COMPLIANCE FLOW - AN INTELLIGENT WORKFLOW MANAGEMENT SYSTEM TO SUPPORT ENGINEERING PROCESSES

By

YEE CHUNG CHEUNG

A doctoral thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

March 2003

Abstract

This work is about extending the scope of current workflow management systems to support engineering processes. On the one hand engineering processes are relatively dynamic, and on the other their specification and performance are constrained by industry standards and guidelines for the sake of product acceptability, such as IEC61508 for safety and ISO9001 for quality.

A number of technologies have been proposed to increase the adaptability of current workflow systems to deal with dynamic situations. A primary concern is how to support open-ended processes that cannot be completely specified in detail prior to their execution. A survey of adaptive workflow systems is given and the enabling technologies are discussed.

Engineering processes are studied and their characteristics are identified and discussed. Current workflow systems have been successfully used in managing "administrative" processes for some time, but they lack the flexibility to support dynamic, unpredictable, collaborative, and highly interdependent engineering processes. The key requirements which have to be addressed by future systems, if they are to succeed in supporting engineering processes, are identified and discussed. A novel framework to deal with such requirements is proposed.

One serious limitation of current workflow systems is that they lack of the ability to ensure the planning and the execution of a process is compliant with a standard. In this thesis, process model reasoning is proposed to identify compliance errors of a user-defined process by matching it against a standard model both during process specification and execution and preventing unacceptable products being made as the result of a wrongly planned process.

This thesis contributes towards a better understanding of adaptive workflow technology, and shows how new techniques can be used to support engineering processes. In particular, the novel ability of managing process compliance with standards forms a key contribution in workflow technology research. It raises the importance and the application of meta-process management, which points the direction to a major new area of research.

Keywords: Workflow Management, Process Compliance Management, Process-based Reasoning, Capability Matching

Acknowledgement

I am indebted to my research supervisor Professor Paul Chung for his guidance, encouragement and enthusiasm throughout the duration of the research.

I would like to express sincere gratitude towards my research director Mr. Ray Dawson and all the members of the Department of Computer Science, both past and present, for their useful advice and assistance.

To my parents

Table of Contents

Part 2: Supporting Engineering Process with a Novel Framework

Appendix

List of Figures

List of Tables

List of Equations

List of Abbreviations

AI	-	Artificial Intelligence
APES	-	Assuring Programmable Electronic Systems
API	-	Application Programme Interface
ALARP	-	As Low As Reasonably Practicable
B2B	-	Business to Business
CA	-	Central Administration
CAD	-	Computer Aided Design
CD	-	Conceptual Distance
CHC	-	Correspondence Handling Centre
CN	-	Change Note
CORBA	-	Common Object Request Broker Architecture
CRN	-	Change Request Note
CSCW	-	Computer Supported Cooperative Works
CTO	-	Chief Technical Officer
DBMS	-	Database Management System(s)
DCS	-	Distributed Controlled System
DFD	-	Data Flow Diagram
ECA	-	Event Condition Action
EDP	-	Electronic Data Process
E/E/PES	-	Electrical/Electronic/Programmable Electronic System
ERD	-	Entity-Relationship Diagram
ERM	-	Entity Relationship Modelling
EUC	-	Equipment Under Control
FTP	-	File Transfer Protocol
FSD	-	Final Switching Device
GPSG	-	Generalised Process Structure Grammars
GUI	-	Graphic User Interface
GUID	-	Globally Unique Identifier
HPN	-	High-level Petri Nets

HR	-	Highly Recommended
HTN	-	Hierarchical Task Network
HTTP	-	Hypertext Transfer Protocol
IE	-	Internet Explorer
IEC	-	International Electrotechnical Commission
IDL	-	Interface Definition Language
IETF	-	Internet Engineering Task Force
IIS	-	Internet Information Server
IN	-	Information Navigation
I/O	-	Input / Output
IPO	-	Input Process Output
IS	-	Information System
IT	-	Information Technology
JCALS	-	Joint Computer-aided Acquisition and Logistics Support
KIF	-	Knowledge Interchange Format
MIME	-	Multipurpose Internet Mail Extensions
MOM	-	Message Oriented Middleware
NIS	-	National Institute of Standard and Technology
NPD	-	New Product Development
NR	-	Not Recommended
ODBC	-	Open Database Connectivity
OIS	-	Office Information System
OMG	-	Object Management Group
OMT	-	Object Modelling Technique
O-O	-	Object Oriented
OSSD	-	Output Signal Switching Device
PA	-	Planning Assistance
PERT	-	Program Evaluation and Review Technique
PIF	-	Process Interchange Format
PN	-	Petri Nets

PSL	-	Process Specification Language
R	-	Recommended
RC	-	Recommendation Check
SIL	-	Safety Integrity Level
SMTP	-	Simple Mail Transfer Protocol
SWAP	-	Simple Workflow Access Protocol
TBPM	-	Task Based Process Management
TCP/IP	-	Transmission Control Protocol over Internet Protocol
UDP	-	User-Defined Process
UML	-	Unified Modelling Language
VB	-	Visual Basic
WAPI	-	Workflow Application Programme Interfaces
WfM	-	Workflow Management
WfMC	-	Workflow Management Coalition
WfMF	-	Workflow Management Facility
WfMS	-	Workflow Management System(s)
WPC	-	Work Processing Centres
WPDL	-	Workflow Process Definition Language
XPDL	-	XML Process Definition Language
XML	-	eXtensible Markup Language

Part 1

Introduction

Chapter 1

Overview

"To be, or not to be: that is the question:
whether 'tis nobler in the mind to suffer
the slings and arrows of outrageous fortune,
or to take arms against a sea of troubles,
and, by opposing, end them."
- William Shakespeare

1.1 Introduction

Workflow Management is a fast evolving technology which is being increasingly exploited by a variety of businesses. Its primary characteristic is the control of tasks execution in accordance with a well defined process plan. After a decade of endeavour, current workflow management systems are widely used for the management of "administrative" business processes. However, they lack flexibility to cope with the more dynamic situations encountered in ad-hoc and collaborative engineering processes.

This chapter gives an introduction to this thesis. It is organised as follow: §1.2 identifies the problem addressed in this thesis and its objectives; §1.3 discusses the approach adopted to meet these objectives; §1.4 outlines the contributions; and the thesis organisation is given in §1.5.

1.2 Problem Statement and Objectives

The problem addressed in this thesis is formulated as:

> *How is it possible to provide intelligent support for the management of dynamic engineering processes, with the focus of ensuring that their specification and performance are compliant with particular industry standards?*

Recently, significant efforts have been made in increasing the flexibility of workflow management where a variety of technologies are proposed. There is, however, no existing framework to integrate these cutting edge technologies to support dynamic engineering processes.

As regards to the management of an engineering project, significant resources are devoted to managing its compliance with industry standards to ensure product acceptability. In this context, process compliance means that there is a clear description of the design stages and, at each stage, the inputs to that stage are fully and unambiguously defined, and also all the objectives and requirements of the standard are met. The standards are generic but every application is different because of the differences in the project details. Thus, ensuring the compliance of a user-defined process with a particular standard forms a key challenge.

In an engineering project, particularly in the safety engineering domain, much of the time of developers, managers and quality assurance teams is occupied with tracking and managing the compliance of the project. A workflow system with compliance management ability can not only shorten the development time and reduce the cost, but also ensure the quality of the process and product. Thus, a solution to this problem is strongly industrially motivated.

It is not the intention of this work to substitute the current risk assessment methods with a computerised system. Instead, the aim is to endow workflow systems with compliance management ability to provide assistance in the management of an engineering project. As the non-compliant elements of a process are identified and handled earlier, the required development time and cost will be reduced.

The objectives of this research are:

- To develop a novel framework that provides a platform where a variety of cutting edge technologies work together to provide flexible support for the management of dynamic engineering processes.
- To develop a series of compliance check algorithms to ensure the specification and the execution of a process comply with a particular standard.

1.3 Approach

In order to meet these objectives, the thesis explores three parallel and interrelated threads:

- The concept of adaptive workflow management and its enabling technologies.
- The specification of engineering process characteristics in relation to workflow management.
- The computer support that is required for compliance checks, using IEC61508 international safety standard as an example.

Insights achieved from understanding the concept of adaptive workflow management and its enabling technologies are used as a basis for the proposal of a new framework. Workflow management is a relatively new technical field with around ten years' history. This report focuses on the utilisation of various technologies in increasing the adaptation of workflow management systems to the dynamic world.

Engineering processes differ from business processes for which the current workflow management systems are well suited. Hence, the characteristics of engineering processes have to be studied and accordingly the requirements of a system which is to succeed in supporting engineering processes are identified. A novel workflow management framework which builds on a variety of technologies is proposed to deal with such requirements.

The international safety standard IEC61508 is used as an example for compliance management. The proposed solution to ensure the compliance of a process with a selected standard is to use a software agent to match a user defined process against a model of standards during the process build-time and run-time to identify possible errors. To do so, there must be a representation of the standards in the system. The modelling of a standard in terms of process management are studied and discussed. Finally, a number of compliance check algorithms are developed for matching the process against the model of standards.

To assess the proposed approach, several key components are implemented and three evaluation studies are performed.

1.4 Contributions

The contributions of this thesis are:

- Through the application of workflow management to engineering processes, it has contributed to an enhanced understanding of "adaptive workflow technology".
- It contributes to the compliance domain by empowering the compliance management at process level where errors are detected and prevented in advance of process execution.
- The novel feature of process compliance management creates a new research direction in the workflow community, and its application in managing project compliance indicates the importance of this meta-process control capability.
- A new approach of using set theory to tackle the problem of traditional "distance" approach to assess goodness of fit (GOF) between two concepts of interest.
- A new approach of integrating the concept of CSCW with workflow management is proposed, allowing users to collaborate while the process is running.
- A new approach of agent selection based on fuzzy capability matching.
- A novel process model that further captures capability knowledge and enables that information objects are linked to their related tasks.
- The proposed framework gives a platform where a variety of technologies can be used to increase the required adaptation of a workflow management system in supporting dynamic processes.

1.5 Thesis Organisation

This thesis is organised into four parts where each part focuses on a particular subject and can be read separately based on the reader's background, as depicted in Figure 1-1.

Part 1: Introduction

Chapter 2 gives an introduction to workflow management technology, process modelling paradigms and the latest workflow standardisation works.

Chapter 3 gives a review of several adaptive workflow products and research prototypes. A variety of different technologies together with their utility in increasing the adaptation of workflow management are introduced and discussed.

Part 2: Supporting Engineering Processes with a Novel Framework

Chapter 4 studies engineering processes, identifies their characteristics and outlines the requirements of a workflow system that is to succeed in supporting engineering processes.

Chapter 5 introduces the proposed framework which integrates several technologies to address the requirements identified in Chapter 4. The comprising components and their underlying technologies are presented and discussed.

Part 3: Managing Process Compliance

Chapter 6 gives an overview of the international safety standard IEC61508 and discusses the modelling of a standard for meta-process management, followed with a proposed solution.

Chapter 7 presents the proposed solution of managing process compliance which is based on the concept of process model reasoning and compliance checks for ensuring standard compliant.

Part 4: System Development and Evaluation

Chapter 8 describes the design and the implementation of the system. The process model to support the proposed framework is introduced. The implementation architecture is presented, followed with the proposed fuzzy matching algorithm for agent selection.

Chapter 9 evaluates the system. Three cases are studied. A comparison, in terms of flexibility, is also made between Compliance Flow and the adaptive workflow systems discussed in Chapter 3.

Finally, Chapter 10 concludes the thesis with a summary of contributions, limitations and future directions.

