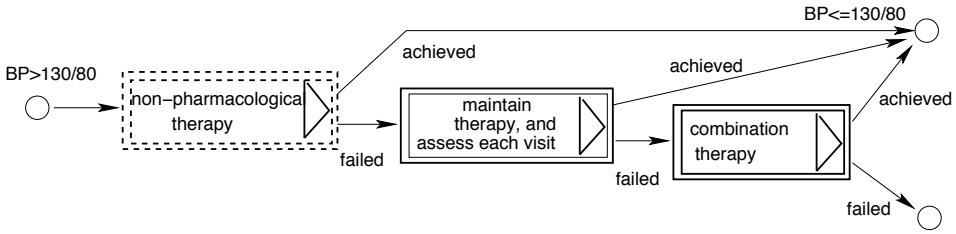


Figure 6.16: **progressive_therapy** *Subworkflow*

pressure is greater than 90mmHg. Rather than handling this as an exception in the running workflow, one can apply a series of schema modification operators as the workflow is built or completed (at the time of care planning). Fig.6.17 illustrates how such a modification can be undertaken using the *add_alternative_task* operator to insert a pathway directly from assessment of blood pressure to combination therapy (implicitly the guideline always allowed this, but it would just be too cluttered to provide every possibility). If the patient has already advanced to combination therapy at the time of care planning, or is known to have a very high systolic blood pressure, one can further tailor the workflow to the

patient by using the *limit_choice* operator to remove the option of attempting non-pharmacological and mono-therapies.

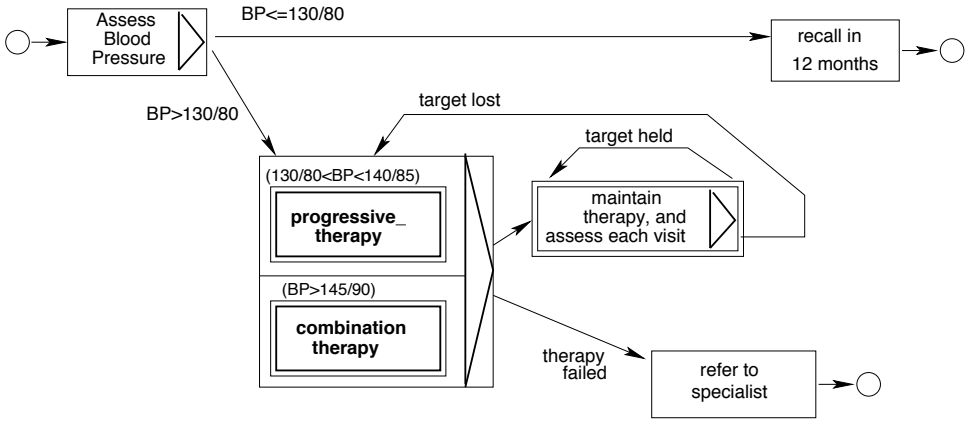


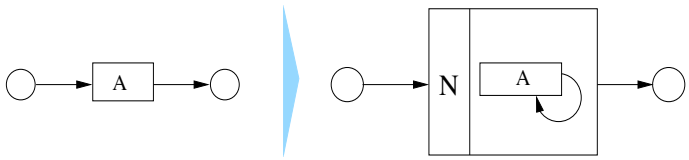
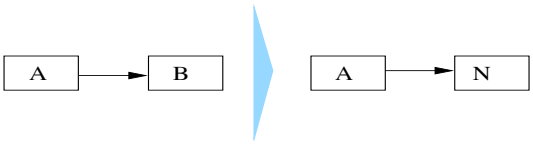
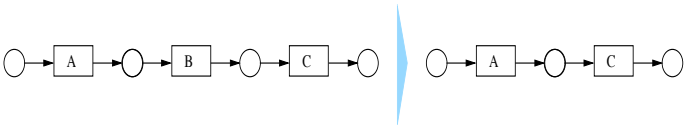
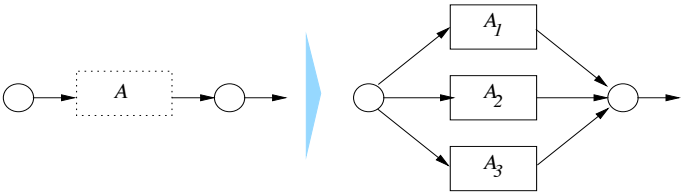
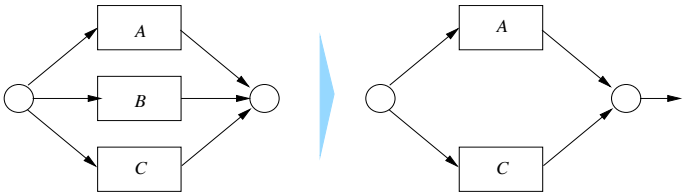
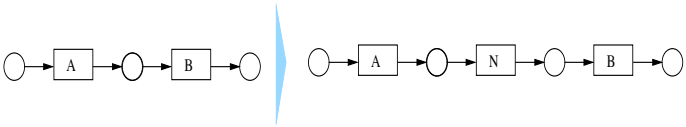
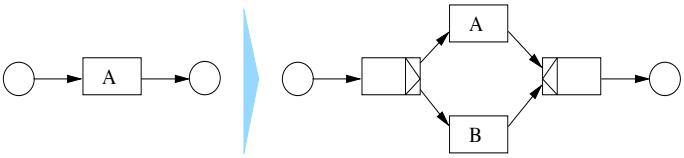
Figure 6.17: *Adding an Alternative Task to Fig. 6.15*

Quaglini[Qua01] and Panzarasa *et al* [Pan02] have already demonstrated that a WfMS can be successfully applied for coordination of real-world healthcare activities that achieves the workflow (or "careflow") indicated in an evidence-based clinical guideline. In their case (post stroke rehabilitation), however, the recommended process could be defined a priori to a greater level of detail than is commonly possible for broader community based care (e.g., as for diabetes management).

6.7 Summary of Operators

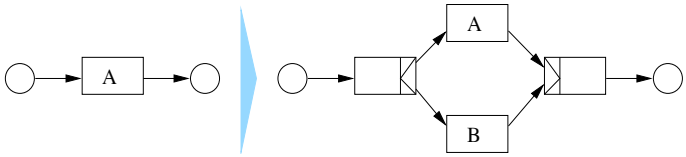
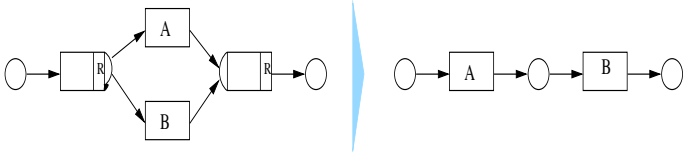
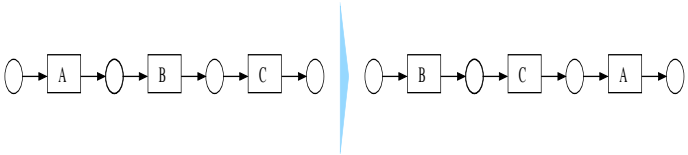
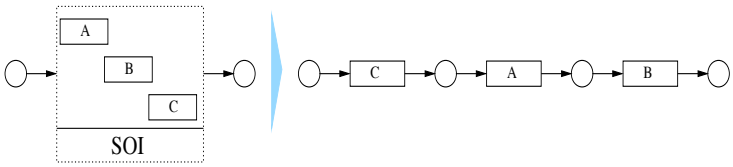
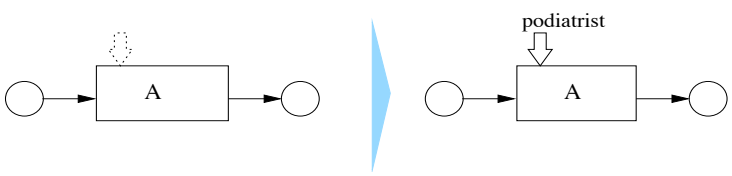
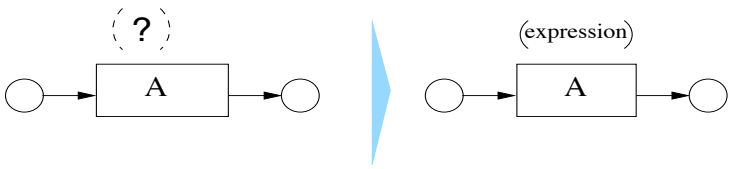
Table 6.2 illustrates the principal modification operators that support workflows encountered in chronic disease management.

Table 6.2: Table of Modification Operators

Operator	**Illustration**
Iterate	
Substitute an activity	
Delete a sequential activity	
Concrete to a choice of activities	
Delete a choice activity	
Insert a sequential activity	
Add an alternative activity	

continued next page

Table 6.2: *continued*

Operator	Illustration
Add a concurrent activity	
Convert parallel to sequence	
Reorder a sequence	
Impose Order	
Assign a role	
Assign an expression	

6.8 Discussion

This chapter has canvased a set of issues that confront the health informatics community when trying to introduce Workflow Management Systems into Chronic

Disease Management. In particular, it has illustrated a compelling requirement for self-modifying workflow schemas, which have the property of containing at least one task dedicated to altering the schema instance. The alteration is restricted to downstream tasks and often involves completion of the schema. A set of operators was introduced that can assist this process by converting abstract schema templates (Base Care Plans) into concrete workflow schemas. And by utilising subworkflows, the concretion can occur at runtime for each patient case. This section discusses some of the ramifications ensuing from adopting a self-modifying schema approach to workflow modelling.

6.8.1 Consistency validation

New workflow schemas need to be validated against a set of consistency constraints that ensure that cases are well-behaved. In most environments employing a WfMS , this validation can be carried out independently of the WfMS itself, and need only occur when a new schema is constructed or modified. In most production or administrative environments, this occurs relatively rarely. In healthcare, where schemas are modified on a per-patient basis, either the validation has to be tightly coupled with the WfMS , or the workflow schema modification rules themselves, have to ensure that the workflow schema remains well-behaved.

6.8.2 Combining workflows for comorbidities - activity crediting

In healthcare, we are often faced with situations where comorbidities (more than one illness occurring simultaneously in a patient) require the cooperation, if not merging of independent workflows. Self-modifying workflow schemas can be extended to support constructs for merging two or more schemas and for crediting

tasks which are common across the schemas. For example, if a diabetes patient has a comorbid condition of ischaemic heart disease (IHD) and both the diabetes and IHD workflows contain a task *measure blood pressure* then under certain circumstances these two tasks could be merged into a single task, rather than the patient being required to repeat the test for each workflow. This *activity crediting* funtionality, enabled by self-modifying workflow schemas, is the subject of the next chapter (7).

6.8.3 Matching constraints to goal parameters

Wherever goals can be realized by specific processes or subworkflows, the goal target parameters should become specific constraints of tasks within the subworkflow. The mapping of goal parameters to task constraints could be formalised to ensure that the complete set of goals for a given patient's care episode are addressed satisfactorily by the resulting workflow schema instantiation. For instance, the author's prototype implementation described in chapter 8 provides for the automatic generation of nested workflow schemas from a given goal hierarchy, yet goal targets still need to be entered manually into the resulting process schema(s). From a practical viewpoint, the desire to automate their mapping implies a formalism to be expressed within clinical systems which may impose unacceptable burdens on clinicians.

6.8.4 Implementation issues

The challenges facing W*f*MSs to provide for self-modifying workflows, and the incorporation of such systems into larger applications such as Clinical Information Systems are indeed significant. Not only do new approaches to storing schemas as metamodels need to be refined, but appropriate tools for workflow schema completion need to be developed. Such tools would, of necessity, require comprehen-

sive "libraries" of domain-specific models, composed of building blocks (workflow schema primitives) similar to the ones described herein. The prototype discussed in chapter 8 supports the creation of such domain-specific task libraries.

It is important to note that it is not envisioned that General Practioners (GPs) should work directly with workflow editor tools. For that matter, there is no evidence that they will wish to work from a graphical view of workflow at all. A recent evaluation from the UK [Rou03] indicates that GPs find it tedious and confusing to navigate graphical flow representations of practice guidelines. Warren et al [War99] implemented a care planning tool for GPs wherein the GP's view was simply in terms of the presence of any of a selectable set of services in the patient's care plan, with the option to adjust the frequency of that service. Such a high-level care planning view could be used to specify the combination of subworkflow templates "behind the scenes" with respect to the doctor's view. When combined with the goals that each of these services supports and a comprehensive sharable electronic patient record system, a powerful care planning system can be created.

6.9 Summary

This chapter examined aspects of modifications of a workflow schema required to meet the changing needs of an individual patient. The changes described are not merely limited to the specification of a static schema, individualised for a patient, and then instantiated. They also include those changes required to support a running workflow instance for the patient, as the patient's circumstances, the environment, medical knowledge and the patient's health providers' organisational structures evolve. The key to supporting these changes is in ensuring that explicit *assess* and *alter* tasks are built into the workflow schema itself. Just as W*f*MSs are designed to ensure that activities are undertaken to achieve goals, so

too should WfMSs ensure that necessary changes, either to the goal targets, or to their associated activities, are also undertaken.

The practical contribution of this dissertation is in providing an approach that extends the scope of explicit representation of the coordinated care process even when the complete plan is not known prior to assessment of the patient. Furthermore, the approach shown here allows re-building of the plan in response to subsequent clinical assessments.

CHAPTER 7

Crediting Redundant Activities

" A world of dew: Yet within the dewdrops - Quarrels."

Kobayashi Issa c.1800.

7.1 Introduction

Distributed workflow management systems (W*f*MSs), where an overall business process is undertaken by a number of different organisations, can lead to a situation where similar tasks might be undertaken by different organisations to achieve the same goal or part thereof. This duplication of services is inefficient, can lead to delays in achieving the overall goal, and can impact on the quality of the services being provided. This is particularly true in healthcare, where unnecessary interventions can have tragic consequences. Patients being managed for one or more chronic conditions, can be particularly prone to such adverse impacts of repeated interventions. A common example of this is with the prescription of medications, whereby several clinicians could be prescribing the same medication without being aware of similar prescriptions provided by other clinicians, either for the same, or sometimes for different reasons.

This chapter introduces to workflow process modelling the concept of activity

crediting, whereby an activity scheduled to be undertaken somewhere in the overall workflow, can be credited, either in part or in full, against a similar activity elsewhere in the workflow. Such crediting could be determined to some extent, at schema definition time, but quite often, with dynamically created workflows or subworkflows, the crediting may need to be undertaken at run-time, after the specific workflow instance has started. The semantics and rules for such crediting can be quite complex, with temporal aspects often playing an important role. Temporal deadlines and activity-outcome time to live parameters need to be taken into consideration when determining the potential for activity crediting to be viable. Another confounder discussed in this chapter is the effect of intermediate tasks on activities that might be candidates for crediting.

There are also related but dissimilar issues to activity crediting - those of activity discrediting, activity dependence, and activity conflict, some of which are also raised in this chapter.

7.2 Healthcare Requirements

This section looks at the specific problems of task overlap encountered in healthcare when workflows span multiple service providers, and how the ideas in the remainder of the chapter contribute towards a solution to this problem. The notion of different *activity target objects*, which arise in healthcare workflows to a greater extent than in most other domains is also revisited.

7.2.1 Problem

The motivations for the use of Workflow Management Systems in healthcare are twofold, firstly to improve health outcomes for individual patients, and secondly to make healthcare service provision more efficient. When care is shared amongst

a number of providers, there is potential for conflicts to occur that might compromise health outcomes. There is also the possibility of redundant activities being undertaken, when workflow is seen merely as a mechanism for coordination of resources, and the relative ordering and timing of tasks. The primary focus of this chapter is to address the duplication of services that often arise in healthcare service delivery. Such duplication affects both the quality of patient care, as well as the efficiency and cost of care delivery.

7.2.2 Requirements

By making information pertaining to the characteristics of tasks available to the WfMS , it is possible to identify situations where tasks can be avoided or modified to alleviate redundancy. Such use of task knowledge, and the ensuing modification of the workflow, is referred to here as *activity crediting*. In some circumstances, redundancy can be identified at the goal level, in which case it is referred to as *goal crediting*. Goal crediting is more likely to occur when care plans are being developed for patients with comorbidities. In these cases, sets of goals from independent clinical guidelines may need to be merged together to create a single integrated care plan for the individual patient. Below, are outlined the primary requirements that need to be considered when addressing the above problem. Some additional issues related to these requirements, are mentioned at the conclusion to this chapter.

Crediting Scenarios

Our requirements are based on a set of scenarios that enable crediting to be classified according to the following perspectives:

Full Crediting: Under some circumstances, the crediting of a goal or activity could be complete, in the sense that the goal or activity could be unconditionally

dropped from the overall business process.

Partial Crediting: In many instances, it is unlikely that the completion of an activity somewhere in a workflow schema will make another activity completely redundant. However, the second activity may need to be modified to take into account the effects of the first activity. Such compensations can be referred to as "partial crediting". Partial crediting equates to constraints being placed on the crediting operations that modify the workflow schema at runtime. Such constraints restrict the validity of the crediting and could be based on time or on specific events.

Temporary Crediting: Temporary crediting refers to crediting being conditional upon the respective time at which similar activities occur. Depending upon their temporal dispositions, this could lead to one activity being cancelled entirely, postponed for some period, or foreshortened in duration.

Permanent Crediting: Sometimes, it will be appropriate for a goal, activity or data item to be dropped permanently from the workflow since its enactment or acquisition has been made redundant by a prior action.

Overlapping Tasks

It is important not to eliminate a target task, if its function overlaps that of the prior crediting or source task. For example, if a prior task achieves a blood pressure of 130/80 by, say medication, and a later task aims to achieve a blood pressure of 135/85 by exercise, it may be desirable not to credit the first task in order to eliminate the second, since there may be additional benefits gained by the exercise task. For the W*f*MS to be able to identify such overlaps, there must be sufficient detail expressed explicitly in the task definition, in this case as a post-condition or Quality of Service parameter of the exercise task.

Intervening Tasks

Even though an activity might have been identified as a candidate for crediting, it is possible that some intervening task, occuring after the first task, but prior to the second task in the crediting pair, could cause the crediting to no longer be valid.

Extraneous State Changes

Similar to the case of intervening tasks invalidating crediting as just described, we could have situations where unexpected state changes occur, to any of workflow state, patient state, environment state or resource state that similarly invalidate a possible activity credit. For example, a patient could be undergoing treatment for diabetes, whereby an exercise regime to reduce blood pressure credited and allowed for the skipping of a medication-based hypertension treatment. If the patient was physically incapacitated by some accident, then the exercise regime would no longer be a valid activity for treating hypertension, and the medication-based treatment might need to be reinstated into the workflow.

Unwanted Crediting

Unwanted crediting refers to situations whereby it is undesirable to cancel an activity because it has already been undertaken somewhere earlier in the workflow. For instance, a second opinion on a diagnosis may be an integral part of a workflow schema, in cases where it is important to have near certainty before proceeding down a path. Automatic crediting, and consequential cancelling of such second opinions would not be wanted or warranted.

Conflict Resolution

Akin to activity crediting is the notion of activity avoidance, due to potential conflict of outcome. Traditional W*f*MSs are unable to deal with such conflicts, since considerable domain knowledge is required to determine potential conflicts. Just like crediting, such conflicts could be resolved either at the goal level, or at the level of individual activities. Any facilities provided by an extended W*f*MS could be utilised to help resolve a conflict once it has been identified, *e.g.* users could be notified and workflow schemas adapted to help provide a solution. A W*f*MS could identify all actors in the healthcare team for a given patient care plan, and notify each actor through their appropriate communication channels. An additional activity could be automatically inserted into the current workflow schema to ensure that the conflict is resolved to the satisfaction of a nominated actor.

Quality Assurance

It is essential in any healthcare system to ensure that mistakes are minimized. Any form of activity crediting should be highly conservative, and subject to runtime validation and manual authorisation by approved and appropriate clinicians. There should be "break-glass" emergency exception handling to allow clinicians to overide perceived system-based activity crediting. Where crediting has already been approved by one clinician, such crediting should be made known to other relevant care providers for that patient.

7.2.3 Contributions

This chapter describes mechanisms for eliminating redundant activities as well as partially crediting the work achieved by previous activities in a given workflow

instance. The contributions include the use of a two-tier methodology for representing and viewing business processes, based on separate goal and process views, and a two-phase methodology for firstly discovering, and secondly crediting selected components of the overall business process. I call the first phase *candidate discovery*, and the second phase *component crediting*. Two classes of crediting are identified, namely *permanent crediting* and *temporary crediting*. The chapter also describes a set of still unresolved issues that need to be addressed for activity crediting to be supported effectively in the clinical setting.

7.2.4 Approach Overview

The strategy adopted to address the problem of duplicated services hinges firstly on an approach to discovering candidate business process components for crediting and secondly on an approach to manage the crediting process itself. For candidate discovery, a two-tier methodology based on the separation of high-level goals from lower-level processes is utilised. For component crediting a self-modifying workflow architecture that embeds specific workflow modification activities into the workflow schema itself is adopted. These activities make use of dedicated crediting operators to achieve and manage the component crediting at runtime. In order to support these two aspects, a representational model is introduced and supported by a separate execution model.

7.3 Representational Model

The representational model uses a two-tier goal/process architecture and a library and ontology of predefined task definitions, as described below.

7.3.1 Two-tier Architecture

The two-tier representational model is based on defining a high-level goal view via a goal hierarchy (see 7.3.1), and a corresponding lower-level process view via a workflow schema (see 7.3.1). The workflow schema is derived from the goal hierarchy as described in Chapter 5. These two views provide for visually representing and interacting with the workflow for each instance.

Goals

The goal hierarchy can then be mapped into a workflow schema for the patient, using a combination of clinical guidelines and organisational business rules and constraints. It is possible to identify potential candidates for activity crediting, even at the goal level.

This chapter continues the example from Non-Insulin Dependent Diabetes Mellitus (NIDDM) management to illustrate a possible goal hierarchy, since diabetes management can involve many service providers (general practitioner, diabetes educator, nurse, endocrinologist, dietician, opthalmologist, podiatrist) and many potential activities and thus a contender for involving activities that interact. Consider the simplified goal hierarchy, for the management of NIDDM, already introduced in Fig.5.1, and repeated below as Fig.7.1.

In this hierarchy, most goals are achieved by the achievement of all their immediate child goals, indicated by a single arc. Alternative goals are indicated by a double arc, as in goals $\mathcal{G}_6$(Beta-blocker therapy) and $\mathcal{G}_7$(ACE-inhibitor therapy). These two goals are mutually exclusive (annotated in the figure by an X), and are prioritised, such that ACE-inhibitor therapy is the preferred goal.

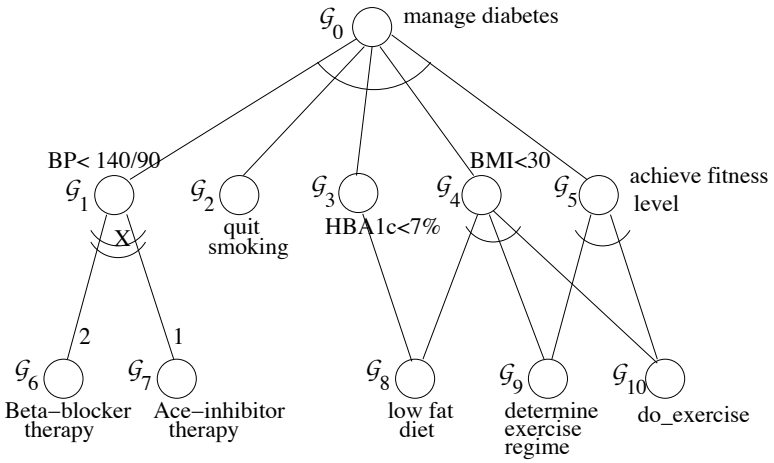


Figure 7.1: *Goal-graph for Diabetes Management (simplified).*

Processes

A *process* is a set of tasks or activities that achieves a goal. A process is implemented by a workflow engine that controls the activities, by either invoking specific activities itself, or placing them on the worklists of participants and flagging them as ready to be undertaken. The ordering of activities in a process is governed by a workflow schema. Thus, a process is an enactment of a workflow schema to achieve a specific goal. Workflows, and thus processes, may be nested and referred to as subworkflows or subprocesses. Each subprocess is designed to achieve a subgoal of the overall objective (root goal) of the goal hierarchy for that case. Such a process skeleton, derived from the diabetes management goal hierarchy is shown in fig 7.2, where each goal corresponding to its respective process is illustrated on the right-hand side at the corresponding vertical position in the diagram.

In moving from the goal view to the process view, we are adding domain, organizational and environment knowledge in order to elucidate the explicit set of activities, resources, conditions and control flow that needs to be applied. Every process $\mathcal{P}$, designed to achieve a goal $\mathcal{G}$, will have a set of activities A; a set of one

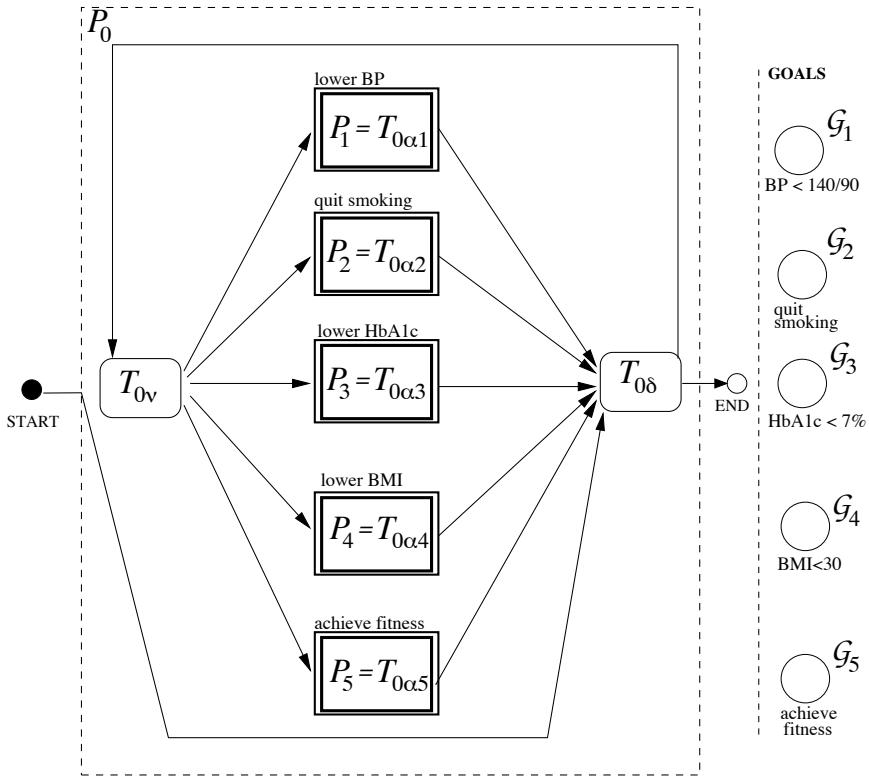


Figure 7.2: *Process model for Diabetes Management (simplified).*

or more objects upon which the process acts O (described below); a set of edges E that join activities, defining temporal dependence (flow) between activities; an associated set of resources R; achievement conditions G; and a goal validity time V. i.e. $\mathcal{P}$ is described by the tuple (G, A, E, O, R, V). G expresses the boundary conditions on the goal, such as the achievement level, achievement tolerance, goal preference, goal priority, etc.

Fig.7.3 shows the expanded form of subworkflow P_4 of fig.7.2. This workflow illustrates the 3 processes corresponding to the subgoals needed to achieve a lower Body Mass Index(BMI). The first of these goals, *low-fat diet* is also a subgoal of G_3, lower-HbA1c.

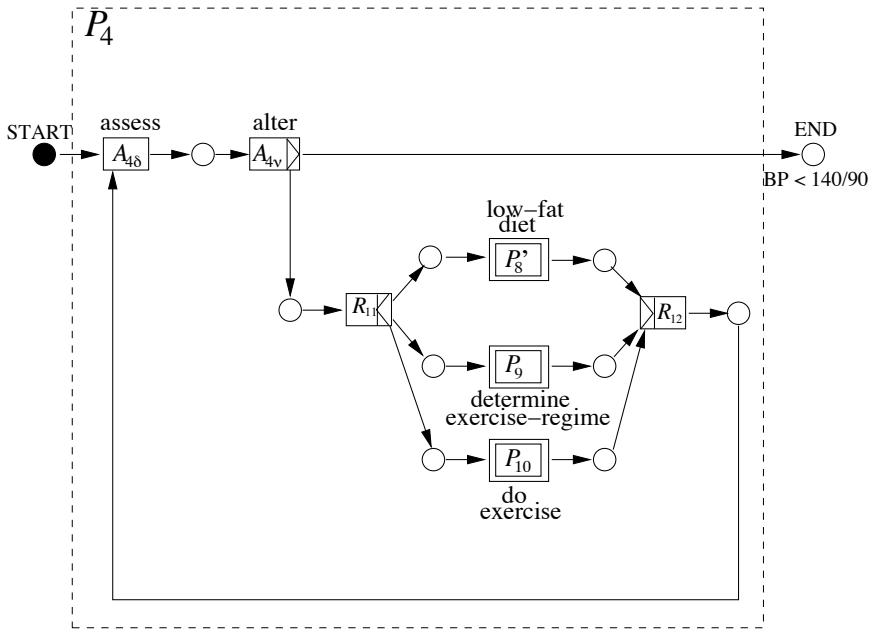


Figure 7.3: *Process model for subworkflow P_4 - LowerBMI.*

Characteristics of Tasks

In order to search for redundant or conflicting tasks within a specific process schema, tasks have to be compared according to a set of characteristics that each task possesses. The following is a list of such characteristics:-

- Task Purpose
- Task Validity time
- Type of treatment
- Task Name
- Task Side effects
- Task Postcondition

- Task Preconditions
- Quality of Service (QoS)

When tasks become activities within a specific workflow schema, they are also given priorities. The priority assigned to a task might even change during the enactment of the case.

Task Library and Task Classification

In order for activity crediting to be viable, the W*f*MS needs to be aware of details of each activity in a given workflow schema. The best approach, based on goals as described previously, is for each schema to be derived from the goal hierarchy, as a set of nested subworkflows. Each subworkflow corresponding to a leaf goal is then constructed by assembling appropriate activities from a common organisational task library. This task library should be built from best-practice clinical guidelines [Fie90], and should vary in scope according to the skills of the organisation. Tasks are classified according to an information model of healthcare concepts, so that tasks to accomplish similar goals are grouped/classified together. The intention, generic pre-conditions and generic post-conditions of each task are made explicit and stored in the task library. Candidates for activity crediting are then identified by the activity's path in the task hierarchy, the activity's intention, and the activity's data pre/post-conditions.

As part of Australia's electronic health records project Health*Connect*, the author has developed an information model, which includes amongst other aspects, a model of "events". This ontological model, which focuses predominantly on healthcare activities, including investigations and interventions, can be used as a basis for a task library. The ontology is shown in fig.7.4.

However, this "high level" ontology belies the complexity of describing health care tasks. It does not even specialise to the level of pathology tests, which

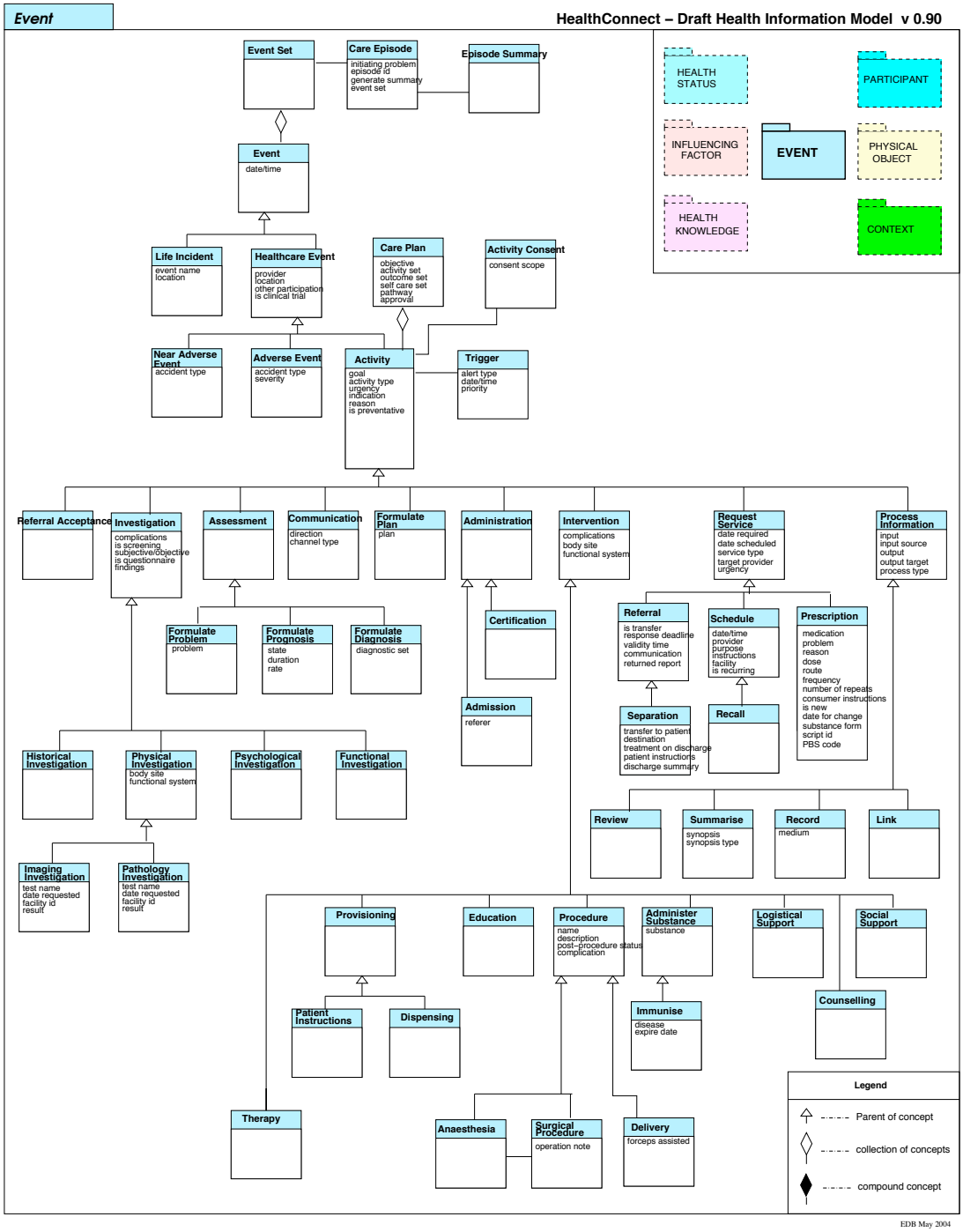


Figure 7.4: *Ontological model of healthcare events*

constitute perhaps one of the best classified areas of healthcare intervention. In Australia, there are some 2500 individial pathology tests classified and coded for use in pathology test ordering. A similar number has been identified by UK's National Health Service in their code bindings to SNOMED-CT names for pathology tests. The names of surgical procedures also runs into the thousands, and once additional parameters are included, the difficulty for task libraries to accurately and consistently describe a significant subset of all potential health services can readily be appreciated.