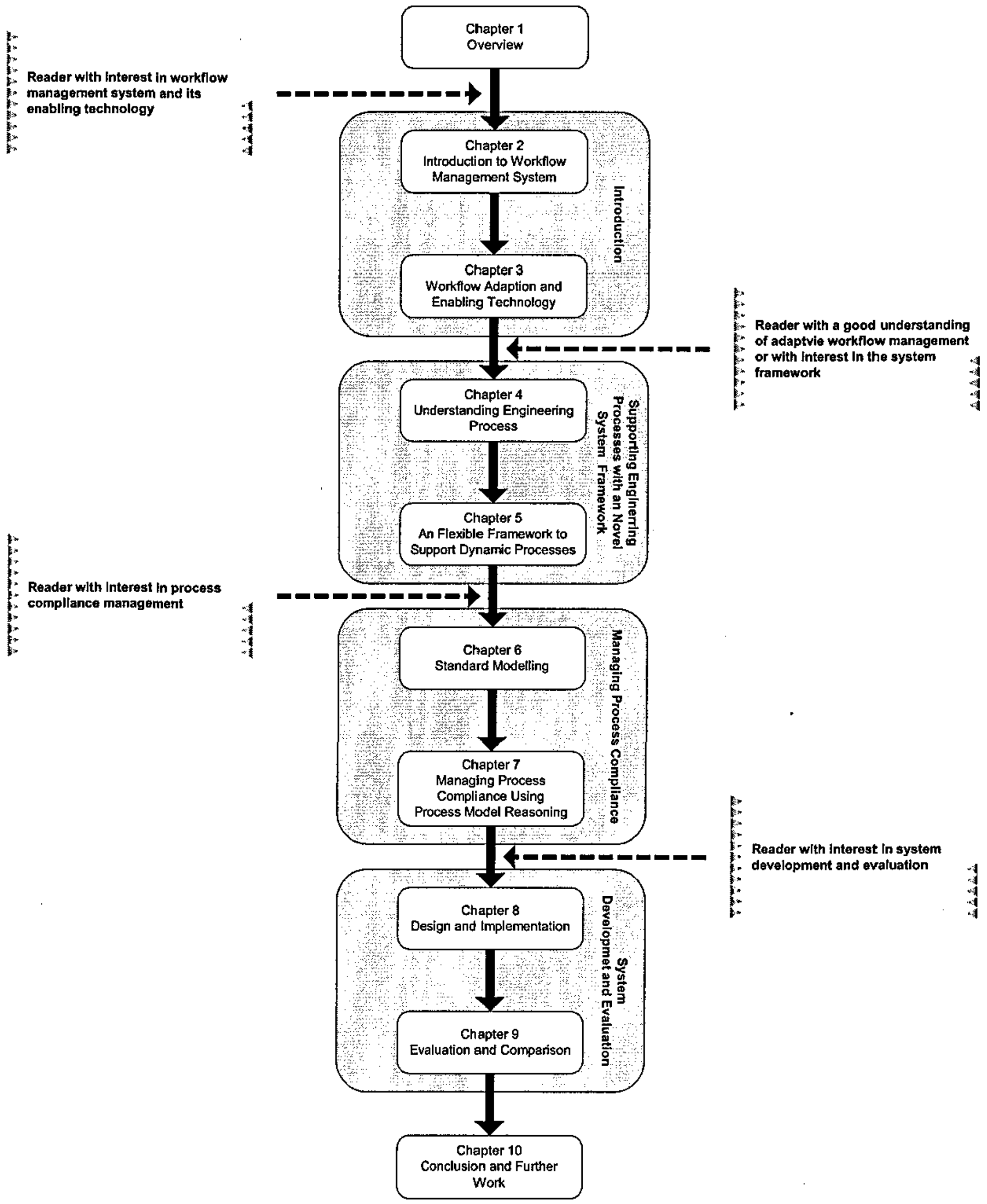

Chapter 1
Overview
Reader with interest in workflow management system and its enabling technology
Chapter 2
Introduction to Workflow Management System
Chapter 3
Workflow Adaption and Enabling Technology
Introduction
Reader with a good understanding of adaptvie workflow management or with interest in the system framework
Chapter 4
Understanding Engineering Process
Chapter 5
An Flexible Framework to Support Dynamic Processes
Supporting Engineering Processes with an Novel System Framework
Reader with interest in process compliance management
Chapter 6
Standard Modelling
Chapter 7
Managing Process Compliance Using Process Model Reasoning
Managing Process Compliance
Reader with interest in system development and evaluation
Chapter 8
Design and Implementation
Chapter 9
Evaluation and Comparison
System Developmet and Evaluation
Chapter 10
Conclusion and Further Work

Figure 1-1. Thesis organisation.

Chapter 2

Introduction to Workflow Management System

"What we observe is not nature itself, but nature exposed to our method of questioning."

- Werner Heisenberg

2.1 Introduction

Workflow technology has been used for decades. It derives partly from the Office Information System (OIS) field and partly from the Computer Supported Cooperative Work (CSCW) field. A workflow system includes a set of technological solutions intended to automate or support business or work processes that are described in an explicit process model. Therefore, workflow technology can be seen as a specific type of groupware for supporting collaboration work in which procedures are planned and expressed in process models. There are two perspectives of workflow: the group work support perspective and the organisational process automation perspective, which shows the breadth of workflow as a technology area.

This chapter gives an overview of the general features of workflow management systems and their enabling technologies. It is organised as follow: §2.2 gives the definitions of some important workflow terms and concepts; §2.3 identifies the workflow characteristics; §2.4 presents Ader's workflow classification scheme; §2.5 discusses contemporary process modelling and workflow specification techniques; §2.6 introduces the idea of workflow engine and discusses implementation issues of workflow systems; §2.7 describes the standardisation effort in the workflow community. This chapter ends with a summary section.

2.2 Definition of Workflow Terms

2.2.1 Workflow

The Workflow Management Coalition (WfMC) published a glossary of terms related to workflow (WfMC, 2000a). Workflow is defined as:

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

Georgakopoulos et al. (1995) gave a more explicit description of the relationship between humans and computer systems in a workflow process. They defined a workflow or a process as a collection of activities organised to accomplish some business goals. An activity can be performed by one or more software systems, one or a team of humans, or a combination of these. Human activities may range from interacting with a computer closely or loosely. In the "closely" case, a human uses computer tools as part of an activity. In the "loosely" case, the human uses a computer system only to communicate that an activity has been completed. In addition to a collection of activities, a workflow may include constraints that influence the order of performing activities as well as information flow between them.

2.2.2 Process Definition, Process Instance and Activity Instance

A process definition provides a template for an actual process instance. A single process definition may be instantiated many times to create a number of process instances. Each process instance consists of activity instances to be executed. Enactment involves instantiating a process definition by assigning activities to humans and software system while maintaining the constraints between activities (Jarvis, 1998). These relationships are illustrated in Figure 2-1 as an Entity-Relationship Diagram (ERD).

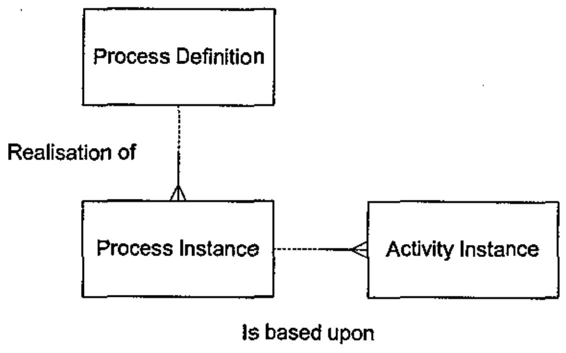


Figure 2-1. Process definition to process instance object model.

The relationships between process definition, process instance, activity instance, and the users of workflow systems are illustrated in Figure 2-2. The process definition on the left is instantiated twice to create two separate process instances. The activities within each process instance are assigned to individuals for execution via work lists.

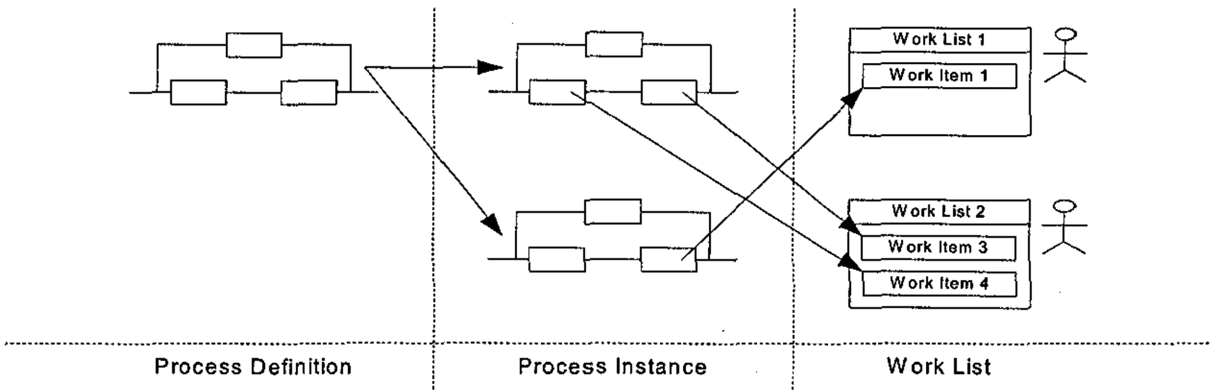


Figure 2-2. Process Definitions, Process Instances and Work Lists.

2.2.3 Workflow Management System

Workflow Management System (WfMS) is a system that aims to provide computer-based support for the task of workflow management. It supports the specification (build-time functions), execution (run-time control functions), and dynamic control of workflows involving humans and information systems (run-time interactions) (McCarthy & Sarin, 1993). WfMS runs on one or more workflow engines that are able to interpret the process definition, interact with knowledge participants and, where required, invoke the use of IT tools and applications (WfMC, 2000a). In some workflow systems, the workflow engines are located in different areas and the communication between them is through the Internet.

Figure 2-3 shows an outline of workflow management architecture. The process analysis, modelling and definition tools facilitate the specification of the components of a workflow as a process definition. The workflow enactment service enacts a process definition by assigning tasks to humans and software systems while also maintaining the constraints between tasks. The workflow control data represents the dynamic state of the workflow system and its process instance, which is managed and accessible by the workflow management system or workflow engine. The workflow relevant data is used by the workflow management system or workflow engine to determine the state transitions of the workflow instance. The application data is strictly used by applications supporting the process instance.

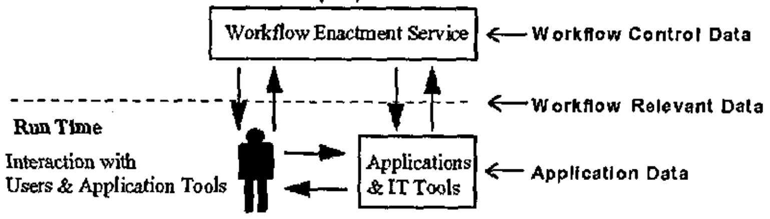

Build Time
Process Design & Definition
Business Process Analysis, Modelling & Definition Tools
Process Definition
Run Time
Process Instanciation & Control
Process changes
Workflow Enactment Service
Workflow Control Data
Workflow Relevant Data
Run Time
Interaction with Users & Application Tools
Applications & IT Tools
Application Data

Figure 2-3. Workflow system architecture and data structure (WfMC, 2000a).

2.3 Workflow Characteristics

The essential workflow characteristics are persons, tasks/activities, application tools and resources (Marshak, 1994; 1997). The (role-playing) persons perform tasks using application tools that provide access to various shared information resources. This characteristics model of workflow is shown in Figure 2-4.