Activity Target Objects

In section 4.1.6, the concept of an *activity target object* was introduced under the general discussion on *state* and the classes of objects for which state must be considered. *Activity target objects* are reintroduced here in order, firstly, to support those special activities in self-modifying schemas that act on the workflow schema instance itself to perform crediting, and secondly, to support the necessary temporal constraints that may be required for crediting.

Thus an activity target object O can be any one element of the set $\{C, P, W, E, R\}$ where C is the workflow case, P is the patient, W is the workflow schema, E is the environment, R is the set of resources involved in the case.

Healthcare activities are often categorised into *Investigations/Observations*, *Evaluations/Assessments* and *Interventions/Treatments*. From our definitions of state above, we can say that Investigations measure patient state (S_P) in order to change workflow data state (S_{Wd}); Assessments change S_{Wd}, often by temporal, spacial and other algorithms applied to prior values from S_{Wd}; Interventions are intended to change patient state (S_P).

7.4 Execution Model

The two-phase execution model consists of a *candidate discovery* phase followed by a *component crediting* phase. These two phases are invoked at the commencement of the execution of each case, and whenever a change occurs to the status of any of the components identified by the *candidate discovery* phase as being involved in potential crediting.

7.4.1 Candidate Discovery Phase

Workflow crediting aims at determining which elements of a business process provide similar functionality, and therefore might be potential candidates for crediting. Three types of components can be identified for which such synergy can occur, namely those of goals, activities and data. A goal, activity or data variable has a high degree of functional correlation with another, if it contributes in a similar way to the overall objective of the business process. Taxonomies or classifications of concepts are important for the determination of functional correlation.

Before detailing the mechanism for candidate discovery, let us first examine how goal discovery is undertaken using the goal-level representational model, whilst activity and data discovery are undertaken at the process-level.

Goal-level Matching refers to the identification of duplicate goals in a goal hierarchy for a specific healthcare service. Such goal hierarchies might be quite extensive in the treatment and/or management of chronic conditions, especially where comorbidities i.e. several concurrent conditions exist. Subprocesses (sets of activities) that would normally be enacted to achieve alternative or duplicated goals would be candidates for dropping or bypasssing in the overall workflow schema. Goal-level crediting allows localisation of modifications to specific are-

as of the workflow schema, minimising the overhead and side-effects of schema modification.

Process-level Matching refers to the identification of particular activities or redundant activity data items (state variables) that might be collected by different activities in the overall workflow. It is not sufficient to simply use the name of the activity as a means of identifying where such redundancies might occur. Many activities in healthcare are aimed at assessing or determining patient state variables, such as blood pressure, weight, blood lipid levels, etc. Many tasks could have the determination of these variables as either a direct or indirect target of the activity.

Candidate components for crediting are determined firstly by a goal-space search of the goal-hierarchy for matching goals. This is undertaken by following each path of the directed acyclic graph, from the root, to all leaves. Duplicate candidates are identified by the name of the goal and are further matched by their target achievement level. Temporal conditions are checked using the prescribed validity times associated with each goal. Once completed, if candidates are found, then these are placed on a candidate list for later processing. Mutually exclusive activities (alternatives) are checked, and if any in an alternative set has been commenced, then the remainder of the set is placed on the candidate list.

Next, the corresponding workflow schema is searched for correlating activities. These are identified by the name and position in the task library of the template task used to create each activity. Any candidate activities are also placed on the list. Next, a search for data in activity post-conditions is undertaken to determine activities which collect or set identical data variables.

Finally, the candidate list is passed to the component crediting phase, which undertakes the crediting as described next.

7.4.2 Component Crediting Phase

Self-modifying Workflow: Activity crediting requires operators to support instance-level adaptation of workflow schemas, together with the cooperative interaction of healthcare participants. To support the required level of flexibility, we extend upon work of chapter 6 which introduces explicit schema modification tasks into workflow models of healthcare to ensure and facilitate the adaptation and modification of an abstract workflow schema at runtime for each patient case. Such activities are designed to change workflow state S_W (both S_{Wd} and S_{Wc}), resulting in additional activities, altered activities, replaced activities, deleted activities or altered flow. As such, these modification activities differ from conventional activities in that they act on their own workflow schema instance as the target object of the activity - hence the label, *self-modifying*. Considerable research, e.g. by Ellis *et al* [Ell95], Reichert & Dadam [Rei98], Sadiq *et al* [Sad01; Sad02], Manolescu & Johnson[Man99] and many others has formed the basis for the ideas leading to the concept of self-modifying workflow.

Every goal in the goal hierarchy has a corresponding workflow process in the workflow schema, such that the goal hierarchy maps to a nested set of subworkflows. Each subworkflow contains a goal assessment activity (*assess*) and a workflow alteration activity (*alter*). Component crediting is implemented by crediting operators, which are activity methods of the top-level *alter* activity (refer to fig. 7.2), or in a similar *alter* activity in one of the subworkflows (e.g. within $P_1 ... P_5$ of the same figure). These *alter* activities have the workflow schema itself as the target object of the activity, rather than the patient, a resource, or the environment. A crediting workflow activity is invoked as part of each alter task and placed on the worklist of users who belong to the crediting role. Any candidate goals, activities or data variables are presented to the creditor for acceptance, if they correspond to the goal/subworkflow that is currently being executed. If

the change is accepted by the creditor, then the corresponding crediting operator is used to apply the change to the running workflow schema, and to the goal hierarchy if appropriate.

Crediting operators refer to the set of workflow modification operators that play a part in any form of crediting. They are intended to alter the workflow schema, downstream of current activities, and do so by bypassing, adding, deleting, replacing or altering activities. Altering of activities may mean adding, deleting, replacing or changing data parameters of activities, specified as either pre- or post-conditions of the activity. The following workflow modification operators are introduced to specifically address changes to workflow schemas where one activity in the schema can be identified as providing some functionality similar to that of another activity in the same schema. These operators can be expressed as an ACTIVETFL [Mü02] formula, where appropriate, utilising the temporal semantics available therein to conditionally apply a change to a schema, either by restricting the change to apply for some duration only, or in response to a specified event. Thus changes to the workflow can be specified as either permanent or temporary. Temporary operators, in turn, may need to credit, or uncredit a workflow component, and crediting can be done conditionally where the temporal constraint is known at the time of crediting, or else unconditionally, in which case a corresponding uncredit component is required. Temporal constraints are the most likely candidates to influence the viability of activity crediting. We need to tag certain state measurements and state changes with their validity time, or else have a mechanism by which the workflow engine can obtain this information generically from clinical knowledge. Pre-conditions for activities will often be specified in terms of predicates on state variables. E.g. "if patient blood pressure is greater than 150/90mm Hg, then ..". If the patient's blood pressure has already been measured by a previous activity, then not only does this need to be known,

for activity crediting to be possible, but also, both the time of last recording, as well as the validity time, need to be known.

Here is a summary of the crediting operators supported:-

- *disableGoal*: disable a goal in the goal hierarchy, such that the corresponding workflow process is not undertaken. Goals are identified by their absolute path in the goal hierarchy, e.g.

$$disableGoal(ManageDiabetes/BMIlt30/lowFatDiet)$$

- *enableGoal*: (re)enable a goal in the goal hierarchy. A goal may need to be reinstated if the crediting goal is not achieved, or if extraneous state changes cause a service provider to deem it desirable to reachieve the goal.

- *deleteGoal*: delete a goal from the goal hierarchy, such that the corresponding workflow process is removed and no longer undertaken.

- *skipActivity*: skip an activity from the current schema instance, for some or all of the remaining execution of the current case. The following rule:

$$WHEN\ \ completed(A_1)$$
$$THEN\ \ skipActivity(A_2)$$
$$VALID{-}TIME[now,now+(7,day)],$$

  states that activity A_2 should be skipped for 7 days on the completion of activity A_1.

- *restoreActivity*: remove the bypass from a skipped activity.

- *skipConcurrentActivity* is a specialisation of *skipActivity*, which bypasses a path from a concurrent path set. This operator is illustrated in fig.7.5, being used to credit (skip) the low-fat diet subworkflow.

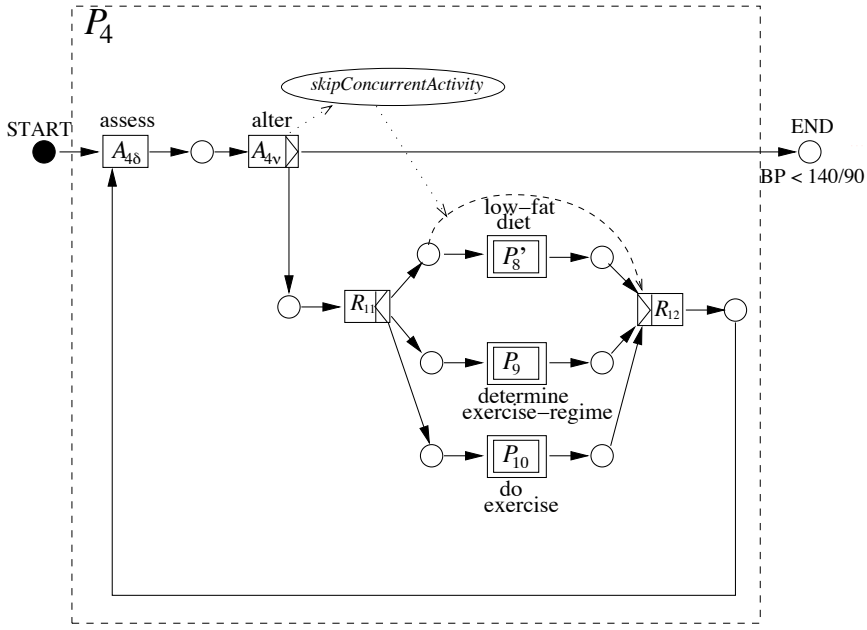


Figure 7.5: *applying a crediting operator to skip lowfat diet*

- *restoreConcurrentActivity*: remove the bypass from a skipped activity belonging to a set of concurrent activities.
- *skipChoiceActivity* is a specialisation of *skipActivity*, which bypasses a path from an exclusive-OR (choice) path set. If this operator is temporally constrained, it may need to be reinstated at a later date/time. E.g.

$$skipChoiceActivity(A_2)$$

$$Unless\ hypertensive(P)\ VALID{-}TIME\ now,$$

The above formula states that activity A_2 should be skipped unless patient P has become hypertensive.

- *restoreChoiceActivity*: remove the bypass from a skipped activity belonging to a set of alternative activities.
- *deleteActivity*: delete a named activity from the workflow schema for this

case.

- *deletePostcondition*: delete a data item required as an output of a specific activity. e.g.

$$deletePostcondition(A, parameter{=}BP),$$

  where A is the activity, and $parameter = BP$ represents the requirement to collect the patient's blood pressure through this activity.

- *addPostcondition*: add a data item required as an output of an activity.

- *addTempPostcondition*: temporarily add a data item required as an output of an activity.

- *deleteTempPostcondition*: temporarily remove a data item required as an output of an activity.

- *alterPostcondition*: alter a postcondition on an activity.

- *disablePrecondition*: disable a precondition on an activity.

- *enablePrecondition*: enable a precondition on an activity.

- *alterPrecondition*: alter a precondition on an activity. This might be used, for example to relax or further constrain a data value, required for a particular activity. e.g.

$$alterPrecondition(A, parameter{=}BP,$$
$$condition{=}''BP{>}140/85'')$$

7.5 Implementation

Activity crediting relies on participants' understanding of the entire care process or workflow schema for each patient. Good process monitoring tools are essential

for this understanding, and for each activity, or change in workflow, a snapshot of the case, including rationale for any changes, needs to be available to all relevant participants. An annotated runtime view of the goal hierarchy can be presented to clinicians as a synoptic view of the case, showing which goals have been achieved, supplemented with times and durations.

As discussed in chapter 8, the author's implementation methodology is based on a prototype workflow engine that extends several of The Workflow Management Coalition's Application Programming Interfaces (API) [Fis01], by providing support for goal views through a Goal Definition Language and Goal to Process Transformer. Support for runtime workflow schema alteration is provided through Event/Condition/Action (ECA) rules similar to those developed for the ACTIVETFL framework used in the workflow management system AGENTWORK WfMS [Mü02]. AGENTWORK has been applied in HEMATOWORK[Uni03] for the management of hemato-oncology, which covers the diagnosis, therapy and follow-up of cancer patients suffering diseases of the hematological and lymphatic node system.

7.6 Summary

This chapter outlined the requirements, and introduced a methodology for addressing the duplication of services that might occur in business processes in complex domains such as healthcare. The approach adopted was one of characterising goals and tasks, and using the set of characteristics to flag potential candidate tasks as redundant or partially redundant. Once these candidate tasks have been identified, the finalised list of candidates is presented to the case manager or care team for action. The possible actions are a) ignore the advice and proceed, b) remove or skip over the activity, or c) alter some aspect of the activity to compensate for

work already undertaken, such as modifying a dose, or other test or treatment parameter.

Several other issues should be considered when implementing activity crediting mechanisms in practice. These include cost implications and resource recovery. When crediting one activity against the other, for partial or complete elimination of one activity from the workflow, it may be important to compare the cost of each activity, and to consider this in determining which activity might be (partially) skipped. The efficiency motivation for activity crediting is premised on the saving of resources. Any recovery of resources no longer required to service activities that have been credited, should itself be handled efficiently to maximize the advantage of such recovery.

Part III

Trial Implementation

Chapter 8 - *Stream*Line Workflow Management System

Chapter 9 - Case study: Management of Diabetes Mellitus

CHAPTER 8

*Stream*Line - Workflow Management System

> "*Things in our life simply don't go according to set decisions. One glides into a new epoch, and the so-called 'decision' is, as a rule, only the final summing-up of items long since entered into the ledger by life itself.*"
>
> Franz Resenzweig, letter, 1928.

8.1 Overview

*Stream*Line is a complete W*f*MS developed by the author to provide a practical demonstration of those flexible, goal-based features discussed in Part II. *Stream*Line is a modular, client server application and extensible application framework, based to a large extent on the Application Program Interface (API) specifications of the Workflow Management Coalition (WFMC). *Stream*Line is a complete application framework, in that it provides full persistence via an abstracted database layer capable of supporting a variety of database engines; client server interaction via a dedicated XML-based command language similar to SOAP[1] or XML-RPC[2]; workflow schema definition; workflow schema enactment;

[1] Simple Object Access Protocol
[2] XML Remote Procedure Call

dynamic schema alteration for each running process. *Stream*Line also provides for extensibility via well-documented C++ API calls through the same modularised classes that are used by the services-generation layer.

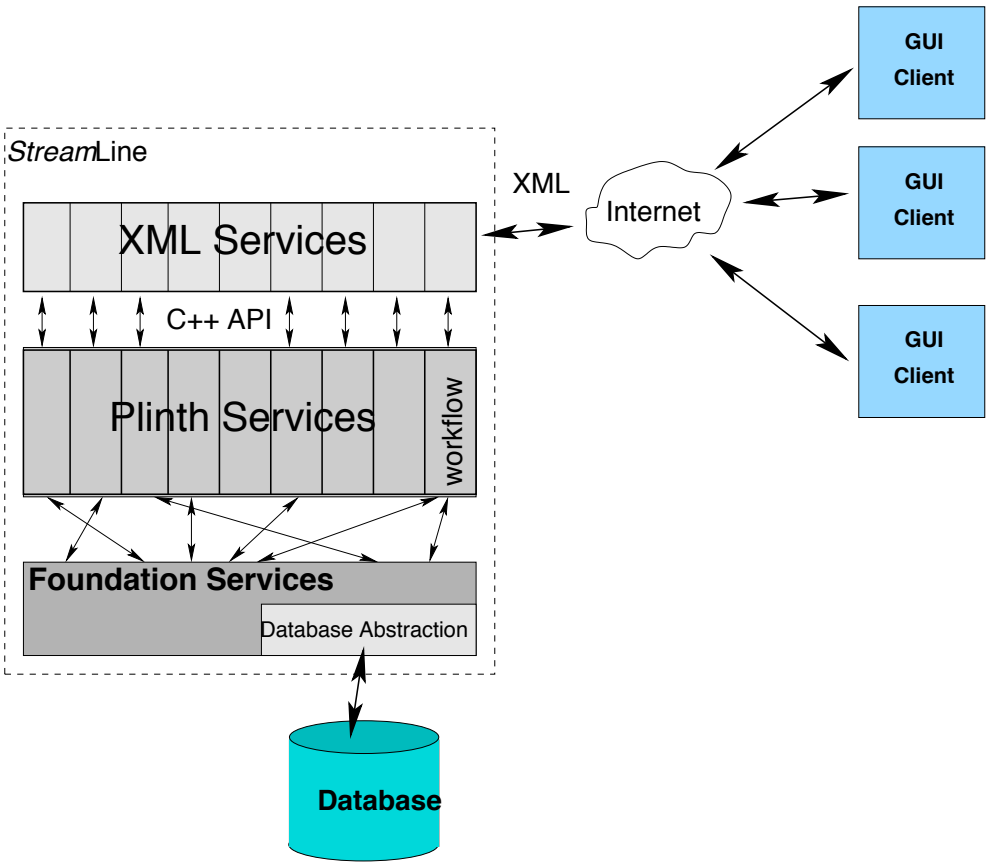


Figure 8.1: *Streamline Overview*

As can be seen from Fig. 8.1, the architecture of *Stream*Line is capable of supporting, through its extensible architecture, sets of services beyond mere workflow. These services are supplied via functional modules and the workflow-specific services include the production of goal schemas described in Chapter 5, together with a service to automatically derive a corresponding skeleton workflow schema from any particular goal schema.

8.2 Features

*Stream*Line 's services are shown in Fig. 8.3 and include support for the creation, deletion and modification of organisational domains, users, user groups, roles, resources, resource groups, goals, workflow processes and process instances, process activities and activity instances, workflow tasks, workitems, worklists. Those services that have yet to be implemented are shaded.

Client applications invoke commands to the *Stream*Line workflow engine by sending command requests expressed in XML over a TCP/IP connection. State is maintained on the client via a sessionID, returned by the server in response to an authenticated login request.

There is a direct C++ API that supports most of the functionality of the W*f*MC 's Interfaces 1,2 & 3. The W*f*MC 's Interfaces are defined in the Reference Model [Wor95] and illustrated in Fig.8.2.

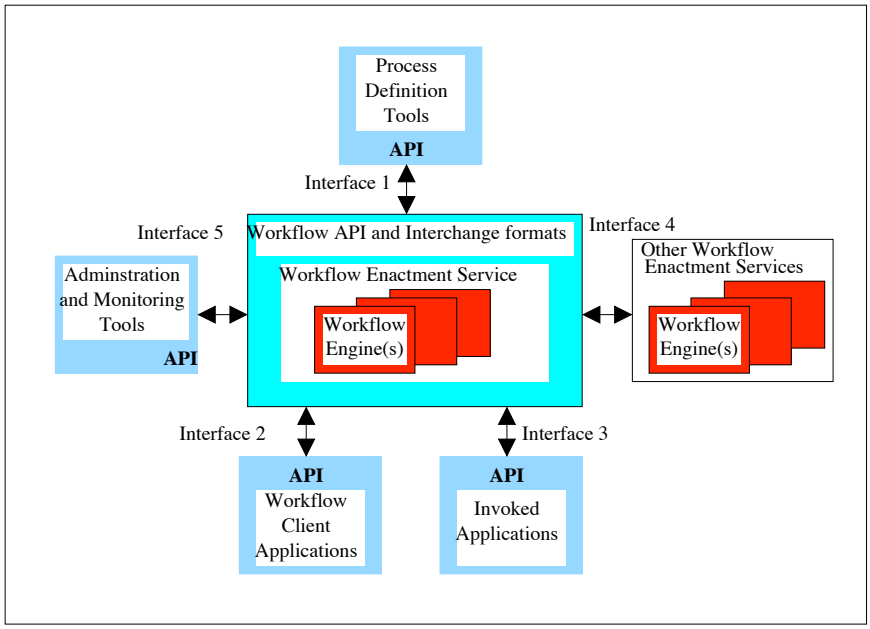


Figure 8.2: *The Workflow Management Coalition's Reference Model*

This API extends the functionality of those interfaces to support the establishment of goals and goal schemas, the automatic generation of a workflow schema from a goal hierarchy, and the ability to interactively and dynamically alter running process instances. The API has been documented using doxygen[3] and represented in both HTML and PDF formats.

*Stream*Line 's modular design means that it not only can be deployed in the traditional W*f*MS sense for providing services to create, edit and enact workflow process schemas and manage worklists and workitem assignment, but can also be used as the basis of a dedicated server for a range of more specific services beyond mere workflow.

Runtime configuration, dynamic loading of functional modules, caching of data for performance reasons, database independence through an abstraction layer and POSIX compliance, are all key features of a robust application framework that allows *Stream*Line to be deployed in a range of environments.

8.3 Implementation

The *Stream*Line workflow server consists of some 80,000 lines of C++ code and has been developed and tested on a Linux (SuSE 8.1) platform. Process and activity schema and instance data are cached in memory, as well as stored in the database. Process schemas are built by clients issuing commands to add/remove/alter activities as required. Schemas can be "cloned" from exisiting schemas already stored in the server's repository.

The author has also written a Java[4]-based GUI client to accompany the *Stream*Line server. The client has been tested on Linux, Microsoft Windows,

[3]doxygen is a documentation system for C++, C, Java, Objective-C, IDL, etc. and licensed under the GNU General Public Licence - see http:www.doxygen.org

[4]Java 2 Standard Edition v1.4.2

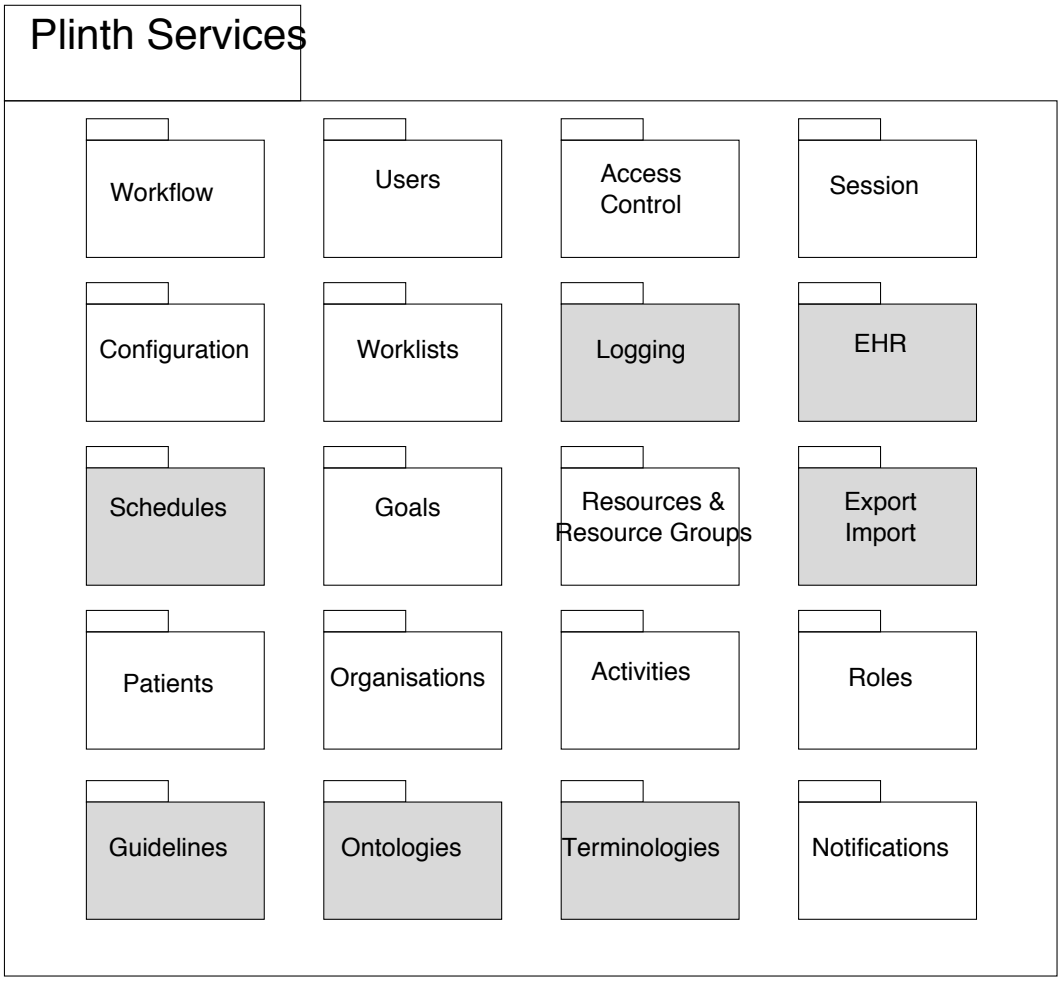


Figure 8.3: *Streamline Services*

and Sun platforms. Apart from having dedicated, tailored screens for many functions as described later, the client can execute any *Stream*Line command in its *Administration* pane, and does so by prompting the user for required and optional arguments. The command execution interface dynamically configures itself based on the number and nature of the command arguments. This information is obtained by querying the server for command parameter details. The advantage of such a facility is that changes and additions to the server can be undertaken and tested without any additional client-side coding. There are also specific in-

terface functions provided for supporting users, activites, processes, goals. There is also a facility that allows users to view the goal hierarchy, prior to invoking the command to automatically generate a corresponding workflow schema. Such a goal view is illustrated in Fig. 8.4

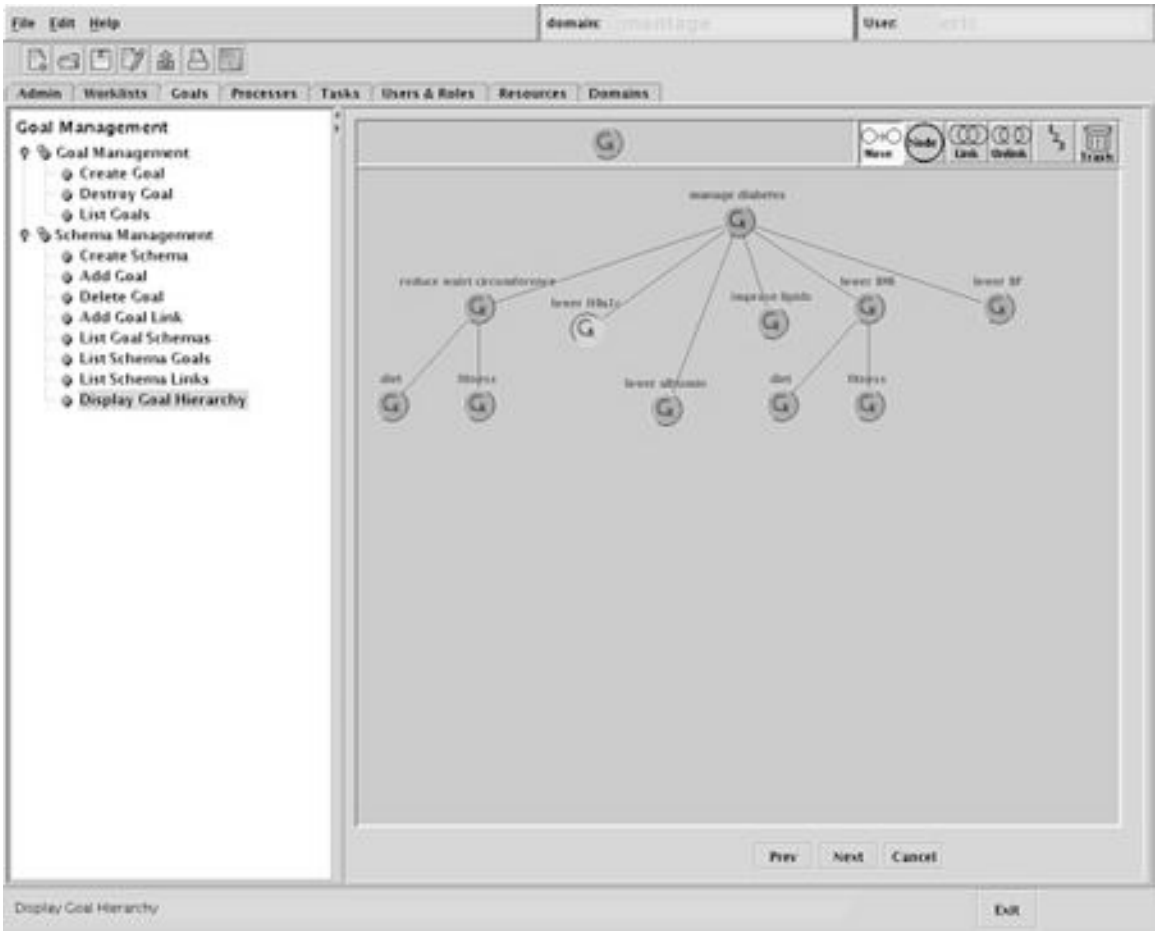


Figure 8.4: *StreamLine 's Goal View of NIDDM workflow*

Different aspects of *Stream*Line 's functionality can be accessed through the client's tabbed panes. Thus, users may flip from a goal view to a corresponding process view and back. Other aspects are tasks, resources, users and roles, worklists, and organisational constructs (hierarchical domains).

8.3.1 Components

*Stream*Line is built from modules, each of which supplies a set of one or more services for an aspect of the application. As previously mentioned, these services extend those specified in the W*f*MC's Interfaces 1,2 and 3. Interface 1 is a specification for the exchange of Process Definitions. This specification is strictly structural, rather than an API, *i.e.* there is no provision for constructing part or all of a process programatically.

*Stream*Line takes a different approach to that of the W*f*MC 's Interface 1. Since most workflow schemas will be constructed dynamically, and will most likely require run-time modifications, *Stream*Line provides a programmatic implementation to support process definition. At programming level, this interface (C++ function calls) adopts the syntactic style of W*f*MC 's Interfaces 2 and 3 with calls for the form *error = workflow_command(input args, output args).* At the service level, these are translated into XML encoded commands whose service element value matches services expressed hierarchically as shown in Appendix A - StreamLine Commands.

Processes

To build a workflow schema, firstly an empty schema is created, to which activities may be added (command `process.definition.activity.add`). These activities are based on tasks which must already exist (*i.e.* they have been specified and added to an appropriate task library), and be available to the organisation constructing the workflow. Activities are either atomic, or compound, in which case the activity represents a subworkflow, *i.e.* another process invoked to perform a subset of tasks of the master process. Processes would normally be expected to map to a specific goal, and have the associated *assess* and *alter* tasks described in previous chapters. However, *Stream*Line also supports the

traditional (non-goal-oriented) approach to constructing and executing workflow schemas equally well.

Transitions

Once activities are added to the workflow schema, they can be linked together by transitions, using the `process.definition.trans.add` command. The reader should recall from earlier discussion that *transition* in this sense, represents the transfer of workflow control from one activity to the next (as per the W*f*MC 's definition), rather than the Petri-Net sense of a transition mapping to a workflow activity.

Tasks

Tasks conform to types, and have associated interfaces. Task types include routing constructs, such as AND-SPLIT, AND-JOIN, as well as MANUALly assigned tasks, DELAY, START and END tasks. The behaviour of the *Stream*Line workflow engine, when an activity is encountered, is determined by the associated task type.

The current set of built-in task types are shown in table 8.1.

Activity Instance States

The W*f*MC 's activity state model is fairly simplistic, providing only *notRunning, running, suspended, completed, aborted* and *terminated*. In contrast, *Stream*Line provides states NOTSTARTED, RUNNING, SUSPENDING, SUSPENDED, TERMINATING, TERMINATED, ABORTED, and COMPLETED. Beyond these states there are a number of additional pseudo-states that are required, particularly with interaction with the worklist manager. A global model of an activity, as implemented in *Stream*Line , is shown in Fig. 8.5.