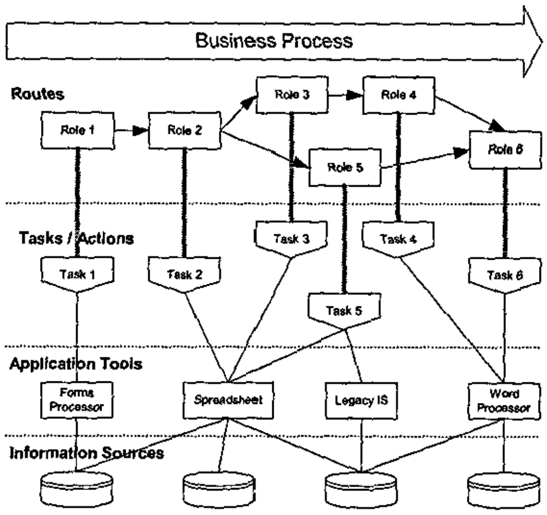


Figure 2-4. Characteristics of workflow (Marshak, 1994).

Marshak (1994; 1997) defines the "3Rs" and the "3Ps" of workflow technology:

Rules. Workflow systems take various business rules into account. The rules should be maintainable and understandable by business professionals.

Routes. A route is strongly coupled to the concept of information logistic that typically support organisation flow of all kinds of objects including documents, forms as well as processes.

Roles. Information is routed to roles rather than to a particular person. The role in an organisation is a group of people with the required skills and authority.

Processes. Business processes span over organisation units and legacy information systems.

Policies. Policies correspond to a normative process model that describes how certain processes should be handled.

Practices. This is the way that a work is actually performed in the organisation.

2.4 Workflow Taxonomy Scheme

There are several attempts to classify current workflow products. The most useful and well known is the Ader's Workflow Classification Scheme (Ader, 1997) which consists of four categories: production, administrative, collaborative and Ad-hoc. Other works include Human-Oriented and System-Oriented classifications (Georgakopoulos, 1995), and Abbot and Sarin's

Dimensions (Abbott and Sarin, 1994).

2.4.1 Production

Production workflow systems are used to manage large numbers of similar tasks, and to optimise productivity. This can be achieved by automating numerous highly repetitive and complex activities, usually in a non-stop manner to achieve Straight-Through-Processing. Human input is required only for managing exceptions. They can be tightly integrated with legacy systems.

2.4.2 Administrative

Administrative workflow systems are used to automate administrative tasks driven by form filling. A very important requirement of this type of tool is that processes can be easily defined. The process definition is hardly changed during the execution. Typically, there are many different processes running concurrently and they tend to involve numerous staff, thus, flexibility is more important than productivity in administrative workflow systems.

2.4.3 Collaborative

Collaborative workflow systems are used to automate business-critical processes that are not transaction oriented. They are used to support collaborative teamwork. Groups can vary from small, project-oriented teams to widely dispersed people with common interests. Throughput is not an important consideration, and process definitions are not rigid and can be amended frequently. The use of Internet and the World Wide Web (W3s) to support team communications across organisations is a critical success factor.

2.4.4 Ad-hoc

Ad-hoc workflow systems are used to support routing and tracking of routine office work that is based on unstructured information. The process definition can be created and amended very quickly to meet new circumstances. Ad-Hoc workflow systems maximise flexibility in areas where throughput and security are not major concerns. Unlike production workflow systems where the processes are owned at the organisation level, in Ad-hoc workflow systems the practitioners own the processes.

The most important distinction between various workflow types is the distinction between production workflow and Ad-hoc workflow. It differs in who is allowed to change the process definition (administrators or users) and when it can be changed (build-time or both build-time and run-time).

2.5 Process Modelling and Workflow Specification

The purpose of process modelling is to produce an abstraction of a process that serves as a basis for workflow specification. The model of a process enables users to understand what tasks, dependencies among tasks and roles are necessary to the process. The level of process abstraction in a workflow specification depends on its intended use. The workflow specifications provide understanding, evaluation, and redesign of processes at conceptual level, and describe the details of the processes at lower levels for workflow implementation.

Many process modelling methodologies, workflow specification techniques and languages have been proposed to deal with various business processes and workflow types. A "silver bullet" for all types of domains is considered unrealistic. Instead, methodologies and techniques should be suitable for their problem domains (Barros and Hofstede, 1998).

2.5.1 Workflow Process Modelling Methodologies

Process modelling methodologies can be divided into three main categories, namely communication-based, artifact-based and activity-based.

2.5.1.1 Communication-based Methodologies

The communication-based methodologies are based on the "Conversation for Action Model" (Winogard and Femando, 1986). The methodologies assume that the objective of business process re-engineering is to improve customer satisfaction. Every action in a workflow system has four phases based on communication between a customer and a performer, as illustrated in Figure 2-5.

Request is the stage when a customer requires an action to be performed. Negotiation is entered when both customer and performer agree on the action to be performed and define the terms of satisfaction. Performance is the stage where the action is performed according to the terms established. Acceptance is the stage when customer informs satisfaction (or dissatisfaction) with the result of performing action.

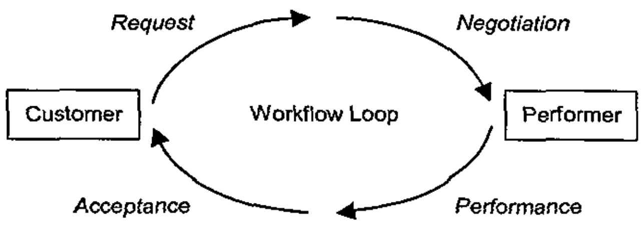


Figure 2-5. Conversation for Action Model.

There could be more than one workflow loop in a process model, each of which can be joined with other workflow loops to form a complete business process. The performer in one workflow loop can be a customer in another workflow loop. In this way a network of connected loops is generated recursively. As a result, an organisation can be seen as a network of a series of workflow loops that fulfil a business process.

Figure 2-6 gives an example of equipment procurement using communicated-based approach to define a process as a network of workflow loops.

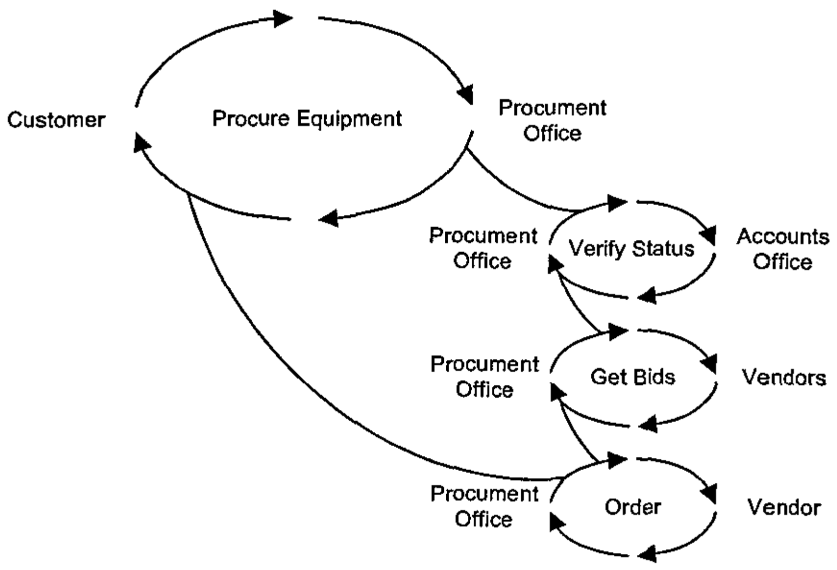


Figure 2-6. Communicated-based workflow of procure equipment process.

In Figure 2-6, the performance of Procedure Equipment loop requires three child loops. The three child loops are: (1) verifying of the account of the customer; (2) getting bids from vendors; and (3) ordering the equipment. While the procurement office serves as a performer in the main loop, it acts as a customer in the three child loops. The satisfaction of the three child loops represents the end of the performance stage of the main loop, followed with the acceptance stage.

The communication-based methodologies emphasise the customer, which assumes the objective of business process is to satisfy customer. However, they are not appropriate for modelling business process with objectives other than customer satisfaction. In addition, the approach does not support the development of workflow implementations for specifications (Georgakopoulos et al., 1995). Other specifications languages that are in different form of process representation is required to model the process for workflow engine execution.

The "Action Workflow Analyst" tool (Medina-Mora et al., 1992) from Action Technologies is based on the "Conversation for Action Model" as is the "Business Transformation Management" tool from Business Transaction Design (Marshak, 1994).

2.5.1.2 Artifact-based Methodologies

Artifact-based methodologies focus on the objects, such as data or information that are created, modified and used within a process. The modelling process is based on work products and their path through series of workflow activities. An example of equipment procurement using artifacts-based approach to define a process is given in Figure 2-7.

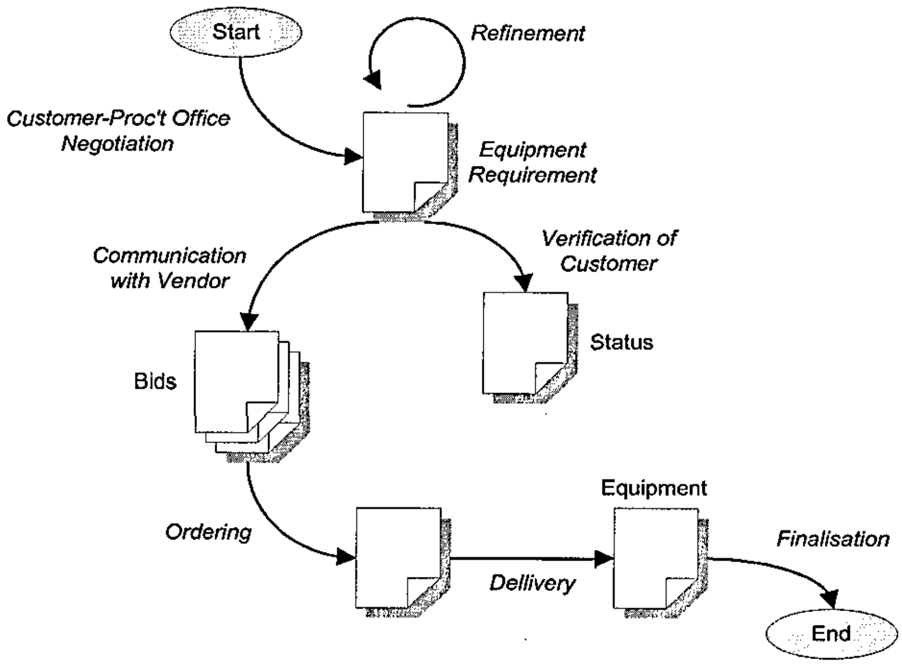


Figure 2-7. Artifact-based workflow of procure equipment process.

2.5.1.3 Action-based Methodologies

Action-based methodologies emphasise modelling the work instead of modelling the commitments among people. Workflows are modelled as a series of tasks with ordering constrain. Workflows may be decomposed into sub-workflows at lower level of abstraction. Unlike communication-based methodologies, they do not capture objectives such as customer satisfaction. The main advantage is that this model can be translated into a workflow specification as the activities are modelled.

Figure 2-8 provides an example of equipment procurement using activity-based approach to define a process. The activity Procure Equipment consists of three sub-activities namely Verify Status, Get Bids and Order. Arrows show the dependencies among the activities.

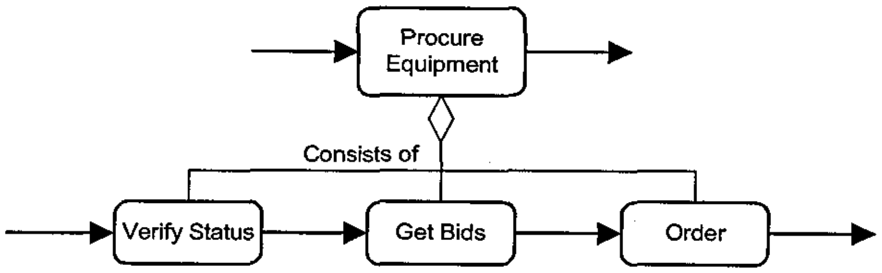


Figure 2-8. Activity-based workflow of procure equipment process.