Task Type	Description
NOP	No Operation (placemarker)
START	Start of a process or subprocess
END	End of a process or subprocess
AND-SPLIT	Branch into 2 or more parallel flows
AND-JOIN	Synchronise 2 or more parallel flows
OR-SPLIT	Branch into 2 or more alternative flows
OR-JOIN	Merge 2 or more alternative flows
ASSESS	Assess a goal target
PROCESS	Launch a subprocess
ALTER	Modify a downstream process flow
MANUAL	Task to be undertaken by user/role
ALARM	Set a timer for specific date/time
DELAY	Delay for a specific duration
EXTERNAL	Task to be performed by an application
NOTIFY	Send notification to a user/role

Table 8.1: *Built-in StreamLine Task Types*

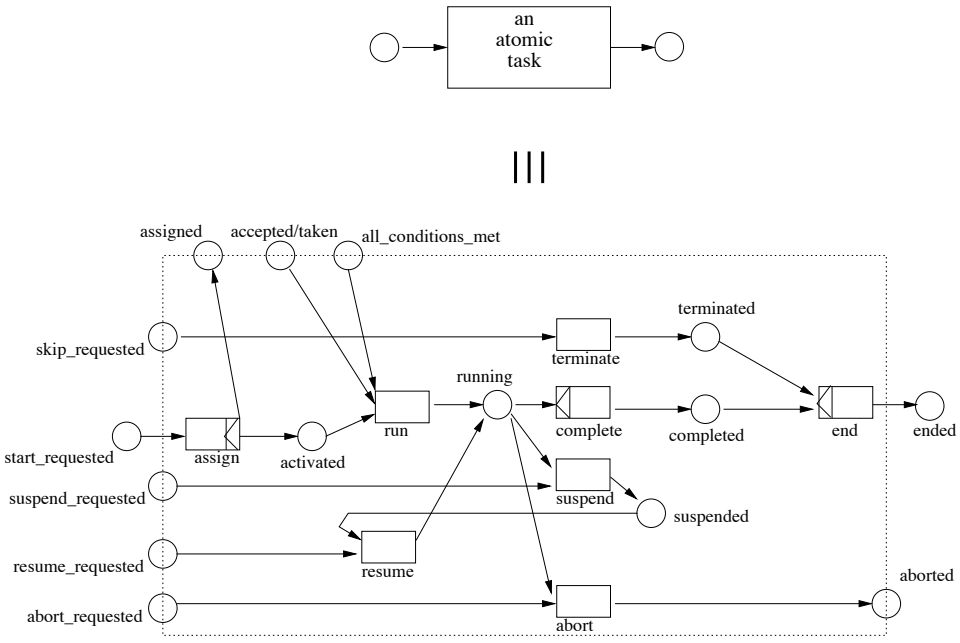


Figure 8.5: *StreamLine 's activity state details*

Expression Parser

An expression parser is used to evaluate activity conditions. The parser is implemented using flex/bison, a GNU variant of the standard unix lex/yacc LR parser.

8.3.2 Goals and Goal Schemas

*Stream*Line supports goals and goal hierarchies, independent of workflow processes. Goals belong to organisational units, and, like task libraries, must exist in a pool available to the organisational unit prior to adding to a particular goal hierarchy. Goal hierarchies or "goal schemas" are created programatically. Initial creation of a goal schema yields an empty schema. Goals are then added to the schema from the goal pool. The hierarchy is then constructed sequentially by linking a source goal to a target goal, using goal schema transitions(edges). This step is repeated for all goals until the hierarchy is complete.

Goal hierarchies can be used to automatically generate corresponding workflow schema templates as described in section 8.4 below.

8.3.3 Services

*Stream*Line version 1.0 provides the following services:-

- **process** - services to support creating and manipulating characteristics of a defined set of tasks and task relationships, including the establishment of task libraries;
- **domain** - services to support defining and manipulating characteristics of organisational units;

- **user** - services to support defining and manipulationg characteristics of a workflow participant capable of performing tasks;
- **group** - services to support defining and manipulating characteristics of a group of users;
- **role** - services to support functional identity(ies) of a user or group of users;
- **resource** - services to support defining and manipulating characteristics of a workflow entity that might be required for tasks;
- **resourcegroup** - services to support a group of resources;
- **goal** - services to support defining and manipulating characteristics of goals and goal hierarchies;
- **activity** - services to support defining and manipulating characteristics of activities;
- **worklist** - services to support tracking and controlling a set of workitems that share a common characteristic;
- **notification** - services to support notification of events to relevant participants and activities.

8.3.4 Commands

*Stream*Line provides commands corresponding to the above services, grouped accordingly. The following tables provide a brief description of the commands of most relevance to the support of the theoretical ideas of Part II. Not all *Stream*Line commands are shown in the following tables. A command always returns a result to the requester. The result is either a structured(XML) resultset

corresponding to the requested service, or an error message. Commands may have zero or more arguments, expressed as a list of keys or key/value pairs.

Goal Commands

Goals are named entities with attributes. They can be constructed, destroyed, linked into hierarchies and these hierarchies used to automatically generate template workflow schemas. Goals can exist independently of goal hierachies, in a similar sense that tasks can exist independently of any process schema. Internally, *Stream*Line refers to goal hierarchies as goal schemas, and goals which exist in a goal schema as schemagoals.

Table 8.2: *Stream*Line Goal Commands

Command	**Explanation**
`goal.create*`	Create a new goal with the supplied attributes.
`goal.describe*`	Describe the existing goal.
`goal.destroy*`	Destroy this goal and all its characteristics.
`goal.exists*`	Check for the existence of the named goal.
`goal.list*`	List all goals known to this organisational unit(domain).
`goal.schema.addgoal*`	Add the named goal into the nominated goal schema. The goal must already exist within the scope of the user's domain.
`goal.schema.addlink*`	Add a link to join two goals in the goal schema.
`goal.schema.clone*`	Create a new goal schema as a copy of an existing schema.
`goal.schema.create*`	Create a new (empty) goal schema.
`goal.schema.list*`	List all the goal schemas.
`goal.schema.listgoals*`	List all the goals in the named schema.

continued next page

Table 8.2: *continued*

Command	Explanation
`goal.schema.listlinks*`	List all the links in the named goal hierarchy.
`goal.schema.root.set*`	Set the named goal as the root goal of the hierarchy.
`goal.schema.unlink*`	Remove a link from the goal hierarchy.
`goal.schema.wf.generate*`	Automatically generate a nested workflow schema from the named goal hierarchy. The details of this generation process are described in section 8.4
`goal.attribute.create*`	Create a new attribute for this goal.
`goal.attribute.destroy*`	Remove this attribute from the named goal.
`goal.attribute.exists*`	Check if the named goal has this attribute.
`goal.attribute.get*`	Get the value of the named attribute for the goal.
`goal.attribute.set*`	Set the value of the named attribute for the goal.
`goal.attributes.list*`	List all the attributes for this goal.

Process Definition Commands

Services to support processes comprise commands which act at the schema level, and those that act at the instance level. Table 8.3 describe the commands used for defining a process schema. The method of constructing a schema programatically consists in a) creating a new empty process schema; b) successively adding activities into the schema; and c) successively linking activities to develop the control flow by creating "transitions" between pairs (source & target) of activities. Process schemas can, as mentioned above, be generated from goal hierarchies. Future versions of *Stream*Line are planned to support importing process schemas formatted in W*f*MC 's XPDL.

Table 8.3: *Stream*Line Process Definition Commands

Command	**Explanation**
`process.definition.activity.add*`	Add an activity(from the pool of available tasks) into the existing process schema.
`process.definition.activity.delete*`	Delete the named activity from the process schema.
`process.definition.activity.list*`	List the activities in the named process schema
`process.definition.attribute.assign*`	Assign a value to the named process attribute
`process.definition.attribute.value.get*`	Get the value of the named process attribute
`process.definition.attributes.list*`	List the attributes associated with the named process schema.
`process.definition.clone*`	Create a new process schema by copying an existing one.
`process.definition.create`	Create a new (empty) process schema.
`process.definition.delete`	Delete the named process schema.
`process.definition.fetch`	Fetch the definition of the named process
`process.definition.list.close*`	Close the currently open list of process schemas
`process.definition.list.open*`	Open the list of process schemas
`process.definition.trans.add*`	Add a transition between a source activity and a target activity of the named process. The source and target activities must already have been added into the workflow process schema.
`process.definition.trans.delete*`	Delete a transition from the process schema

continued next page

Table 8.3: *continued*

Command	Explanation
`process.definition.trans.list*`	List all the transitions, including their source and target activities in the named process schema.

Process Instance Commands

A care plan for a given patient is normally produced by cloning a process schema, or, in the case of a plan based on a goal hierarchy, cloning the base goal hierarchy and generating the template workflow schema therefrom. Activities are then added into the process schema and an instance of the schema started. It is the instance which carries the state information and for which there is a corresponding set of activity instances. Table 8.4 describe most of *Stream*Line 's process instance commands. The W*f*MC in their Interface API specifications provide access to attributes via a process instance attribute list, which operates like a stack. Client applications have access to only the top of the stack, and need to cycle through the stack to obtain a specific value. *Stream*Line supports direct access to process attributes, both at the schema and instance levels. The stack/list operations are provided for compatability with the W*f*MC APIs.

Table 8.4: *Stream*Line Process Instance Commands

Command	Explanation
`process.instance.attribute.assign`	Assign a value to the named attribute for this process instance
`process.instance.attribute.fetch`	Fetch the value of the next attribute in the open list of attributes for the name process instance.

continued next page

Table 8.4: *continued*

Command	**Explanation**
`process.instance.attribute.value.get*`	Get the value of the named attribute belonging to a specific process instance.
`process.instance.attributes.list*`	List the names and values of all attributes of the nominated process instance.
`process.instance.attributes.list.close`	Close the currently open list of attributes belonging to the nominated process instance.
`process.instance.attributes.list.open`	Open the the list of attributes belonging to the nominated process instance.
`process.instance.create`	Create a new process instance, ready for starting.
`process.instance.fetch`	Fetch details regarding a the next process instance in the currently open list of instances of a specific process.
`process.instance.get`	Get details regarding a specific process instance
`process.instance.start`	Start a specific process instance, identified by the ID returned from a prior process.instance.create command.
`process.instance.state.change*`	Set the state of the current process instance to one of the predefined *Stream*Line process states. This is normally done by the Workflow Engine itself, but can be used to issue suspend/resume requests to a running instance.
`process.instance.state.fetch*`	Get the current state of a specific process instance.
`process.instance.terminate`	Terminate the currently running nominated process instance.

continued next page

Table 8.4: *continued*

Command	Explanation
`process.instances.abort`	Abort the nominated process instance.
`process.instances.attribute.assign*`	Assign a value to the named attribute for all instances of a specific process.
`process.instances.list*`	List key information concerning all instances of one or all processes.
`process.instances.list.close`	Close the list of instances of a specific process.
`process.instances.list.open`	Open the list of instances of a specific process.
`process.instances.state.change*`	Assign a new state to all instances of a given process.
`process.instances.terminate*`	Terminate all currently running instances of a specific process.

Activity Instance Commands

Activities are named tasks, together with attributes and associated participants and resources. Activities are added into an existing process schema using the process definition commands specified in Table 8.3. Table 8.5 lists the commands that act upon one or more instances of a specific activity within a specific process. Each activity instance is identified by a unique ID.

Table 8.5: *Stream*Line Activity Instance Commands

Command	Explanation
`activity.instance.attribute.assign`	Assign an attribute to an activity instance.

continued next page

Table 8.5: *continued*

Command	**Explanation**
`activity.instance.attribute.fetch`	Fetch the value of the next attribute in the currently open list of a specific activity instance.
`activity.instance.attribute.value.get*`	Get the value of the named attribute of a specific activity instance.
`activity.instance.attributes.list.close`	Close the attribute list associated with a specific activity instance.
`activity.instance.attributes.list.open`	Open the attribute list associated with a specific activity instance.
`activity.instance.create`	Create an instance of the named activity. This would normally be done automatically by the workflow engine itself, but is included for testing and initiating the start activity of a process or subprocess.
`activity.instance.fetch`	Fetch information pertaining to the next activity instance in the activity instance list.
`activity.instance.get`	Get information pertaining to the identified activity instance of a specific activity.
`activity.instance.start`	Start an instance of the named activity. The Workflow Engine returns the unique ID of the instance.
`activity.instance.state.change*`	Change the state of the named activity instance to the one supplied. Normally invoked internally by the Wf Engine.
`activity.instance.state.fetch*`	Fetch the state of the next next activity instance.

continued next page

Table 8.5: *continued*

Command	Explanation
`activity.instances.attribute.assign*`	Assign a common value to the named attribute, for all instances of a given activity.
`activity.instances.list`	List all the instances pertaining to the named activity.
`activity.instances.list.close`	Close the list of instances pertaining to the named activity.
`activity.instances.list.open`	Open the list of instances pertaining to the named activity. Instance activity information can then be fetched.
`activity.instances.state.change*`	Change the state of all instances of the named activity. Could be used for terminating or aborting instances of an activity.

Task Definition Commands

Tasks describe the kind of work that needs to be undertaken to perform a step in a workflow process. However, they exist independently of any specific process schema. "Taking a patient's blood pressure (BP) using a sphygmomanometer" describes a task common to the detection and monitoring of many conditions. In *Stream*Line , when this task is assembled into a particular workflow process schema, it is referred to as an activity. Thus, commands to set/get the value of attributes, such as the "measured systolic BP" are undertaken on activities and not tasks. Similarly with users and resources - these are allocated to tasks once they become activities in a specific process schema. Thus, the commands that act on tasks form a small set, as shown below in table 8.6.

Table 8.6: *Stream*Line Task Definition Commands

Command	Explanation
`wf.task.definition.create*`	Create a new task, based on a predefined set of task types, as described in Table 8.1
`wf.task.list*`	List the names of all tasks. Note: tasks exist independently of any workflow process schema.
`wf.task.types.list*`	List all fundamental task types, as denoted in Table 8.1

Worklist and Workitem Commands

The following worklist/workitem services provide for interacting with the *Stream*Line workflow engine to obtain or set information regarding worklists and workitems, as well as supporting the acceptance or reassigning of responsibility to undertaken individual items of work.

Table 8.7: *Stream*Line Worklist & Workitem Commands

Command	Explanation
`worklist.close*`	Close the named worklist.
`worklist.create*`	Create a named worklist.
`worklist.list*`	List all available worklists.
`worklist.open*`	Open the named worklist.
`workitem.attribute.assign*`	Assign a value to a workitem attribute.
`workitem.attribute.fetch*`	Fetch the value of the next attribute in the open list of attributes for this workitem.
`workitem.attribute.value.get*`	Get the value of the named attribute for this workitem.
`workitem.attributes.list.close*`	Close the list of attributes of the given workitem.

continued next page

Table 8.7: *continued*

Command	Explanation
`workitem.attributes.list.open*`	Open the list of attributes of the given workitem, for access.
`workitem.complete*`	Flag the workitem (and therefore also the associated activity) as complete.
`workitem.create*`	Create a new workitem. Normally this is performed by the workflow engine.
`workitem.execute*`	Accept responsibility for undertaking to perform the activity denoted by the nominated workitem.
`workitem.fetch*`	Fetch information pertaining to the next workitem from the named worklist.
`workitem.get*`	Get details of a specific workitem.
`workitem.list*`	List all workitems in a given worklist.
`workitem.reassign*`	Reassign a workitem to the specified participant.

8.3.5 Workflow Management Coalition Compatability

The following excerpt from the *Stream*Line Reference Manual illustrates the implementation of a standard W*f*MC WAPI[5] call, as implemented in C++. In this case, **SL** is the *Stream*Line namespace, **Workflow** denotes the singleton class implementing workflow services, and **WMAssignActivityInstanceAttribute** is the C API call as specified in Interface 2.

```
Workflow::WMTErrRetType SL::Workflow::WMAssignActivityInstanceAttribute(
in WMTPSessionHandle psession_handle,
in WMTPProcDefID pproc_def_id,
in WMTPActivityInstID pactivity_inst_id,
in WMTPAttrName pattribute_name,
in WMTInt32 attribute_type,
in WMTInt32 attribute_length,
in WMTPText
pattribute_value )
```

[5]Workflow Application Programming Interface

This command tells the WFM Engine to assign an attribute, to change an attribute or to change the value of an attribute of the activity instance within a named process definition.

This command changes the value of the attributes of a activity instance. These attributes of activity instances are of the kind called Process Control and Process Relevant Data. These attributes are specified as quadruplets of name, type, length and value.

Parameters:

psession_handle	Pointer to a structure containing information about the context for this action.
pproc_inst_id	Pointer to a structure containing the unique process instance ID.
pactivity_inst_id	Pointer to a structure containing the activity instance identification for which the attribute will be assigned.
pattribute_name	Pointer to the name of the attribute.
attribute_type	Type of the attribute.
attribute_length	Length of the attribute value.
pattribute_value	Pointer to a buffer area provided by the client application where the attribute value will be placed.

Wherever *Stream*Line extends the W*f*MC 's API, it adopts similar naming conventions and method arguments, replacing the **WM** prefix with a **WME**. Thus, an example of a *Stream*Line command not supported by W*f*MC 's WAPI is **WMEListProcessInstances** as shown below:-

```
Workflow::WMTErrRetType SL::Workflow::WMEListProcessInstances (
in WMTPSessionHandle psession_handle,
in WMTPProcDefID pproc_def_id,
out ProcInstances & processInstances )
```

This command lists all process instances of a particular process schema.

Parameters:

psession_handle	Pointer to a structure containing information about the context for this action.
pproc_def_id	Pointer to a structure containing a unique process definition ID.
processInstances	Reference to a structure contaning details of process instances.

In the above example, the argument *processInstances* is a reference to a structure which is also not defined in the WAPI, but is an extension provided by *Stream*Line supporting additional information such as the creation times of the instances. Thus, notwithstanding its considerable general application functionality potential, from a workflow functionality perspective, *Stream*Line essentially

provides a super-set of that defined by the W*f*MC in its Interface 1, 2 and 3 specifications.

8.4 Operation

*Stream*Line runs as a multithreaded server process, listening for TCP/IP-based requests. Users authenticate themselves to the server process and obtain a randomly generated session ID. Subsequent requests for services utilise this session ID, and pass appropriate service parameter information as required, in their XML input stream.

A key service is the generation of a template workflow schema from a goal hierarchy, which follows these steps:-

1. finding the root goal in the goal hierarchy;
2. creating a wf schema based on the name of this goal, suffixed by the unique ID of the goal schema ;
3. traversing the child goals, and creating a subworkflow for each non-leaf goal, together with 'assess' and 'alter' tasks.
4. creating assess, alter and do tasks for each leaf goal.

For a simplified diabetes goal hierarchy comprising only 3 level 1 goals, and a total of only 4 leaf goals, the result of autogenerating corresponding workflow schema templates using *Stream*Line 's `goal.schema.wf.generate` command yields the 5 process schemas shown in fig.8.6.

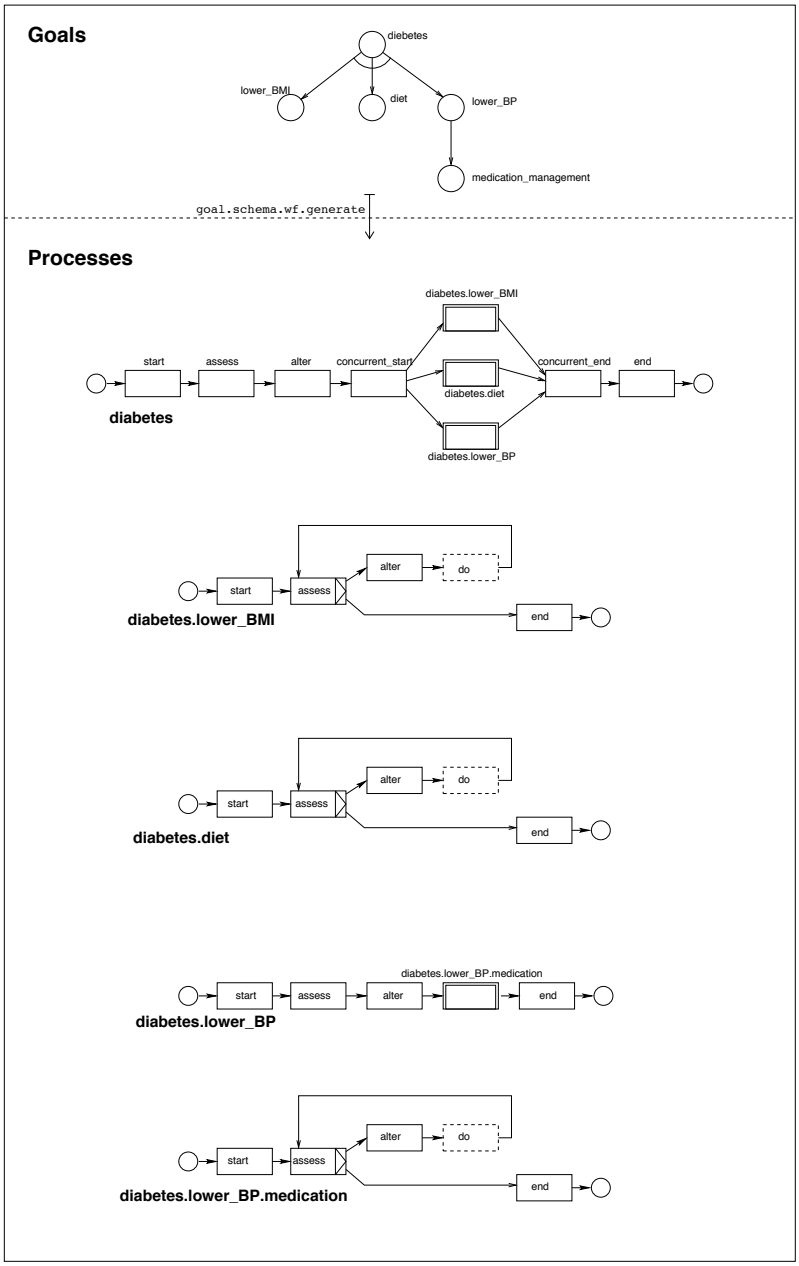


Figure 8.6: *Generation of Wf schema from goal hierarchy for diabetes(simplified) using StreamLine 's* `goal.schema.wf.generate` *command.*

8.5 Functional Requirements not Supported

Not all of the functional requirements of a W*f*MS that were described in Table 4.1 are supported by the current version of *Stream*Line . This section discusses some of these requirements and explains the approach that could be adopted when extending *Stream*Line to support them.

The current incarnation of *Stream*Line , version 1.0, provides no automated features to support activity crediting. The only support provided is through the use of goal hierarchies, and the potential use of a hierarchical task naming convention when creating task libraries. Manual runtime alteration of workflow schemas can be used to avoid conflicting or redundant tasks. As pointed out in the discussion on activity crediting in Chapter 7, clinicians will always need to be the final arbiter in deciding whether any specific task can be dropped from a workflow process, in any case.

There are currently no services to support scheduling of activities to occur at nominated dates and times. The only way this can be achieved in the current version is via DELAY activities, wherein the target date and time of a desired activity needs to be converted into a offset from a precursor activity and that offset entered as a parameter to the inbuilt delay activity. This may mean, that for activities that must occur on a specific date, that a delay relative to the process start be modelled into the schema.

Other significant services planned, but not yet incorporated into *Stream*Line include support for terminology services and support for accessing patient data records. The Australian Electronic Health Record (EHR) initiative "*Health*Connect" [Aus04], planned for initial implementations to commence at the end of 2005, will most likely publish a standard API for accessing shared EHRs in the near future.

8.6 Summary

This chapter introduced the author's prototype W*f*MS *Stream*Line and illustrated its features and potential for supporting the author's extensions outlined in part II of this thesis. The important features of *Stream*Line are its:-

- flexibility
- compatability with interfaces and APIs defined by the W*f*MC .
- internet deployability
- support for goal hierarchies
- autogeneration of process schemas from goal hierarchies
- support for resources and resource groups
- support for organisational domains, user groups and roles
- extensibility to provide additional services. Such services can be encapsulated modules which can be configured for inclusion in specific applications as required.
- process schemas that can be built up and modified programmatically based on tasks from a task library.

CHAPTER 9

Case Study - Management of Diabetes Mellitus.

" The world is disgracefully managed, one hardly knows to whom to complain."

Ronald Firbank, Mrs Shamefoot on Vainglory, 1915.

9.1 Introduction

This chapter examines the preceding theoretical and prototyped enhancements to standard W*f*MSs in the context of the current Australian practice of managing patients with Type II diabetes. It firstly outlines those current practices and some important issues faced by healthcare providers. The chapter then looks at how a future workflow-enabled Clinical Information System can improve care and care efficiency through its support for care plan tailoring and enactment. It furthermore looks at additional obstacles that may need to be overcome in order for healthcare providers to adopt and adapt to these new systems.

9.2 The Problem

Non-insulin-dependent Diabetes Mellitus (NIDDM) is a significant problem in the developed world and is continuing to grow in significance at an alarming

rate in many of those countries. It is considered by many people a disease of the affluent, but even in the most affluent of countries, it is often the poorest in the community that present the highest incidence. The case study was designed to ascertain how well a WfMS , enhanced with the facilities of two-tier goal-process modelling, self-modification and activity crediting could support care plan creation and enactment for patients suffering from NIDDM.

9.3 Methodology

The Australian guidelines for Diabetes were used as the basis for testing *Stream*Line's ability to provide flexible care plans. The guidelines articulate explicit goal targets for NIDDM patients, which are shown in table 9.1.

These goals were entered into a goal hierarchy in *Stream*Line and a workflow schema generated therefrom, using *Stream*Line 's inbuilt schema generator. The workflow schema comprised hierarchical subworkflow schemas as described in chapter 5, whereby each subworkflow corresponded to the goals outlined above.

9.3.1 GP Focus Group

*Stream*Line 's underlying goal-focused philosophy was evaluated by GPs through a focus group composed of care planning experts from the Adelaide Western Division of General Practice (AWDGP). AWDGP has undertaken a number of projects to improve chronic disease management, and has been looking for and assessing various software innovations to support and improve care plan management for a number of years.

Goal Aspect	**Target**
HbA1c	7%
Blood Glucose Level	fasting 3.5 - 6.0 mmol/L post-prandial 3.5 - 8.0 mmol/L
Blood pressure	$< 130/85$ ($< 125/75$ in people with known microalbuminuria or proteinturea)
Lipids	total cholesterol < 4.0mmol/L triglycerides < 2.0 mmol/L HDL > 1.0 mmol/L LDL < 2.5 mmol/L
Albumin	$< 20\mu$/min timed overnight collection < 20 mg/L spot collection < 3.5 mg/mmol women, < 2.5 mg/mmol men albumin creatine ratio
Weight	Body Mass Index ≤ 25
Waist circumference	advise no further weight gain: men ≥ 95 cm, women ≥ 80 cm. advise weight loss: men ≥ 100cm, women ≥ 90cm (based on European populations)
Smoking	complete cessation
Exercise	at least 30mins walking 5 days/week

Table 9.1: *Australian NIDDM goal targets*

9.4 Focus Group Findings

Key themes were extracted from transcripts of the focus group and returned to the GPs for validation, interpretation and comment. The principal issues and findings are described below.

9.4.1 Care Planning is almost universally ad hoc

Although Medical Benefits Scheme (MBS) incentive payments were introduced some years ago in order to encourage the adoption of care planning for chronic disease management throughout general practice in Australia, GPs report that the use of financially supported care planning (via the Mecial Benefits Scheme) is

now virtually non-existent. An MBS-stipulated requirement for "intercourse with a consultant" at the time care plans are being drafted, is seen as the primary (of a number) of obstacles to care plan adoption. Other obstacles include bureaucratic inflexibility, particularly regarding timing of consultations. However, a number of GPs do initiate care plans for patients, albeit without any financial reimbursement. Despite the MBS-stipulated requirements, there is currently no standardisation of care plans within or across divisions, although AWDGP and some other divisions see a need for standardisation.

9.4.2 Care Plans are about Shared Care

Care Plans are usually only formulated when more than one health service provider is involved in the ongoing care of the patient (The MBS reimbursement requires the involvement of two or more health care providers other than the GP). Pharmacists and endocrinologists are seen as key contributers to diabetes care planning. Care planning artefacts (paper or electronic proformas) are seen as potentially providing a useful tool for exchange of patient information.

9.4.3 Care Plan Format is important.

A care plan should consist of:

- a clear statement of the objectives (rationale, including risk of complications)
- the solution - tasks required to achieve the objectives
- schedule of tasks, participants, review timetable and may be augmented with

- tool(s) to monitor and collect data in a format consistent with current guidelines.

GPs see that an additional benefit from consistent care planning can arise from public health monitoring/aggregation of de-identified data to inform Divisions of General Practice on their performance with respect to diabetes management.

Monitoring of progress against goal targets is currently difficult, because of lack of feedback from allied health providers, especially given the long time intervals associated with visits to some providers. For instance, it was not generally known what records podiatrists might keep regarding patients sent to them by GPs for diabetes-related care. There was rarely any feedback from the podiatrist to the GP following the patient visit(s). GPs tended to judge from the state of the patient's feet, what tasks might have been performed, but in some cases it is difficult to judge the state of the patient relative to the previous visit to the GP, and patients could deteriorate significantly in between successive visits to the GP. Some proformas for capturing information regarding visits to allied health practitioners only capture "whether the patient has visited xxx" not "what actions the xxx took" or "what assessments were made", and rarely do they capture whether the patient's condition has improved, remained stable or deteriorated.

The diversity of layout and content of guidelines and data capture proformas is seen as an issue that needs addressing. Some GPs might use a dedicated diabetes module supplied with their Clinical Information System (CIS). Others might simply record the data in progress notes within the same package, due to the cumbersome nature of the user interface. Some GPs use tear-off paper slips provided by pharmaceutical and other vendors. This lack of consistency is exacerbated by the fact that guidelines, including goal targets, change fairly frequently.

9.4.4 Referral Issues

Some GPs use a standard referral template (part of their CIS) to refer diabetic patients to specialists, irrespective of the specialist type. This implies each specialist receives the same information about the patient. Other GPs provide information specific to the referree. Some patients see specialists (e.g. podiatrist) without a referral, particularly once an inital relationship has been established. The Queen Elizabeth Hospital (TQEH) Diabetes Assessment Unit is seen as a very good model. They have a well-defined process which coordinates various specialities, and they keep consistent records resulting in less duplication of tests.

9.4.5 Report Issues

As with referrals there is no standardisation of process or format of reports from specialists back to GPs. In fact, there is currently very little information flow back to the GP. GPs often remain unsure if some aspect of the patient or patient condition has been checked, or recorded, or whether the relevant information has simply not been sent. An exception was TQEH Diabetes Assessment Unit who collate all assessments into one report to return to the GP.

9.4.6 Care Plan Flexibility

Care Plans are seen as evolving documents. The MBS defines reimbursement rates based on 3 fixed levels of care plan service only, each attract a different payment. Furthermore, the MBS only supports reviewing a care plan at minimum of 6-monthly intervals. These artificial constraints compromise a WfMS 's ability to provide the required flexibility to meet many individual patients' changing requirements - this complicates the flexibility issue somewhat. Comorbidities are usually dealt with by adding to the existing care plan. Common comorbidities

with Type 2 Diabetes are: cardiovascular disease, depression, sexual dysfunction, arthritis and thyroid problems. Comorbidities can lead to significant medication management problems. A home medication review by a pharmacist is seen as highly desirable for some patients. The goal of a medication review is seen as helping to "optimise the patient's overall health management" rather than managing/curing a single symptom. Other aims to be taken into account are minimizing the number of concurrent drugs, and utilising the most cost-effective drugs.

9.4.7 Clinical Guidelines

Currently GPs feel that there is too much variability in both content and format of guidelines and there are too many sources of guidelines for a given condition. These sources include the RACGP, Diabetes Australia,Clinical Guidelines university research units, WHO, many GP Divisions, every state health department, etc. Guidelines need to be concise, consistent and unambiguous - *e.g.* what is a "low carbohydrate diet"?

9.4.8 Focus Group Conclusions:

1. Care Plans should start with goals. Goals lead to a process view composed of individual tasks. Any data to be collected should be associated with the corresponding task and made available to other providers participating in the care plan.

2. Care Plans need to be flexible and cater for different and changing patient conditions.

3. Any system or tool provided to clinicians to support care planning must match the workflow of the clinician. Tools with cumbersome interfaces, or

bureaucratic requirements or inefficiences will simply not be adopted by busy clinicians.

9.5 Summary

This chapter examined the theoretical proposals to improve W*f*MSs that were detailed in Chapters 5, 6 and 7, together with the prototype W*f*MS of Chapter 8, *Stream*Line , in the context of the current Australian practice of managing patients with Type II diabetes. Workshops undertaken by the author sought the views of practioners involved with, and committed to care planning and examined current practices and issues faced by those and other healthcare providers. The primary findings indicated that although there is support for the adoption of workflow-enabled care planning tools, there are still technical and political challenges to overcome. The author's approach of linking activities to goals and making the associated goal targets a focus of care plans in a way that provides considerable runtime flexibility was vindicated.