The WfMC adopts activity-based methodologies for process modelling. This characteristic is reflected in their taxonomy of workflow concepts presented in Figure 2-9. In this figure, the concepts related to organisational and mission-critical knowledge are absent. The expanded view of organisational process might include representations of authority. Several research projects such as Task-based Process Management (TBPM) (Stader et al, 2000), address various aspects of organisation theory in workflow process.

Many commercial WfMS provide activity-based workflow models. For example, workflow models supported by InConcert (Marshak, 1997) (McCarthy and Sarin, 1993) are comprised of tasks. Each task may be further divided into sub-tasks. Each task has dependencies on other task at the same level. Task may also contain a description of the capabilities required of a performer. GTE's RAPID (Eckerson, 1994) also adopts activity-based methodologies. There are two workflow models in RAPID: a high-level workflow model for performing conceptual business process analysis and a lower-level model for describing the corresponding information process.

A high-level workflow model contains the tasks for a particular business process. These tasks can be ordered to indicate alternatives or parallel execution of business process steps.

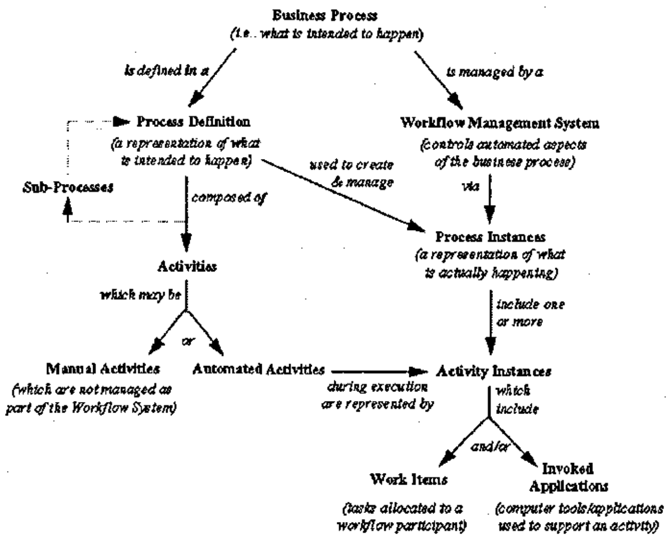

Business Process
(i.e.. what is intended to happen)
is defined in a
is managed by a
Process Definition
(a representation of what is intended to happen)
Workflow Management System
(controls automated aspects of the business process)
used to create & manage
via
Sub-Processes
composed of
Process Instances
(a representation of what is actually happening)
Activities
which may be
include one or more
or
Manual Activities
(which are not managed as part of the Workflow System)
Automated Activities
Activity Instances
during execution are represented by
which include
and/or
Work Items
(tasks allocated to a workflow participant)
Invoked Applications
(computer tools/applications used to support an activity)

Figure 2-9. WfMC's relationships among basic terminology (WfMC, 2000a).

2.5.2 Process Descriptive Views

It is noted that for any one process or workflow representation, all three methodologies can be used simultaneously. The concepts that are used for the purpose of process modelling include, or combine, the three process descriptive views, namely functional view, structure view, behavioural view (Christie, 1995).

2.5.2.1 Functional View

The functional view is focused on activities and the entities that flow into and out of these activities. This view is often expressed based on Data Flow Diagrams (DFD).

A DFD is essentially a static structure that expresses all possible data flow and data store interactions of a process. The strength of DFD is that its basic concepts are simple and general. Its decomposition feature provides comprehensibility. DFD is useful for the early phase of analysis where the broader functionality of a system is still being determined. An example of a procure equipment process modelled using DFD is given in Figure 2-10.

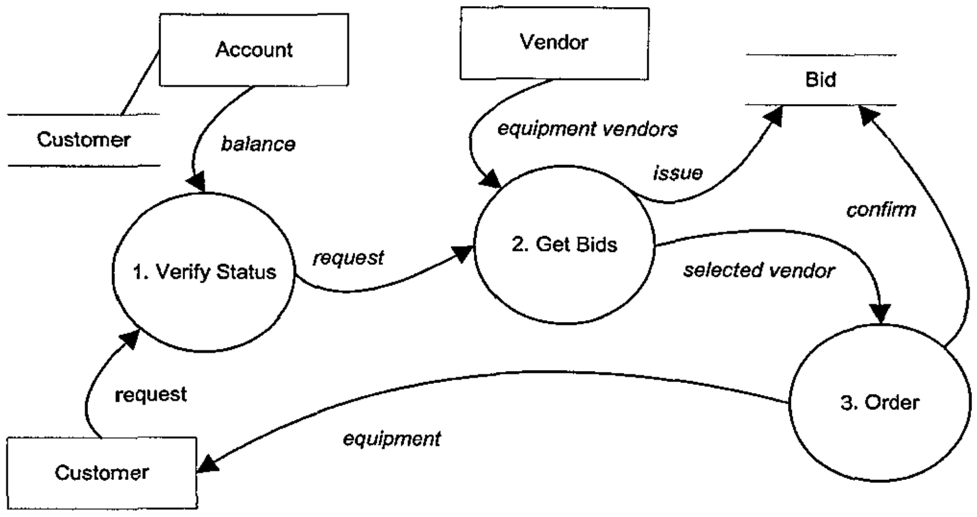


Figure 2-10. Data flow diagram of procure equipment process.

The process control, however, is implicit in detailed process specification meaning that the execution structure is hidden or unavailable. The absence of formal semantics is a critical omission (Opdahl and Sindre, 1993), which lead to ambiguities and inconsistencies. As a result, the precision that workflow specification requires cannot be achieved.

2.5.2.2 Structural View

Structural View is focused on the static aspect of the process. It captures the objects that are manipulated and used by a process and the relationships between them. This view is often expressed based on the Entity-Relationship Diagrams (ERD) or more comprehensive object diagrams that are used by various object-oriented methods, such as Object Modelling Technique (OMT).

Figure 2-11 is an example of procure equipment process described using OMT diagram. This object model can be interpreted as: Procurement Office and Accounts Office are the specifications of an Office. Customers can request equipment from Procurement Office and their Accounts are managed by the Accounts Office. Procurement Office is associated with Accounts Office for the purpose of customer verification. Procurement Office is associated with Vendors and base on the Bids the selected Vendor is used to provide the required equipment to Customer.

The relationship between the objects in a process can be easily captured using object-oriented modelling. Objects can be used to model workflow resources such as documents, processes and performers. The main advantage of object-oriented modelling is that objects provide a close mapping to real-life things and message passing is more powerful than the function call approach.

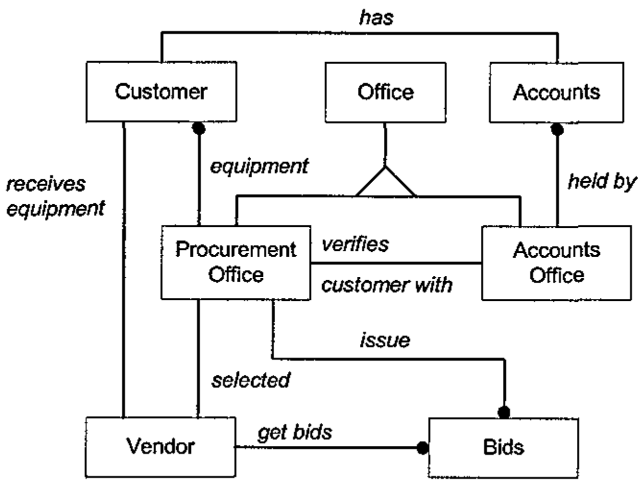


Figure 2-11. OMT object diagram of procure equipment process.

2.5.2.3 Behavioural View

The behavioural view is focused on the execution dependencies between processes. It captures the control aspect of the process model which is often expressed based on the state diagram or Petri Nets.

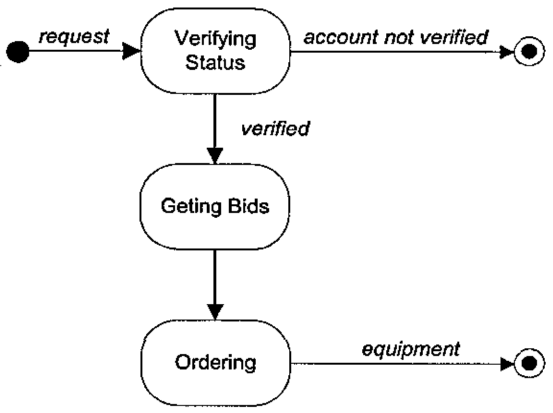


Figure 2-12. OMT state model integration for the procure equipment process.

An example of the procure equipment process defined using state diagram is given in Figure 2-12. The event *request* (in the state diagram) triggers the action of account verification to create a Bid object (having no previous state) in the state Verifying Status. The Bid object is stated Getting Bids after account is verified and stated Ordering after a vendor is selected and the order is in process.

On the other hand, Petri Nets (PN) can precisely capture behavioural aspects of process models in a formal and expressive way, with the ability to exhibit asynchronous and concurrent activities (Peterson, 1977). PN are widely used in the context of logistics and manufacturing production control. Through PN, a precise integration of data and process models is possible. High-level Petri Nets (HPN) are PN extended with 'colour', 'time' and 'hierarchy', which have been proposed to improve the expressiveness and readability of classical PN (W.M.P. van der Aalst, 1998).

The Behaviour Network Model (Kung, 1993) integrates DFD, ERD and PN together, which transfer behavioural aspects from higher levels of abstraction to the lowest level (Barros and Hofstede, 1998). DFD are used at high levels, and each process is transformed into a PN at lowest level. A PN is tightly coupled with an ER schema. Thus, PN specifications replace traditional structured English. At all levels of abstraction, model integration is process-centric.

Using graphical representation in behavioural process models, however, is problematic, even for basic specifications the process models will become cluttered. The use of abstraction/decomposition and conceptual specification languages can reduce the impact on the graphical representation. Using PN does not guarantee a formal foundation; formal semantics still have to be defined. Algebraic systems like Process Algebra provide an alternative mechanism for the definition of formal semantics (Barros et al., 1996).

2.5.3 Workflow Specification

A workflow needs to be specified in some way to enable a workflow engine to understand and execute it. A number of workflow specification languages have been defined for representing process models. However, almost all vendors use proprietary formats for process specification, which makes it difficult to migrate a process definition from one workflow engine to another. Thus, several initiatives have emerged that focus on the development of a standard language of the workflow specification.

Current workflow specification languages describe a workflow in either text format or visual diagram. The advantage of text-based languages is the ability to interchange a process definition among different workflow engines which adopt the same language. Text-based languages enable different visual presentations of a process model which will be turned into a text format for execution.

On the other hand, a standard visual presentation of process model enables the integration of different tools from different suppliers at information system (IS) level. However, the lower level process specification converted from visual diagram for workflow engine interpretation in these tools may vary, and therefore a conversion is also necessary when exchange a process definition between two different workflow engines. A workflow system may adopt two workflow specification languages, one for process model presentation and another for process interchange.

Following are six prevalent meta-languages (Michael and Joerg, 1999) for process specification. A meta-language is a superset of constructs that can be found in process modelling languages and that can be used to map concepts from one process modelling language to another.

2.5.3.1 XML Process Definition Language

The XML Process Definition Language (XPDL) (WfMC, 2002) is an extension of the Workflow Process Definition Language (WPDL) which was first introduced by WfMC in 1994. XPDL uses eXtended Markup Language (XML) to represent a process, forming a common interchange standard that enables different workflow products exchange the process definition through a batch import/export procedure. The keywords of XPDL are based on the terms defined in the WfMC glossary.

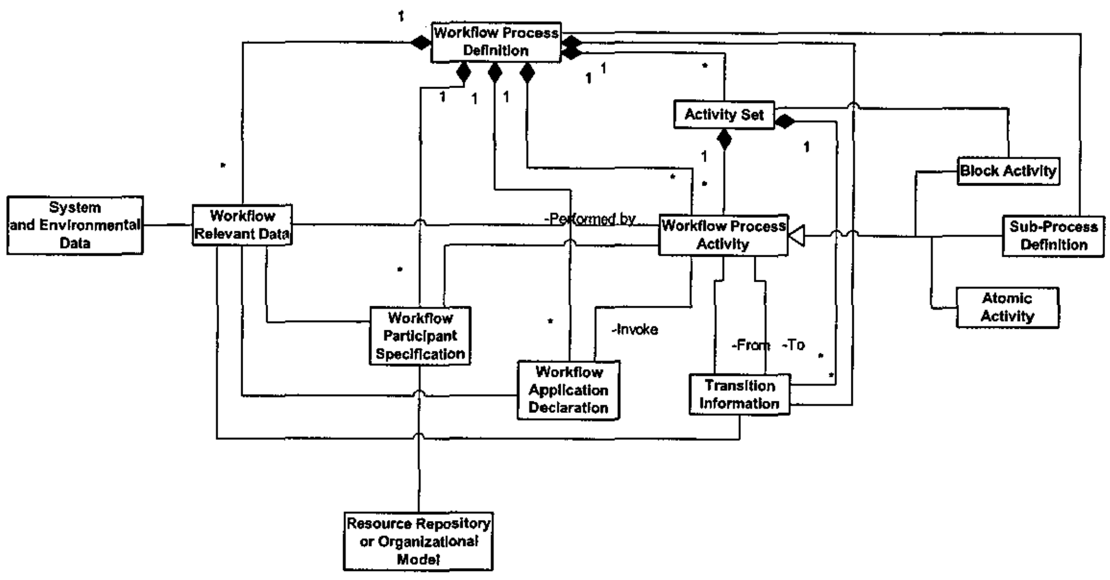

Workflow Process Definition
Activity Set
Block Activity
System and Environmental Data
Workflow Relevant Data
-Performed by
Workflow Process Activity
Sub-Process Definition
Atomic Activity
Workflow Participant Specification
-Invoke
Workflow Application Declaration
-From
-To
Transition Information
Resource Repository or Organizational Model
1
*

Figure 2-13. Workflow process definition meta-model (WfMC, 2002).

The language is design based on a meta-model which defines the basic components that have to be supported by a workflow engine. The meta-model is depicted in Figure 2-13. The core concept of XPDL is a Workflow Process Definition that is comprised of one or many Workflow Process Activities. The ordering of activities is determined by Transition Information elements that connect single activities. For more complex routing, a Transaction may relay on Workflow Relevant Data. The organisational model and system environmental mode are specialisation of the Workflow Participant Specification and thus are not elementary components.

2.5.3.2 Process Interchange Format

The Process Interchange Format (PIF) is built on top of Ontolingua (Gruber, 1992) and Knowledge Interchange Format (KIF) (Genesereth, 1999) that is designed for the interchange of knowledge among separate computer systems. The advantage of PIF is its ability to explicitly represent the similarities and differences among related processes and to easily find or generate sensible alternatives on how a given process could be performed (Lee et al, 1998).

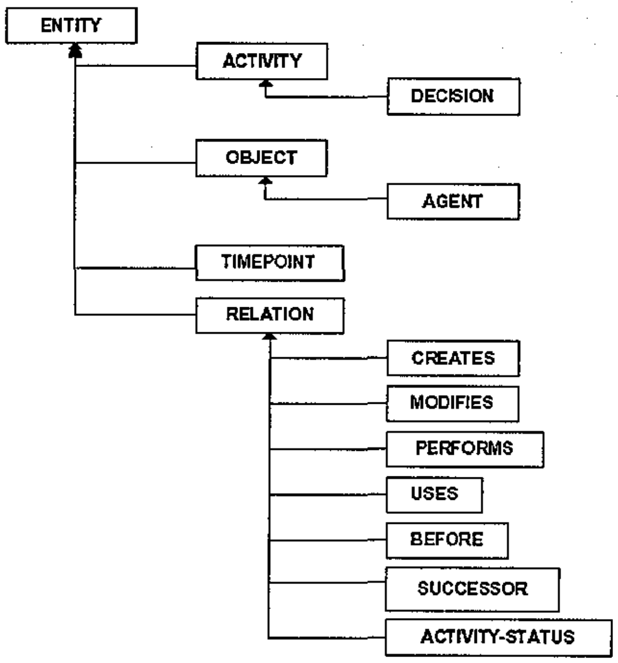


Figure 2-14. Hierarchy of PIF core components (Lee, 1999).

The description of a process in PIF is based on a set of frame definitions and each of which denotes an entity type that can be instantiated. These types are arranged in a hierarchy as shown in Figure 2-14. Each type has a set of predefined attributes that define various aspects of the instance. The attributes of each type can be inherited into its sub-types and their instances.

The PIF is a powerful exchange platform for process models. Through its modular design, it can be easily extended to accommodate the needs of workflow process modelling.

2.5.3.3 Process Specification Language

The Process Specification Language (PSL) steams from a National Institute of Standard and Technology (NIS) project to investigate a common unifying model of processes. The core concept of PSL for the mapping between two applications is to represent the processes of these two applications using KIF and transformed into processes that conforms to the PSL ontology.

The basic components of PLS are:

Activities – the tasks to be performed.

Objects – the resource, such as human resources or machines, or states, such as pre and post activity state.

Timepoints – the temporal relationships between activities or the duration of performance.

Through this intermediate process, therefore, the process definitions can be exchanged and understood by both applications.

2.5.3.4 Generalised Process Structure Grammars

While WPDL, PIF and PSL represent Input-Process-Output (IPO) process modelling languages, Generalised Process Structure Grammars (GPSG) (Glance et al., 1996) adopts a constraint-based approach to process modelling. For the modelling of process, a specific grammar is constructed, which contains the legal elements of the process together with their relationships. The grammar spans a process space that contains only the vital constraints and construction rules, there are no restriction by default until the time of process enactment. Each GPSG can contain two kinds of rules:

Activity-centric rules. These rules separate a process goal into sub-goals together with execution constraints.

Document-centric rules. These rules describe the data object used in process.

GPSG-based process definition provides more flexibility during process enactment, because the processes are not executed following a strict set of control flow paths and conditions, but rather emerge within process space opened by the process-specific grammar.

Freeflow (Dourish et al., 1996) is one of prototype systems using the GPSG approach for process representation. The complexity of the constraint-based workflow model restricts the process specification within textual representation approach, while graphical representation seems difficult because of many possible paths of the process at run-time. The process specification of transactional workflows using GPSG will result complex grammar, therefore it may be done more efficiently using IPO-based modelling language.

2.5.3.5 Petri Nets

The concept of Petri nets that was first proposed in Carl Adam Petri's thesis submitted in 1962 (Petri, 1962). A Petri net is a graphical and mathematical modelling tool. It consists of places, transitions, and arcs, as illustrated in Figure 2-15.

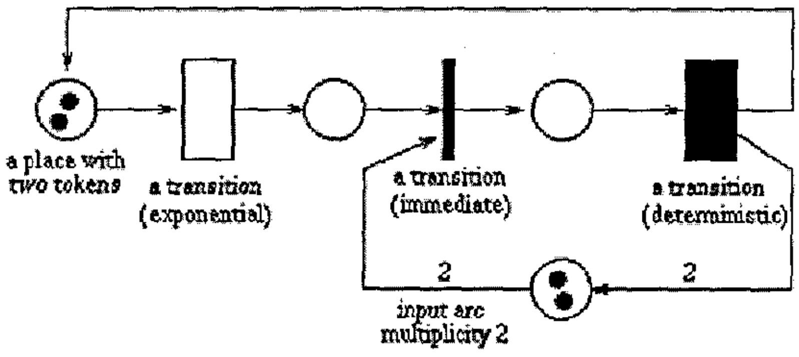


Figure 2-15. Example of a Petri Net.

Input arcs connect places with transitions, while output arcs start at a transition and end at a place. Places can contain tokens. The current state of the modelled system is given by the number of tokens in each place. Transitions are active activities, which are used to model activities. The current state of the modelled system (the marking) will be changed when the modelled activities occurs (the transition fires). The transitions are only allowed to fire when all the pre-conditions for the activity are fulfilled (there are enough tokens available in the input places). When a transition fires, it moves tokens from its input places to its output places. The number of tokens move depends on the cardinality of each arc.

Petri nets are able to precisely and fully model the workflow processes. The relationship of the components between workflow and Petri net are shown in Table 2-1.

The main distinction between Petri-net and other process modelling languages is that Petri-net is state-based but others are event-based. In event-based methodology, the tasks are modelled explicitly and the states between subsequent tasks are suppressed. There are several reasons for using a state-based description (W.M.P. van der Aalst, 1996) in which the provision of a clear distinction between the enabling of a task and the execution of a task is the key motivation.

Workflow	**Petri Net**
Activity	Transition
Subject / Role	Attribute of Transition
Execution of Activity	Firing of Transition
Control Flow	Flow Relation
State of Workflow	Marking of Petri Net

Table 2-1. Workflow and Petri Net.

2.5.3.6 Unified Modelling Language

The Unified Modelling Language (UML) defines a standard notation for object-oriented systems. The UML is comprised of different diagram types, each of which provides the design of different views of a system. However, no single notation in UML can be used to model all aspects of a workflow model. Instead, several diagram types have to be employed in order to model all aspects of workflow processes (Hruby, 1998). These diagram types include:

Use Case Diagrams. These diagrams are used to depict the interaction of a system with its environment. The workflow process use cases can be used to model the interaction between process and actors.

Sequence Diagrams. These diagrams depict the temporal and logical order of activities and involved participants. The different actors within a workflow process can be arranged in parallel lanes in sequence diagrams, therefore the interaction between participants can be modelled in workflow processes.

Collaboration Diagrams. These diagrams depict the interaction between actors and use cases are described in terms of the messages that are sent between the different elements of the diagram. In addition, collaboration diagrams also depict the ordering and directed relationship of messages, thus it can be seen as an extension of Use Case Diagrams.

State Transition Diagrams. These diagrams depict all possible states of a use case and the transitions between these states. In workflow management, a state transition diagram can be used to describe the possible starting and ending point of a workflow model together with legal transitions between states. In addition, state transition diagrams can also be used to describe the transformation of process objects in a workflow process.

Activity Diagrams. These diagrams display all possible paths of action between activities. The activity diagrams depict relationships between activities. The transition between two activities is only active when the proceeding activity has finished and an optional constraint at the transition is evaluated to be true.

The UML offers a variety of diagram types that accommodate several aspects of workflow process modelling. However, there are still a number of shortcomings in UML with regard to workflow process modelling (Wiegert, 1998), and the necessity of using different diagram types to model all aspects of a workflow process makes the current UML standard difficult to be used in current workflow tools. With the development of new versions of the UML this situation is likely improve overtime.

2.6 Workflow Engines and Workflow Implementation

Workflow engines provide the run-time environment for activating, managing, and executing workflow processes. There are three kinds of paradigms employed, namely scheduler-based, data-flow and information pull paradigms (Cichocki, 1998).

Scheduler-based paradigms. These paradigms are the most common and are supported by WfMC, where workflow engines instantiate process specification, decomposes it into taskable activities, and then allocates activities to be performed. The implementation and deployment of the scheduler-based approach to workflow can be seen in terms of state transition machine. The process instances change state in response to workflow engine decisions or external events.

Data-flow paradigms. The workflow in these paradigms is referred to as an information repository in which the data is passed between process components according to sets of rules. Thus, the data-flow paradigms are especially suitable for the workflows that can only be partially specified, dynamic and goal oriented.

Information-pull diagrams. These paradigms are developed base on the network and information management, where the requirement for information drives the creation and enactment of processes. This kind of diagrams is relatively new in the workflow domain.

The workflow engines manage the transition of process states, selecting process to be instantiate, initiating activities by scheduling them to work lists, and controlling and monitoring the whole progress. In addition, the contemporary WfMS should be able to perform enactment and management of process within highly dynamic and uncertain environments. However, most of the workflow engines are limited that are only able to provide support for predicable process and handling expectable events (Georgakopoulos et al., 1995) (Han et al., 1998) (Peter et al., 1999).