CHAPTER 10

Conclusions

> *" In the field is neither wind nor cry, Nor a lonely white willow. We will go out with the last star to search for our grandfather's truth... The centuries will depart in sequence. And it's not for us to understand even the grass."*
>
> Andrei Platonov, 19xx.

The author has presented the requirements, and both a theoretical and practical framework to support the adoption of W*f*MSs into the arena of chronic disease management. In so doing, the author concentrated on new modelling approaches in a few key areas, primarily addressing the requirement for flexible runtime adaptation of processes to match individual patient goals. This chapter summarises the findings, issues and possible future research directions arising from the author's experiences with these modelling approaches.

10.1 Thesis Summary

An analysis of the problem space was presented in chapter 2 which identified some seven aspects of health care that place significant obstacles in the path of modelling and supporting the enactment of processes. Different types of computing systems and applications in health care were described to illustrate where

W*f*MSs might sit in the spectrum of other approaches already being trialed, such as computer interpretable guidelines, electronic decision support systems, care planning and electronic health records.

This health-specific analysis was followed, in chapter 3, by an analysis of current workflow approaches in both health care and other domains, particularly where issues of runtime flexibility have been addressed. Related research into goal-oriented process modelling, active object databases, database schema evolution and versioning, various approaches to dynamic/flexible workflow modelling and finally web services composition was visited to establish the space for the author's contribution.

The central body of work was presented in Part II, and comprised an overview of the author's ideas, concepts and approach (Chapter 4), followed by analyses of **goal-focused workflow** (Chapter 5), **self-modifying workflow schemas** (Chapter 6), and **activity crediting** to reduce redundancy (Chaper 7).

An understanding of the concepts of *state* and *activity target objects* was deemed important, when trying to analyse workflows in a complex domain such as healthcare. The associated analysis led to a separation of activity target objects based on *workflow state*, *patient state*, *environment state*, and *resource state*. Thirty-three requirements for workflow enabled systems to support chronic disease management were enunciated, and the ensuing chapters aimed at addressing some of those requirements.

The chapter on goal-focused workflow schemas introduced both the philosophy, borrowed from the domain of Software Requirements Engineering, as well as an implementation methodology devised by the author. This implementation methodology centred around a method for describing goals in the form of a goal hierarchy, and using hierarchical decomposition to map the goal hierarchy to a set of nested subworkflow schemas. Each workflow schema provides a specific task

to **assess** if the corresponding goal target has been achieved, and a task to **alter** components of the workflow to either better achieve the goal, or to change the target parameters. The nested set of subworkflows, corresponding to a specific goal hierarchy, becomes a *template* workflow description, which then needs completing by a clinician. This completion is undertaken partially prior to the workflow being instantiated for a given patient, by inserting typical tasks from a task library. This generates a "care plan" for a patient. The care plan is then enacted over the course of the patient's treatment, by providing modifications to parameters for assessments and interventions, and in some cases, inserting, modifying or dropping specific activities.

In order to support dynamic modification of care plans (workflow schemas), the author discussed, in Chapter 5, those operators that are required to modify a given schema. When these operators are made available to the **alter** task in a given schema, they provide the functionality described by the author as *self-modifying workflow*, since both the requirement for, and the ability to individualise the schema, are built into the schema itself.

Mechanisms for eliminating redundant activities as well as partially crediting the work achieved by previous activities in a given workflow instance were described in Chapter 7. The contributions of that chapter built on the use of the two-tier methodology for representing and viewing business processes, based on separate goal and process views, to develop a two-phase methodology for firstly discovering, and secondly crediting selected components of the overall business process. The first phase is called *candidate discovery*, and the second phase *component crediting*. Two classes of crediting were identified, namely *permanent crediting* and *temporary crediting*. The chapter also described a set of still unresolved issues that need to be addressed for activity crediting to be supported effectively in the clinical setting. Not least of these is the difficulty in describing tasks in

such a way that their pre- and post-conditions can be compared efficiently by computational means. Placing tasks in a hierarchical task ontology is seen as a precursor to providing these descriptions.

Part III firstly introduced the author's prototype W*f*MS , *Stream*Line , which expands on the functionality of the W*f*MC 's Interfaces 1,2 & 3 and provides support for self-modifying, goal-focussed workflow schemas. *Stream*Line already provides for the autogeneration of template workflow schemas from a goal hierarchy. *Stream*Line 's design, current implementation and operation were described in Chapter 8.

In Chapter 9, the testing of the author's ideas for managing diabetes was described. This entailed consultations with local doctors experienced in, and dedicated to improving care planning for chronic conditions in general, and diabetes in particular. Their concerns with current practice, and a requirement for managing patient compliance, the provision of timely and non-redundant services and efficient patient self-management according to prescribed guideline targets, vindicated the author's commitment to introducing self-modifying, goal-focussed W*f*MSs .

10.2 Future Directions

Future work is proposed which would develop models to address the levels of change required to support the goal-focussed approach outlined in Chapter 5. The ultimate aim would be the production of good user interfaces to support the underpinning, comprehensive language that allows future Workflow Management System implementations to have business processes modelled at the goal level as well as the process level. This would be done through achievement constraints, separate goal assessment and choice/change tasks, and a set of rules governing

the range of legitimate choice and change operations that are permited. Practical validation of these models is planned to be undertaken in conjunction with the South Australian Department of Health (formerly Human Services). Validation is intended to be undertaken, in the first instance, via the author's prototype workflow engine *Stream*Line to support the interorganisational requirements of chronic disease management within the Department.

The nature of goals is an area where further research could bear fruit. In analysing current work practices for the management of chronic conditions, it became evident that there are a class of tasks undertaken that are not aimed at achieving goals of direct patient care *per se*, but instead, play a secondary or coordinating role. Such an example is that of a "medication review". Medications are already prescribed and administered by other tasks for direct patient benefit. The medication review analyses and coordinates the results of those individual tasks both for potential risk to the patient, and for efficient use of medications. The latter corresponds, to some extent, to the author's activity crediting notion. The former, that of risk mitigation, is a class of task for which no direct goal would be identified when performing a hierarchical decomposition of a goal hierarchy for direct patient care. The author postulates that once a goal hierarchy has been identified, it should be reviewed with the express intention of enunciating coordinating and risk management goals. The workflow schema would then be derived from a combined goal hierarchy that included both direct and indirect goals.

Research that investigates the interaction between activities carried out by different health care providers could lead to automatic methods for reducing adverse events and redundant investigations. The author's research presented in Chapter 7 concluded that much better methods for the characterisation of tasks is required. Such characterisation needs to encompass formal expressions of a task's

purpose, validity time, side effects, pre- and post-conditions, and quality of services. Future research in the field of web services orchestration and choreography could inform this area.

From a practical perspective, activity crediting relies on participants' understanding of the entire care process or workflow schema for each patient. Good process monitoring tools are essential for this understanding, and for each activity, or change in workflow, a snapshot of the case, including rationale for any changes, needs to be available to all relevant participants. An annotated runtime view of the goal hierarchy can be presented to clinicians as a synoptic view of the case, showing which goals have been achieved, supplemented with times and durations. Further work needs to be undertaken on user interfaces that support evolving views of complex processes - views which link to historical and predictive data about the patient on the one hand, as well as clinical guideline knowledge on the other.

10.3 Final Message

Although W*f*MSs and workflow modelling theory have made some progress towards meeting the flexibility required in health care, there is still a need to look afresh at how best to meet the challenges of complexity, variability, fuzziness and non-determinism of the health domain. This thesis has addressed some of the major issues faced when trying to apply Workflow Management Systems technology to chronic disease management, and has documented some of the author's theoretical and practical approaches to workflow modelling that can assist this process.

In canvasing some shortcomings of the application of traditional workflow modelling to chronic disease management, and presenting some potential impro-

vements, this thesis goes only a tiny way to surmounting the significant obstacles to the incorporation of WfMSs into clinical information systems. The reader should not be left with the impression that usable solutions are just around the corner - the journey is still only beginning.

-ooOOOoo-

APPENDIX

Appendix A - StreamLine Commands

Here is a list of relevant StreamLine Commands that can be executed by sending an XML SOAP-like message to the StreamLine server as described in chapter 8. The commands are listed in alphabetical order, where the first component of the name usually corresponds to the service category. Note also, that these are only the names of commands. Each command takes zero or more arguments, which are not shown in the list. For each command, there is a corresponding C++ API call, whose name generally reflects those names found in the W*f*MC 's Interface definitions.

```
activity.instance.attribute.assign
activity.instance.attribute.fetch
activity.instance.attribute.value.get
activity.instance.attributes.list.close
activity.instance.attributes.list.open
activity.instance.create
activity.instance.fetch
activity.instance.get
activity.instance.start
activity.instance.state.change
activity.instance.state.fetch
activity.instance.states.list.close
activity.instance.states.list.open
activity.instances.attribute.assign
activity.instances.list
activity.instances.list.close
activity.instances.list.open
activity.instances.state.change
```

```
activity.start

domain.create
domain.destroy
domain.exists
domain.list
domain.modify

goal.create
goal.describe
goal.destroy
goal.exists
goal.list
goal.modify
goal.schema.addgoal
goal.schema.addlink
goal.schema.create
goal.schema.list
goal.schema.listgoals
goal.schema.listlinks
goal.schema.root.set
goal.schema.unlink
goal.schema.wf.generate
goal.attribute.create
goal.attribute.destroy
goal.attribute.exists
goal.attribute.get
goal.attribute.set
goal.attributes.list

group.create
group.describe
group.destroy
group.exists
group.list
group.modify

notification.create
notification.describe
notification.destroy
notification.exists
```

```
notification.list

process.definition.activity.add
process.definition.activity.change
process.definition.activity.delete
process.definition.activity.list
process.definition.attribute.assign
process.definition.attribute.value.get
process.definition.attributes.list
process.definition.clone
process.definition.close
process.definition.create
process.definition.delete
process.definition.fetch
process.definition.list.close
process.definition.list.open
process.definition.open
process.definition.state.change
process.definition.state.fetch
process.definition.states.list.close
process.definition.trans.add
process.definition.trans.change
process.definition.trans.delete
process.definition.trans.list

process.instance.attribute.assign
process.instance.attribute.fetch
process.instance.attribute.value.get
process.instance.attributes.list
process.instance.attributes.list.close
process.instance.attributes.list.open
process.instance.create
process.instance.fetch
process.instance.get
process.instance.start
process.instance.state.change
process.instance.state.fetch
process.instance.states.list.close
process.instance.states.list.open
process.instance.terminate
process.instances.abort
```

```
process.instances.attribute.assign
process.instances.list
process.instances.list.close
process.instances.list.open
process.instances.state.change
process.instances.terminate

process.list

resource.create
resource.describe
resource.destroy
resource.exists
resource.modify
resource.group.add
resource.group.exists
resource.group.list
resource.group.remove
resource.list
resource.attribute.create
resource.attribute.destroy
resource.attribute.exists
resource.attribute.get
resource.attribute.list
resource.attribute.set

role.can
role.create
role.describe
role.destroy
role.exists
role.list
role.modify
role.rule.add
role.rule.exists
role.rule.remove
role.rule.replace

user.create
user.describe
user.destroy
```

```
user.exists
user.list
user.modify
user.role.add
user.role.exists
user.role.list
user.role.remove
user.rule.list

wf.connect
wf.definition.close
wf.definition.open
wf.disconnect
wf.entities.list.close
wf.entities.list.open
wf.entity.attribute.fetch
wf.entity.attribute.value.add
wf.entity.attribute.value.assign
wf.entity.attribute.value.clear
wf.entity.attribute.value.fetch
wf.entity.attribute.value.get
wf.entity.attribute.value.list.close
wf.entity.attribute.value.list.open
wf.entity.attributes.list.close
wf.entity.attributes.list.open
wf.entity.create
wf.entity.delete
wf.entity.fetch

wf.task.definition.create
wf.task.list
wf.task.types.list

workitem.attribute.assign
workitem.attribute.fetch
workitem.attribute.value.get
workitem.attributes.list.close
workitem.attributes.list.open
workitem.complete
workitem.create
workitem.execute
```

```
workitem.fetch
workitem.get
workitem.list
workitem.reassign

worklist.close
worklist.create
worklist.list
worklist.open
```

References

Note: Each reference listed below ends with the page or pages where the associated citations are made. In electronic (PDF) versions of this dissertation, these citation backreferences are included as hyperlinks, so that clicking the page number will take the reader to the page(s) which include(s) the citation.

[Aal98] W. M. P. van der Aalst. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, vol. 8(1):pp. 21–66, 1998. 35, 79

[Aal99] W. M. P. van der Aalst. Inheritance of workflow processes: Four problems - one solution? In *Second OOPSLA Workshop on the Implementation and Application of Object-Oriented Workflow Management Systems*, pp. 1–22. 1999. 39

[Aal00] W. M. P. van der Aalst, A. P. Barros, A. H. M. ter Hofstede, and B. Kiepuszewski. Advanced workflow patterns. In O. E. en P. Scheuermann, editor, *7th Int. Conf. on Cooperative Inf. Systems (CoopIS 2000)*, vol. 1901, p. 1829. Springer-Verlag, Berlin, 2000. LNCS. 88

[Aal01] W. M. P. van der Aalst and T. Basten. Identifying commonalities and differences in object life cycles using behavioral inheritance. In *Proc. of 22nd International Conf. on Applications and Theory of Petri Nets 2001 (ICATPN 2001)*, vol. 2075 of *Lecture Notes in Computer Science*, pp. 32–52. June 2001. 39

[ACQ01] ACQSH. Safety in practice: Making health care safer; second report to the Australian health ministers conference. Tech. rep., Australian Council for Quality and Safety in HealthCare, August 2001. 12

[Ard02] Arden Syntax. E1460-92 standard specification for defining and sharing modular health knowledge bases (arden syntax for medical logic systems), c/o Health Level 7 Inc., 2002. 9, 98

[Arm97] D. G. Armstrong, L. A. Lavery, W. H. van Houtum, and L. B. Harkless. Amputation and reamputation of the diabetic foot. *J Am Podiart Med Assn*, vol. 87(6):pp. 255–259, 1997. 101

[Asp03] L. Asp and J. Petersen. A conceptual model for documentation of clinical information in the EHR. In *Medical Informatics Europe (MIE 2003) Congress*. National Board of Health, Denmark, 2003. URL: http://www.sst.dk/upload/papermie2003_asp013.pdf. 32

[Aus04] Australian Government. *Health*Connect - a health information network for all australians, 2004. An Australian online health record initiative, URL: http://www.healthconnect.gov.au. 161

[Bak02] S. Bakken, J. J. Warren, A. Casey, D. Konicek, C. Lundberg, and M. Pooke. Information model and terminology model issues related to goals. In *AMIA Annual Symposium Proceedings*, pp. 17–21. 2002. 69

[Bar03] S. A. Barretto, J. Warren, A. Goodchild, L. Bird, S. Heard, and M. Stumptner. Linking guidelines to electronic health record design for improved chronic disease management. In *AMIA Annual Fall Symposium*. Washington DC, 2003. 32

[Bar04] S. A. Barretto, J. Warren, and A. Goodchild. Designing guideline-based workflow-enabled electronic health records. In *37th Hawaii International Conference on System Sciences (HICSS-37)*. 2004. 32

[Bea01] T. Beale. Health information standards manifesto, 2001. URL: http://www.deepthought.com.au. 29

[Bea03] T. Beale. *The openEHR Archetype Reference Model*. The *open*EHR Foundation, 2003. URL: http://www.openehr.org. 32

[Ber03] M. Bernauer, G. Kappel, G. Kramler, and W. Retschitzegger. Specification of interorganizational workflows - a comparison of approaches. In *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI2003)*, pp. 30–36. Orlando, USA, July 2003. [ISBN 980-6560-01-9]. 35

[Bic97] P. Bichler, G. Preuner, and M. Schrefl. Workflow transparency. In *Advanced Information Systems Engineering, 9th International Conference CAiSE'97, Barcelona, Catalonia, Spain, June 16-20, 1997, Proceedings*, vol. 1250 of *Lecture Notes in Computer Science*. Springer, 1997. 39

[Bro04] E. Browne. Workflow in healthcare, 2001-2004. URL: http://workflow.healthbase.info. 23

[Bur00] J. Bury, J. Fox, and D. Sutton. The PRO*forma* guideline specification language: progress and prospects. In *Proceedings of the First European Workshop, Computer-based Support for Clinical Guidelines and Protocols (EWGLP2000)*. leipzig, nov 2000. 9

[Car03] J. Cardoso and A. Sheth. Semantic e-workflow composition. *Journal of Intelligent Information Systems*, vol. 21(3):pp. 191–225, 2003. 41

[Cas96] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow evolution. In *International Conference on Conceptual Modeling / the Entity Relationship Approach*, pp. 438–455. 1996. 39

[Ces02] B. Cesnik. Governance issues in electronic decision support systems. in Report of the Electronic Decision Support Governance Workshop, 2002. National Institute of Clinical Studies, Melbourne. 30

[Coi00] E. Coiera, R. Jayasuriya, J. Hardy, A. Bannan, and M. Thorpe. Communication loads on clinical staff in the emergency department. *The Medical Journal of Australia*, vol. 176(9):pp. 415–418, May 2000. 12

[Dar93] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, vol. 20(1-2):pp. 3–50, 1993. 67

[Daz97] L. Dazzi, C. Fassion, R. Saracco, S. Quaglini, and M. Stefanelli. A patient workflow management system built on guidelines. In *Proc. American Medical Informatics Association (AMIA)*, pp. 146–150. 1997. 17

[Dia04] Diabetes Aust. and RACGP. *Diabetes Management in General Practice*. Diabetes Australia and Royal Australian College of General Practice, 2004. 62

[Div03] M. N. Div. of General Practice. Assessing the new MBS items - groups 18 & 19, may 2003. URL: http://www.mndgp.org.au/Accessing new MBS items.pdf. xiv, 100, 101, 102

[Dog97] A. Dogac, E. Gokkoca, S. Arpinar, P. Koksal, I. Cingil, B. Arpinar, N. Tatbul, P. Karagoz, U. Halici, and M. Altinel. Design and implementation of a distributed workflow management system: METUFlow.