2.6.1 Process Selection and Instantiation

The selection and instantiation of process templates is one of the key responsibilities of the workflow engines. In response to some stimuli such as triggering event, the workflow engines will select a suitable process from the template library. The workflow engines also handle the instantiation of the relevant processes. There may have more than one applicable process against the same stimulus; thus, the comparison of the triggering conditions must be performed in order to select the most appropriate one.

2.6.2 Task Allocation

Once a process is selected and instantiated, the process manager forwards activities to a worklist handler to control the task of selecting suitable performer for the activities. Each activity is assigned to a process performer according to its capability and availability, with temporal and ordering constraints of the activity. The worklist handler can be implemented as a simple in-tray of work items, or a complex set of agenda-based load balancing and interrelated work assignments with complex supervisory roles. Allocating tasks in workflow management involves scheduling techniques. The techniques such as straightforward enumerative or heuristic-based algorithms have been widely employed. Thus, the workflow engines take a centralised role in coordinating the operation of performers.

The complexity increases when one of more workflow engines make up a workflow domain that provides a homogeneous process execution environment. The workflow enactment service is required to support the execution of specific workflow over one or more workflow engines, which may be in one or more separate domains. Thus, more sophisticated approaches that provide robust reactive scheduling is critical to accommodate this situation. For example, there are some works on exploiting the dependencies between activities within the workflow task allocation problem (Attie et al, 1996).

2.6.3 Enactment Control, Execution Monitoring, and Failure Recovery

The workflow engines must maintain all the knowledge and internal control data to identify the states of each instantiated activities, transition conditions, and connections among processes. The WfMC defines two types of data, as shown in Figure 2-3, which is relevant to the control and monitoring of workflow processes:

Workflow Control Data. These internal data represent dynamic state of workflow system and process instances. It is internal information managed directly by the workflow engines and is not normally accessible by applications.

Workflow Relevant Data. These external data are used by WfMS to determine when to enact new processes and when to transit among states within enacted processes. It may be manipulated by workflow applications as well as by the workflow engine.

The WfMS provide support for monitoring of process execution and responding to the events appropriately when detected. Some WfMS provide Graphic User Interfaces (GUI) that can present different view of workflow execution. GUI are provided for both graphical workflow specification and graphical task specification. Graphical workflow specification languages support the iconic representation of workflow tasks and the ability to sequence those tasks graphically. Managers can use these monitoring tools to access workflow statistics such as task completion times, workloads, and user performances as well as to generate reports and provide periodic summary of workflow executions.

Contemporary WfMS have limited abilities to recover from failures. Workflow failures may be caused by a process that cannot be completed correctly or on time, or by an unexpected event in the environment. The work in this area at present has focused on the use of transactional methods to ensure data correctness and reliability, especially when concurrent activities must share data.

The concurrency control is essential in many situations where two or more user access same data object. However, commercial WfMS adopt different approaches to concurrency control. For example, InConcert workflow supports a form of check-in and check-out on documents therefore users can lock data to preclude concurrent access by other users.

Other WfMS, such as Lotus Notes, provide access to the same data object concurrently. When one user update the data object, the existing version will be saved into the versions achieve which would be merged later by users. The rationale for this approach is the assumption that data object updates are rare. Thus, the consistency can be maintained by users through viewing the version history and decide which one they want to keep. However, this approach is not suitable for WfMS as there hundreds objects are accessed by multiple users. This will result in thousands of object versions all requiring user intervention. Besides time consuming, it also poses a problem for conflicting information that cannot be correctly merged.

Still other WfMS, such as StaffWare, use a pass-by-reference and pass-by-value approach for concurrency control. Documents that can be shared among multiple users are passed by reference (pointer).

There are two popular techniques in workflow recovery, namely checkpoint and rollback.

Checkpoint. These schemes rely on the use of log files that store safe states. Thus, system can be reset to the latest available safe state when problem occurs, and then processing restarted.

Roll back. These schemes relay on the availability of invertible actions. The system will perform corresponding inverse actions in reverse order when problem occurs, until the expected former stage is achieved.

These recovery techniques are suitable for data-driven workflow. However, they are inadequate for the domains that involve highly dynamic environment where changes continuously and unpredictably, therefore actions may not be undoable. Thus, more powerful and flexible methods for failure recovery are required in order to be effective in a broader range of application domains.

2.7 Workflow Standards

Today, many software vendors provide workflow management (WfM) products, and there is a continuous introduction of more products into the market. Individual product vendors focus on particular functional capabilities where users have adopted particular products to meet specific application needs. It has been recognised that all WfMS have some common characteristics which enable them potentially to achieve a level of interoperability through the use of common standards for various functions. With the emergence of business to business (B2B), business processes have a heavy dependence on widely-accepted and viable standards for their successful deployment. There are a number of attempts to standardise workflow in many aspects. The following introduces the latest works of workflow standardisation.

2.7.1 Workflow Management Coalition's Reference Model

The Workflow Management Coalition (WfMC) was founded in 1993 as a non-profit, international organisation of workflow vendors, users, analysts and university/research groups. WfMC has proposed a framework for the establishment of workflow standards, which includes five categories of interoperability and communication standards that will enable multiple workflow products to coexist and interoperate within a user's environment. The illustrations in this section are taken from the standards published by WfMC.

WfMC's workflow reference model is developed from the generic workflow product structure by identifying the interfaces within this structure, which enable products to interoperate at a variety of levels. The reference model with its major components and interfaces is illustrated in Figure 2-16.

The interface around the workflow enactment service is the workflow application programme interface (WAPI) – workflow APIs and Interchange format. WAPI denotes a unified service interface to support workflow management functions across the five functional areas, rather than five individual interfaces.

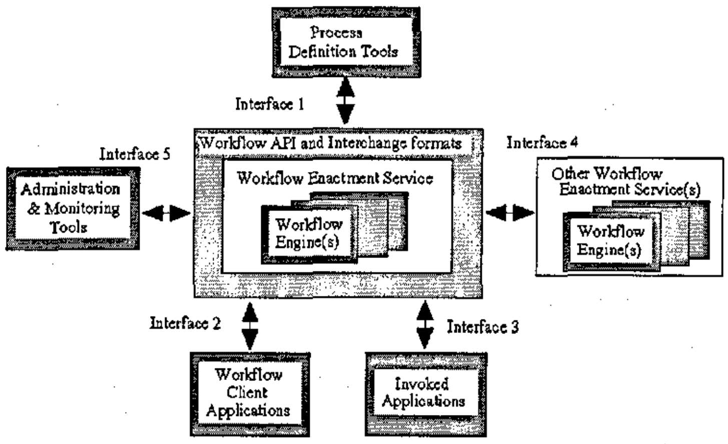

Process Definition Tools
Interface 1
Workflow API and Interchange formats
Workflow Enactment Service
Workflow Engine(s)
Interface 5
Administration & Monitoring Tools
Interface 4
Other Workflow Enactment Service(s)
Workflow Engine(s)
Interface 2
Interface 3
Workflow Client Applications
Invoked Applications

Figure 2-16. WfMC's Workflow reference model.

WfMC's standardisation is ongoing. Some of the interfaces are rudimentary specifications that have been published and will be withdrawn when new version is derived.

2.7.1.1 Workflow Enactment Service

The workflow enactment service is a software service that may consist of one or more workflow engines in order to create, manage and execute workflow instances. Applications may interface to this service through the WAPI.

The workflow enactment service may be centralised or functionally distributed, homogeneous or heterogeneous. Process definitions are imported and exported through interface 1: Process Definition Tools. Interaction with external resources accessible to the enactment service is via Interface 2: Workflow Client Application Interface. Specific applications to undertake a particular activity is activated by interface 3: Invoked Applications. Interface 4: Workflow Interoperability Interface enables a standardised interchange across workflow engines. Finally, Interface 5: Administration & Monitoring Tools provides access to common administration and monitoring functions, possibly in heterogeneous environment.

2.7.1.2 Interface 1: Process Definition Interchange

The output from a process modelling activity is a process definition, which can be interpreted at run-time by a workflow enactment service. The nature of the interface is an interchange format and API calls, as depicted in Figure 2-17. Interface 1 provides the support for exchange of process definition information over a variety of physical or electronic interchange media.

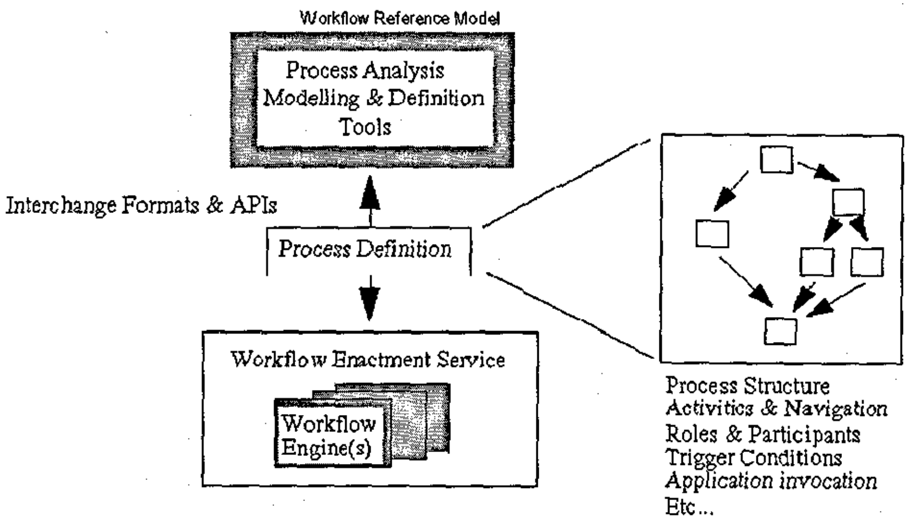


Figure 2-17. Process definition interchange.

By using this standardised from, the build-time and run-time environments are separated, enabling a process definition generated by one modelling tool to be used as input to a number of different workflow run-time products. In addition, it offers the potential to export a process definition to several different workflow products that could cooperate to provide a distributed run-time enactment service. The XML Process Definition Language (WfMC, 2002) is proposed which takes advantage of eXtended Markup Language (XML) to support process definition import and export over any transport mechanism.

2.7.1.3 Interface 2: Workflow Client Application Interface

Worklist is a well-defined interface which handles interaction between the client application and the workflow engine. It presents to the end user the work items assigned by workflow engine that he is responsible for. It may also invoke tools to present tasks and related data with corresponding deadlines, status etc.

The Worklist handler is a generic application that may be part of a workflow system. It may also be customised by a workflow user to integrate it with other applications or services, like e-mail, into a common desktop environment. The overall approach to the workflow client application API is illustrated in Figure 2-18. The work items in worklist may contain items that relate to several different active instances of a single process and/or individual items from instantiation of several different processes. In addition, it may be interacting with several different workflow engines and several different enactment services.

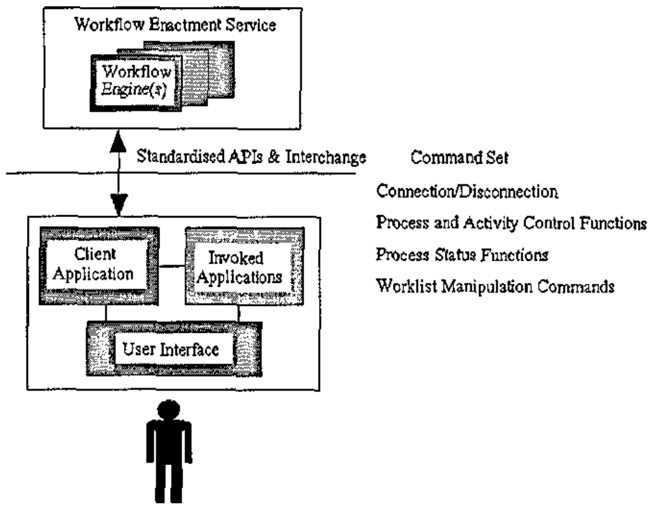


Figure 2-18. Client application interface.

2.7.1.4 Interface 3: Invoked Application Interface

For invoking external applications, it is assumed that no particular WfMS will have sufficient logic to understand how to invoke all potential applications that might exist in a heterogeneous product environment. Thus, WfMC's initial work focuses on developing a catalogue of interface types, together with a set of APIs for use in future workflow specific applications, as illustrated in Figure 2-19. The application agents are used to contain a variety of method invocation behind a standard interface into the workflow enactment service, so that the invocation of applications in heterogeneous environment becomes possible.

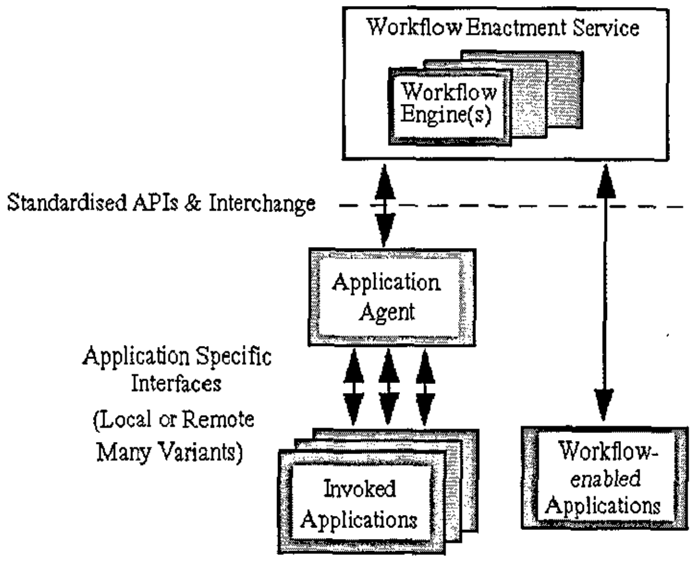

Workflow Enactment Service
Workflow Engine(s)
Standardised APIs & Interchange
Application Agent
Application Specific Interfaces
(Local or Remote Many Variants)
Invoked Applications
Workflow-*enabled* Applications

Figure 2-19. Invoked application interface.

2.7.1.5 Interface 4: Workflow Interoperability Interface

A key objective of the WfMC's standardisation work is to allow workflow system produced by different vendors to pass work items seamlessly between one another. In order to achieve this objective, a variety of interoperability scenarios have been developed and described. Interoperability is described at a number of levels, ranging from simple task passing to full workflow application interoperability with interchange of process definitions and workflow relevant data. In addition, a number of solutions against each scenario are introduced. The general nature of the information and control flows between heterogeneous workflow systems is illustrated in Figure 2-20.

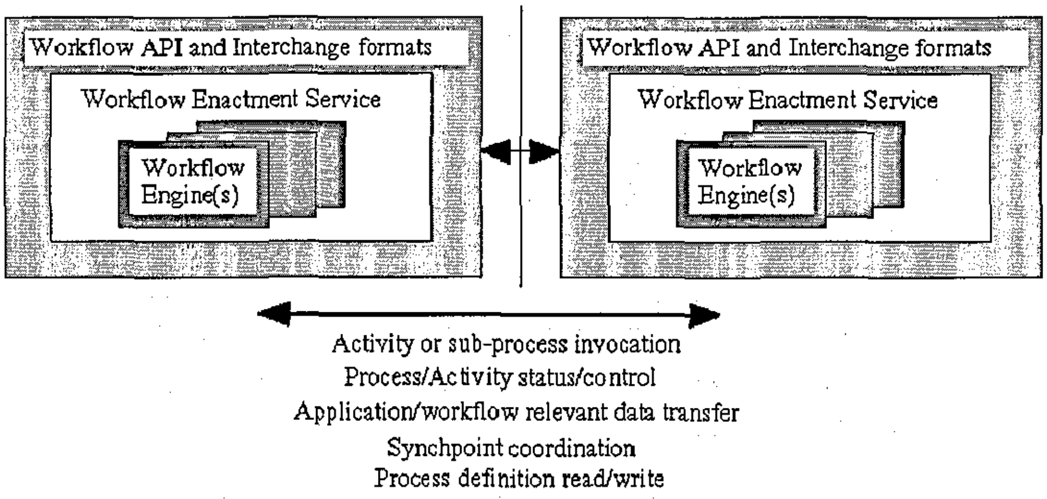

Workflow API and Interchange formats
Workflow Enactment Service
Workflow Engine(s)
Workflow API and Interchange formats
Workflow Enactment Service
Workflow Engine(s)
Activity or sub-process invocation
Process/Activity status/control
Application/workflow relevant data transfer
Synchpoint coordination
Process definition read/write

Figure 2-20. Workflow interoperability interface.

2.7.1.6 Interface 5: Administration & Monitoring Tools Interface

The final area of the proposed reference model is a common interface standard for administration and monitoring functions that will allow one vendor's management application to work with another's engine(s). This will provide a common interface that enables several workflow services to share a variety of common administration and system monitoring functions, as illustrated in Figure 2-21.

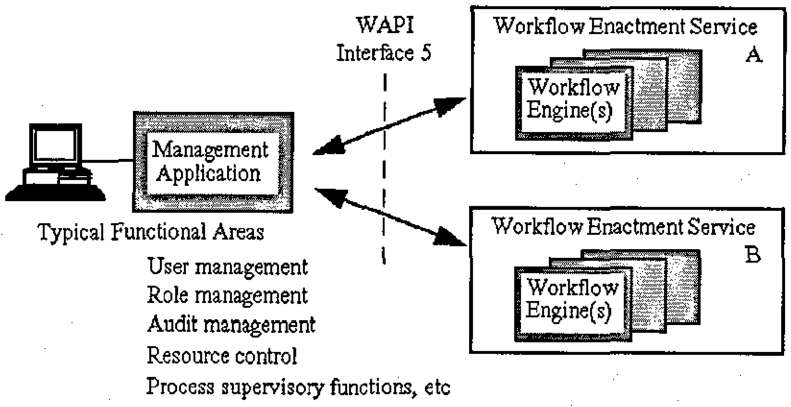


Figure 2-21. Systems administration & monitoring interface.

2.7.2 Workflow Management Coalition's Workflow Interoperability

The WfMC Workflow Interoperability standard (WfMC, 2000b) defines an abstract protocol for peer-to-peer interaction of workflow enactment services across different business domains, such as an inter-enterprise workflow. The standard concerns only the interoperability on the process level; internals of a process instance need not to be aware across business domains.

According to the standard, a set of workflow process attributes define the operating context and another set of attributes present results produced during the process lifetime, and another property represents the current state of the process, including running, completed, and terminated.

The standard defines three types of interactions namely Request, Acknowledgement and Response between a service requester and a service provider. The request operations include:

CreateProcessInstance. These requests create new instances of workflow process to be controlled by the service provider.

Get or ChangeProcessInstanceState. These requests obtain or change the current remote process state.

Get or SetProcessInstanceAttribute. These requests provide access to the values of the context of a process or result data attributes.

An Acknowledgement is used in asynchronous implementations by a service requester receiving a Wf-XML message to inform the service provider that the message has been received.

The service provider answers each request with a matching response. The service provider can also send notifications to the service requester, including:

ProcessInstanceStateChanged. These notifications indicate that the state of the remote process was changed due to some event in the service provider's domain.

ProcessInstanceAttributesChanged. These notifications indicate that the context or result data of the remote process was changed.

WfMC also propose a binding of the standard which uses asynchronous interaction via email as the transport with Multipurpose Internet Mail Extensions (MIME) coding of the information to be exchanged. The MIME binding of standards (WfMC, 2000b) specifies the protocol for exchange interoperability between service requester and service provider via email, taking into account e-mail's unreliability as a communication vehicle.

2.7.3 Object Management Group's Workflow Facility

The Object Management Group (OMG) is an international organisation with over 800 members. Founded in 1989, OMG promotes the theory and practice of object-oriented technology in software development, defines and standardise a common architecture framework across heterogeneous hardware and software platforms, so called the Object Management Architecture (OMA) (OMG, 1995). WfMC and OMG attempt to coordinate their activities to integrate workflow and CORBA technology. In 1997, a Request for Proposal (RFP) (OMG, 1997) for a Common Object Request Broker Architecture (CORBA) based Workflow Management Facility (WfMF) was issued by OMG., and a joint submission so called jFlow specification was adopted in 1999 which is based on standards and technology interfaces defined by WfMC.

The WfMF specifies interfaces of a framework for implementation of distributed workflow applications, enabling interoperability of workflow process components, monitoring of workflow execution and association of workflow components to resources involved in a workflow process. The core workflow interfaces are defined in the Workflow Model module presented in Figure 2-22 in UML class and object interaction diagrams and specified by Interface Definition Language (IDL) interfaces.

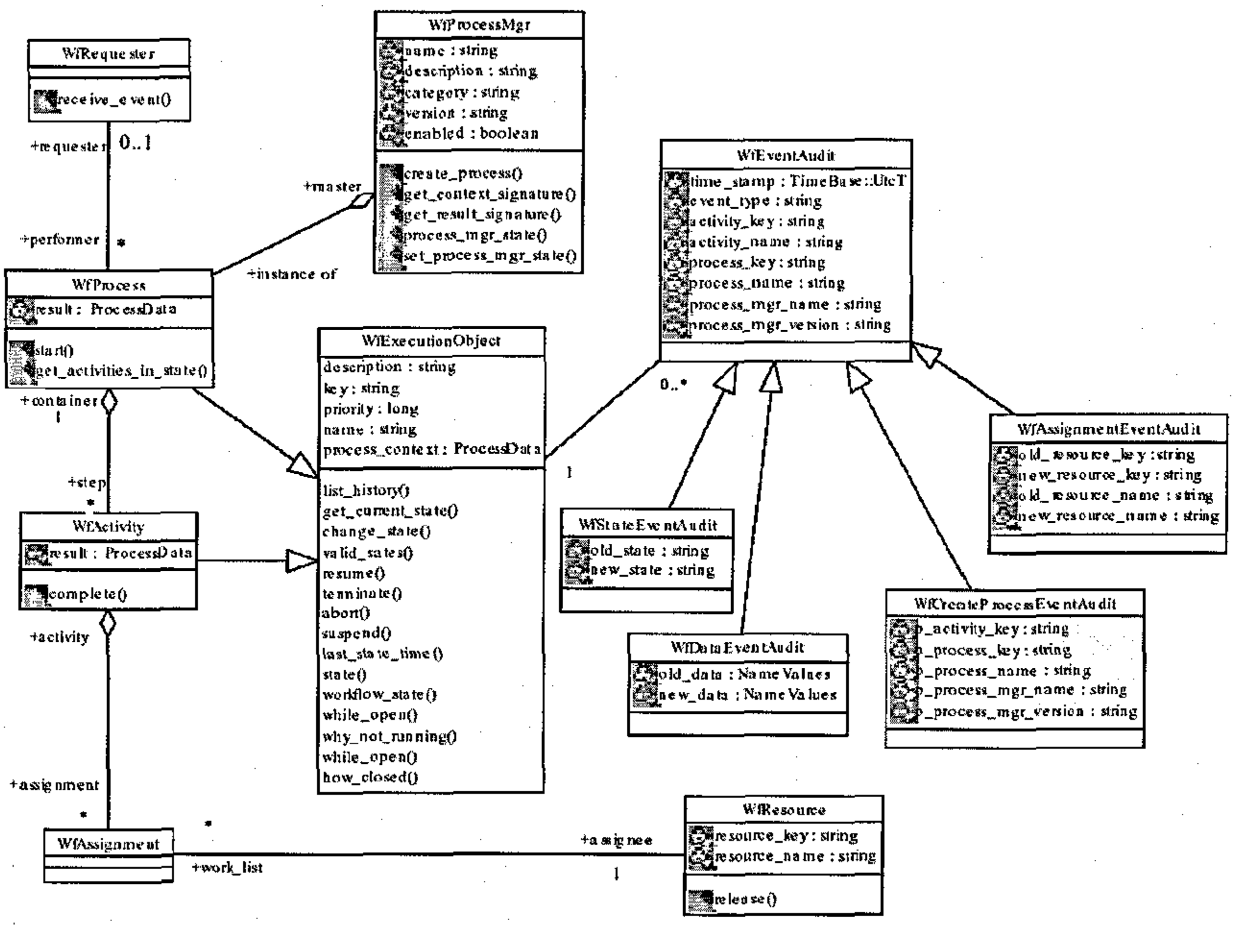


Figure 2-22. Joint Workflow Management Facility Model (OMG, 1998).