In A. Dogac, L. Kalinichenko, T. Ozsu, and A. Sheth, editors, *NATO-ASI on Workflow Management Systems and Interoperability*, pp. 60–90. August 1997. 25

[DoH02] DoHA. Enhanced primary care and the new MBS items (health assessments, care plans and case conferences, June 2002. URL: http://www.health.gov.au/pubs/mbs/mbs5/amendme1.htm. 11

[Ecc02] M. Eccles, E. Mccoll, N. Steen, N. Rousseau, J. Grimshaw, D. Parkin, and I. Purves. Effect of computerised evidence based guidelines on management of asthma and angina in adults in primary care:cluster randomised controlled trial. *BMJ*, vol. 325, 2002. 10

[Ell95] C. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan, editors, *Proceedings of the Conference on Organizational Computing Systems*, pp. 10–21. ACM SIGOIS, ACM Press, New York, Milpitas, California, August 1995. 38, 83, 129

[Fie90] M. Field and K. Lohr, editors. *Clinical Practice Guidelines: Directions for a New Program*. National Academic Press, Washington, DC, 1990. 8, 36, 124

[Fis01] L. Fischer. Workflow handbook 2001. Tech. rep., Workflow Management Coalition (WfMC), 2001. 14, 134

[Fra02] M. J. Franz, J. P. Bantle, C. A. Beebe, J. D. Brunzell, J.-L. Chiasson, A. Garg, L. A. Holzmeister, B. Hoogwerf, E. Mayer-Davis, A. D. Mooradian, J. Q. Purnell, and M. Wheeler. Evidence-based nutrition principles and recommendations for the treatment and prevention of diabetes and related complications. *Diabetes Care*, vol. 25:pp. 148–198, 2002. 11

[Gia00] G. de Giacomo, Y. Lesperance, and H. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, vol. 1-2(121):pp. 109–169, August 2000. 37

[Gor97] C. Gordon, I. Herbert, P. Johnson, P. Nicklin, and P. Reeves. Telematics for clinical guidelines: A conceptual modelling approach. In *Procs, Medical Informatics Europe (MIE97)*. 1997. 9

[Gor99] C. Gordon, M. Veloso, and the PRESTIGE Consortium. Guidelines in healthcare: the experience of the prestige project. In *Procs, Medical Informatics Europe (MIE99)*. 1999. 9

[Gro01] D. Gross and E. Yu. Evolving system architecture to meet changing business goals: an agent and goal-oriented approach. In *ICSE-2001 Workshop: From Software Requirements to Architectures (STRAW2001)*, pp. 13–21. Toronto, Canada, May 2001. 36

[Han98] Y. Han, A. Sheth, and C. Bussler. A taxonomy of adaptive workflow management, 1998. Workshop of the 1998 ACM Conference on Computer Supported Cooperative Work, Seattle, Washington, USA, November. 35

[Hau02] R. Haux, E. Ammenwerth, W. H. W, and P. Knaup. Health care in the information society. a prognosis for the year 2013. *Int J Med Inf*, vol. 66(1-3):pp. 3–21, Nov 2002. 27

[Hes99] G. Hesina, D. Schmalstieg, A. Fuhrmann, and W. Purgathofer. Distributed open inventor: A practical approach to distributed 3D graphics. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST99)*, pp. 74–81. ACM Press, New York, November 1999. 80

[HL7] HL7. Health Level Seven (HL7). URL: http://www.hl7.org. 31, 69

[IOM99] IOM. To err is human: Building a safer health system. Tech. rep., Institute of Medicine, 1999. 12

[Jab94] S. Jablonski. Mobile: A modular workflow model and architecture. In *Procs. Fourth International Working Conference on Dynamic Modelling and Information Systems*. Noordwijkerhout, The Netherlands, September 1994. 24

[Jac95] S. Jacobs and R. Holten. Goal-driven business modelling: Supporting decision-making within information systems development. Tech. rep., RWTH, Aachen, 1995. 36, 67

[Joe99] G. Joeris and O. Herzog. Managing evolving workflow specifications with schema versioning and migration rules. Tech. rep., University of Bremen, 1999. 39, 92

[Kap00] G. Kappel, S. Rausch-Schott, and W. Retschitzegger. A framework for workflow management systems based on objects, rules and roles. *ACM Computing Surveys (CSUR)*, vol. 32(1es):p. 27, 2000. 37

[Kli00] J. Klingemann. Controlled flexibility in workflow management. In *Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE'00)*, pp. 126–141. Springer Verlag, Stockholm, June 2000. 40, 67

[Kou99] M. Koubarakis and D. Plexousakis. Business process modeling and design: AI models and methodology. In *IJCAI-99 workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*. 1999. 36

[Kra98] M. Kradolfer and A. Geppert. Dynamic workflow schema evolution based on workflow type versioning and workflow migration (tr 98.02). Tech. rep., Dept. of Computer Science, University of Zurich, 1998. 92

[Lee96] W. Lee, G. E. Kaiser, P. D. Clayton, and E. H. Sherman. OzCare: A workflow automation system for care plans. In *Proc. American Medical Informatics Association (AMIA)*. 1996. 23

[Lee04] Y. Lee, C. Patel, S. A. Chun, and J. Geller. Compositional knowledge management for medical services on semantic web. In *WWW2004, May 17 22, 2004, New York,*. New York, USA., May 2004. 41

[Maj04a] S. Majithia, D. W.Walker, and W.A.Gray. Automated composition of semantic grid services. In *AHM 2004, Nottingham, UK, August 2004*. Nottingham, UK, August 2004. 41

[Maj04b] S. Majithia, D. W.Walker, and W.A.Gray. Automated web service composition using semantic web technologies. In *International Conference on Autonomic Computing (ICAC-04)*. New York, USA, May 2004. 41

[Man99] D.-A. Manolescu and R. E. Johnson. Dynamic object model and adaptive workflow. OOPSLA'99 Metadata and Active Object-Model Pattern Mining Workshop, 1999. 39, 129

[Man03] D. J. Mandell and S. A. McIlraith. Adapting BPEL4WS for the semantic web: The bottom-up approach to web service interoperation. In *The Proceedings of the Second International Semantic Web Conference (ISWC2003)*. Sanibel Island, Florida, 2003. 42

[Mat99] C. Mathers, T. Vos, and C. Stevenson. The burden of disease and injury in Australia. Tech. Rep. AIHW Catalogue no. PHE 17, Australian Institue of Health & Welfare, Canberra, 1999. 1

[Mat00] Mater UQ Centre for General Practice. Management of diabetes mellitus in adults - the Queensland standard pathway 2000, 2000. URL: http://www.uq.edu.au/cgpmh/gp-paths/gp-22diabpath.htm, accessed 28 May 2003. 100, 101

[Mik97] S. Miksch, Y. Shahar, and P. Johnson. Asbru: A task-specific, intention-based, and time-oriented language for representing skeletal plans. In *7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*. Milton Keynes, UK, Jan 1997. 9, 66, 68

[Mil96] J. Miller, A. Sheth, K. Kochut, and X. Wang. Corba-based run-time architectures for workflow management systems. *Journal of Database Management, Special Issue on Multidatabases*, vol. 7(1):pp. 16–27, winter 1996. 25

[Mor04] I. Morrison, B. Lewis1, S.-T. Liaw, and E. Deveny. Modelling the clinical processes of prescribing (MCPOP) - information, clinical workflow and processes. In *Australian Health Informatics Conference (HIC2004)*. Brisbane, Australia, July 2004. 41

[Mü99] R. Müller and E. Rahm. Rule-based dynamic modification of workflows in a medical domain. In A. Buchmann, editor, *Proceedings of BTW99*, pp. 429– 448. Springer, Berlin, Freiburg im Breisgau, March 1999. 26

[Mü02] R. Müller. *Event-Oriented Dynamic Adaptation of Workflows: Model, Architecture, and Implementation*. Ph.D. thesis, Fakultät für Mathematik und Informatik der Universität Leipzig, 2002. 27, 130, 134

[Myl99] J. Mylopoulos, L. Chung, and E. Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, vol. 42(1):pp. 31–37, 1999. 36, 67

[Obj03] Object Management Group. Unified modelling language,v1.5, chapter 2 - uml semantics, 2003. URL: http://www.omg.org. 47

[OM98] L. Ohno-Machado, J. H. Gennari, S. N. Murphy, N. L. Jain, S. W. Tu, D. E. Oliver, E. Pattison-Gordon, R. A. Greenes, E. H. Shortliffe, and G. O. Barnett. The guideline interchange format: a model for representing guidelines. *J Am Med Inform Assoc*, vol. 5:pp. 357–372, 1998. 9

[Ozb97] J. G. Ozbolt. Multiple attributes for patient care data: Toward a multiaxial, combinatorial vocabulary. In *AMIA Annual Symposium Proceedings*. 1997. 69

[Pan02] S. Panzarasa, S. Maddè, S. Quaglini, C. Pistarini, and M. Stefanelli. Evidence-based careflow management systems: the case of post-stroke rehabilitation. *J of Biomed Informatics*, vol. 35:pp. 123–129, 2002. 106

[Pay00] C. B. Payne. Diabetes-related lower limb amputations in australia. *Medical J Aust*, vol. 173(7):pp. 352–354, 2000. 101

[Pel03] M. Peleg, A. Tu, J. Bury, P.Ciccarese, J. Fox, R. Greenes, R. Hall, P. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E. Shortliffe, and M. Stefanelli. Comparing computer-interpretable guideline models: a case study approach. *J Am Med Inform Assoc*, vol. 10(1):pp. 52–68, 2003. 10

[Phi02] P. Phillips and A. Evans. One pair must last a lifetime - case studies of foot care in diabetes. *Australian Family Physician*, vol. 31(6):pp. 546–549, June 2002. 100, 101

[Pur99] I. Purves, B. Sugden, N. Booth, and M. Sowerby. The PRODIGY project - the iterative development of the release one model. In *Proc AMIA Symp.*, pp. 359–363. 1999. 9

[Qua00] S. Quaglini, M. Stefanelli, A. Cavallini, G. Micieli, C. Fassino, and C. Mossa. Guideline-based careflow systems. *Artificial Intelligence Medicine*, vol. 20(1):pp. 5–22, August 2000. 24

[Qua01] S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa. Flexible guideline-based patient careflow systems. *Artificial Intelligence in Medicine*, vol. 22(1):pp. 65–80, 2001. 9, 24, 106

[RAC03] RACGP. *Step-by-step care planning*. Royal Australian College of General Practioners, 2003. URL: http://www.racgp.org.au/espcite/cp.asp. 18

[Ram04] P. Ram, D. Berg, S. Tu, G. Mansfield, Q. Ye, R. Abarbanel, and N. Beard. Executing clinical practice guidelines using the SAGE execution engine. In M. F. *et al* , editor, *MEDINFO 2004*. IOS Press, Amsterdam, 2004. 11

[Rei98] M. Reichert and P. Dadam. ADEPT flex -supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, vol. 10(2):pp. 93–129, 1998. 26, 129

[Rod95] J. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, vol. 37(7):pp. 383–393, 1995. 38

[Rou03] N. Rousseau, E. McColl, J. Newton, J. Grimshaw, and M. Eccles. Practice-based, longitudinal, qualitative interview study of computerised evidence based guidelines in primary care. *BMJ*, vol. 326:p. 314, 2003. 111

[Sad01] S. Sadiq, W. Sadiq, and M. E. Orlowska. Pockets of flexibility in workflow specification. In *20th International Conference on Conceptual Modelling (ER2001)*, pp. 513–526. Yokohama, Japan, November 2001. 40, 129

[Sad02] S. Sadiq and P. Mangan. On building workflow models for flexible processes. In *Thirteenth Australasian Database Conference*, vol. 5. Australian Computer Society, Inc, Melbourne, Australia, 2002. 40, 129

[SAG02] SAGE. Standards-based **s**harable **a**ctive **g**uidelines **p**roject, 2002. URL: http://www.sageproject.net. 11

[Sch01] G. Schadow, D. Russler, and C. McDonald. Conceptual alignment of electronic health record data with guideline and workflow knowledge. *Intl J Med Info*, vol. 64:pp. 259–274, 2001. 32

[Sha98] Y. Shahar, S. Miksch, and P. Johnson. The asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artificial Intelligence in Medicine*, vol. 14(1-2):pp. 29–51, 1998. 37

[Shi00] R. Shiffman, B. Karras, A. Agrawal, R. Chen, L. Marenco, and S. Nath. GEM: A proposal for a more comprehensive guideline document model using XML. *J Am Med Informatics Assoc*, 2000. 9

[Sir03] E. Sirin, J. Hendler, and B. Parsia. Semi-automatic composition of web services using semantic descriptions, April 2003. Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003. 41

[Sit02] D. F. Sittig, B. L. Hazlehurst, T. Palen, J. Hsu, H. Jimison, and M. C. Hornbrook. A clinical information system research landscape. *The Permanente Journal*, vol. 6(2):pp. 62–68, 2002. 28

[Sno85] R. Snodgrass and I. Ahn. A taxonomy of time databases. In *Proceedings of the 1985 ACM SIGMOD international conference on Management of data*, pp. 236–246. 1985. 53

[Sta] Stanford Medical Informatics. Protégé ontology editor and knowledge acquisition system. URL: http://protege.stanford.edu. 9

[Tex00] Texas Diabetes Council. *Hypertension algorithm for diabetes mellitus in adults*. Texas Department of Health, 2000. URL: http://www.tdh.state.tx.us/diabetes/PDF/algorithms/HYPER.PDF. 103, 104

[Tu01] S. Tu and M. Musen. Modeling data and knowledge in the EON guideline architecture. In *Proc. MedInfo 2001*, pp. 280–284. London, UK, 2001. 9

[Tu03] S. W. Tu, M. A. Musen, R. D. Shankar, J. R. Campbell, K. Hrabak, J. McClay, S. M. Huff, R. McClure, C. Parker, R. Rocha, R. Abarbanel, N. Beard, J. Glasgow, J. G. Mansfield, P. Ram, Q. Ye, E. Mays, T. Weida, C. G. Chute, K. McDonald, D. Mohr, M. A. Nyman, S. M. Scheitel, H. Solbrig, D. Zill, and M. K. Goldstein. Modeling guidelines for integration into clinical workflow. Tech. rep., Stanford Medical Informatics, 2003. 9

[Uni03] Universität Leipzig. HematoWork: A workflow system for protocol-directed care in distributed hemato-oncology, 2003. URL: http://dbs.uni-leipzig.de/de/Research/hemato.html. 134

[War99] J. Warren, G. Beliakov, J. Noone, and H. Frankel. Chronic disease coordinated care planning: flexible, task-centered decision support. In *Proceedings of the 32nd Hawaii International Conference on System Sciences (HICSS-32)*, vol. 4. Maui, Hawaii, January 1999. 13, 25, 111

[Wol02] D. Wollersheim. A review of decision support formats with respect to Therapeutic Guidelines Limited requirements. In *Procs. Australian Health Informatics Conference, 2002*. 2002. 9

[Wor95] Workflow Management Coalition. *The Workflow Reference Model*, TC00-1003 edn., January 1995. 139

[Wya91] J. Wyatt and D. Spiegelhalter. Field trials of medical decision-aids: potential problems and solutions. In P. Clayton, editor, *Proceedings of the Fifteenth Annual Symposium on Computer Applications in Medical Care*, vol. 33, pp. 3–7. American Medical Informatics Association, Washington, 1991. 30

[Zim01] P. Zimmet and T. Welborn. Diabetes and associated disorders in Australia 2000. the Australian diabetes, obesity and lifestyle study, 2001. 100