The primary interfaces relevant to workflow interoperability that concerns mainly on service requester and service provider are:

WfRequester. It represents the requests for some work to be done. As it provides operations to be used by the process to propagate status update, it has a direct concern with the execution and results of a workflow process.

WfProcess. It performs work requests and provides operations to control execution of the work request and to observe its state.

WfProcessMgr. It presents a template for a specific workflow process which is used to create instances of workflow process.

WfActivity. It is a step in a process that is associated with a single WfProcess. There are can be many active WfActivity objects in a WfProcess at the same time. A WfActivity represents a request for work in the context of the containing WfProcess.

WfMF also defines interfaces for worklist handling (WfAssignment and WfResource) and process auditing (WfEventAudit). The standardisation of workflow process definitions is not included in WfMF, but another RFP for extending WfMF to standardise these definitions is in development.

The OMG workflow standard defines a unified object model covering the different WfMC standards, which provides the base for future evolution of the WfMC standards. Examples include the Simple Workflow Access Protocol (SWAP) and Wf-XML message set.

2.7.4 Simple Workflow Access Protocol

The SWAP proposal (Swenson, 1998) attempts to define an Internet-based workflow access flow protocol to control and monitor workflow process instances. The basic idea of the SWAP proposal is to render the interaction between components of workflow applications as XML-encoded, providing a platform for the integration of message-based enterprise application.

The SWAP object model defines four types of Internet resources namely Observers, ProcessInstances, ProcessDefinitions, and Activities. These correspond to the WfRequester, WfProcess, WfProcessMgr, and WfActivity interfaces in the WfMF.

In order to achieve workflow and tool integration on the Internet, SWAP defines a Hypertext Transfer Protocol (HTTP) based protocol to enable the task request and task execution to take place at workflow engines located at different Internet locations. The information in the interactions between the two parties is Extensible Markup Language (XML) encoded.

The SWAP proposal was presented to the Internet Engineering Task Force (IETF) at its 43rd meeting in 1998. SWAP was successfully demonstrated in a prototype linking the U.S. Government Joint Computer-aided Acquisition and Logistics Support program (JCALS) workflow engine and IBM MQSeries workflow engine in different areas within the General Dynamics Electic Boat Division offices (Computer Sciences Corp., 1998).

2.7.5 Workflow Management Coalition's Wf-XML Specification

The Wf-XML specification (WfMC, 2001) addresses the issues of interoperability between workflow systems by taking advantage of XML standards. It is based on the WfMC's reference model, OMG's WfMC and SWAP. Wf-XML provides a structured and well-formed XML message set encoding for both synchronous and asynchronous message-handling that is independent of the particular transport mechanism. Business data associated with an inter-organisational process would be passed as attachments in XML format, as show in Figure 2-23.

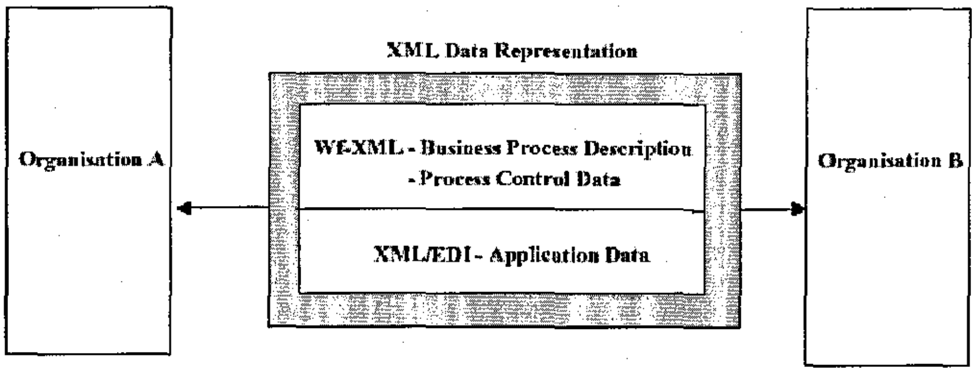


Figure 2-23. WfXML Data Representation

Wf-XML specification does not define a standard for business process interfaces; rather, it provides a language and protocol that enables interaction among such processes. Wf-XML can work with HTTP or a number other transport mechanisms, including email and direct Transmission Control Protocol over Internet Protocol (TCP/IP) connection, Simple Mail Transfer Protocol (SMTP) or Message Oriented Middleware (MOM).

The key benefit of Wf-XML is summrised by Jon Pyke, the Chair of the WfMC and Chief Technical Officer (CTO):

"The release of this proven XML-based standard provides organisations with an interoperability standard that facilitates a consistent method of interfacing between disparate workflow engines. This means that organisations will be able to simply and effectively implement Business to Business process integration as well as maximising their existing investment in BP technology and legacy applications... There is no doubt that take up of this standard will enhance the availability, quality and timeliness of management information regardless of location and technology deployed."

2.8 Chapter Summary

This chapter gave a presentation of the workflow technology area, focusing on its underlying technologies. It first, introduced some important workflow terms, concepts, workflow characteristics, and Ader's classification scheme of workflow products. It further provided a comprehensive discussion on the process modelling, the underlying concepts and workflow specification languages. From which, it can be seen that the approach of process modelling is application domain dependent. As each of WfMS within the market focuses on particular functional capabilities, their approaches on process modelling are different. The chapter proceeded to introducing workflow engines and their general functionalities. Finally, the standardisation works in workflow community was given.

WfMC proposed a framework for the establishment of workflow standards, which includes five categories of interoperability and communication standards that will enable multiple workflow products to coexist and interoperate within a user's environment. To enable the interoperability among different workflow engines, WfMC's workflow interoperability standard defines an abstract protocol for peer-to-peer interaction of workflow enactment service across different business domains. As the earlier standards of WfMC are not concerned with object-oriented standards, a joint submitted WfMF called jFlow specification was proposed in response to a Request for Proposal from OMG. WfMF proposes an object-oriented workflow model which specifies interfaces for implementation of distributed workflow applications. With the emergence of the Internet and B2B businesses, SWAP made an attempt to define a workflow access flow protocol to control and monitor process instances over the Internet. By taking advantage of XML, Wf-XML enables the business processes and their data to be passed among workflow engines regardless of the location and technology deployed.

Chapter 3

Workflow Adaptation and Enabling Technologies

"In order to form an immaculate member of a flock of sheep one must, above all, be a sheep."
- Albert Einstein

3.1 Introduction

The change in today's business process is a norm, not an exception. It occurs at every level in an organisation structure. To support current business process, WfMS needs to address these changes.

This chapter introduces workflow adaptation and considers various enabling technologies. It is organised as follows: §3.2 surveys four workflow systems which demonstrate the current best practices of workflow flexibility; §3.3 introduces Han's framework of workflow adaptation. The requirements of handling workflow adaptation at each level in the framework are identified; §3.4

presents and discusses the five technologies in the context of enabling adaptive WfMS; finally, a chapter summary is given in §3.5.

3.2 Survey of Adaptive Workflow Products

The following are four adaptive workflow products which adopt different approaches to flexible process support. The first two are classic workflow systems, InConcert Workflow and TeamWare Flow. Their success in turning from research prototypes into two of the most famous commercial workflow products in the market indicates that their process models possess vital flexibilities required by nowadays business processes support systems. The rest are two recent research prototypes demonstrating two types of cutting edge technologies. TBPM system focuses on the support of scale-up processes. Agent Enhanced Workflow utilises software agent technology to solve a typical workflow management problem occurring in a traditional workflow system. It is noted that these systems are devised to deal with the requirements from different domain, and therefore their approaches are different.

3.2.1 InConcert

InConcert from XSOFT is a commercial workflow product implemented using object-oriented technology upon client server architecture. This product, like other traditional workflow products, offers routing of work based on roles, providing users to-do lists and management information regarding process tracking and reporting. In addition, InConcert provides special features for dynamic workflow modification and ad-hoc routing. An application programming interface (API) is provided which includes a full range of application development options for creating custom solutions.

3.2.1.1 Workflow Process

InConcert's Process model (Sarin et al., 1991) is simple, consisting of jobs (corresponding to processes) containing tasks. The process model is represented as abstract object classes with operations. The top level concept job describes a multi-person collaborative activity with some goal. A job consists of a task structure and a shared workspace. The job's tasks are represented through a simple hierarchically decomposed network of tasks and their dependencies. The shared workspace, an important notion in CSCW research (Bannon and Schmidt, 1991), is represented by pointers to documents. A document is an abstract data object that may be manipulated as a unit at some chosen level of granularity determined by the application.

A task in InConcert is a unit of work to be performed by a user through application of the role concept. A role is a placeholder for the task's actor, which can be a person, a user, or a program. The actor concept refers to as user, and each user is associated with a quest of "ready" tasks. Users manipulate tasks through the operations on a queue, such as acquire (get a task), release (back to the queue), and transfer (to another user).

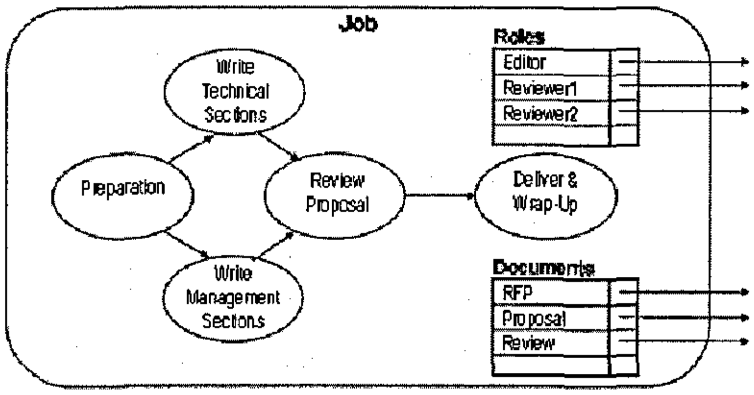


Figure 3-1. Example job structure in InConcert (Sarin et al., 1991).

Jobs are composed of tasks where the task structure (dependencies), the shared workspace (documents) and organisational role references are included. The InConcert API allows for viewing of the whole job as a single data object with task structure, shared workspace and role assignment as constant objects. Tasks may be hierarchically decomposed into subtasks to form a Hierarchical Task Network (HTN). The most interesting is its process model which resembles Input-Process-Output (IPO) models as depicted in Figure 3-1. There is no concept of initial input or final output. All work process results and deliverables are captured indirectly through side effects on the state of running jobs and tasks, and through side effects on updated or created documents that actually are deliverables. Thus, all the "flows" in process models correspond to simple "signals" or triggers that capture synchronisation and dependencies.

The document model is document centric but nevertheless not limited to simple "document flow". It covers the flow of work as jobs composed of tasks and documents that are indirectly referred to, updated and created by theses tasks (Abbott and Sarin, 1994). It is assumed that documents are managed by an external document management system, and InConcert provides facilities for checking documents in and out, and for versioning. The InConcert meta-model is illustrated in Figure 3-2.

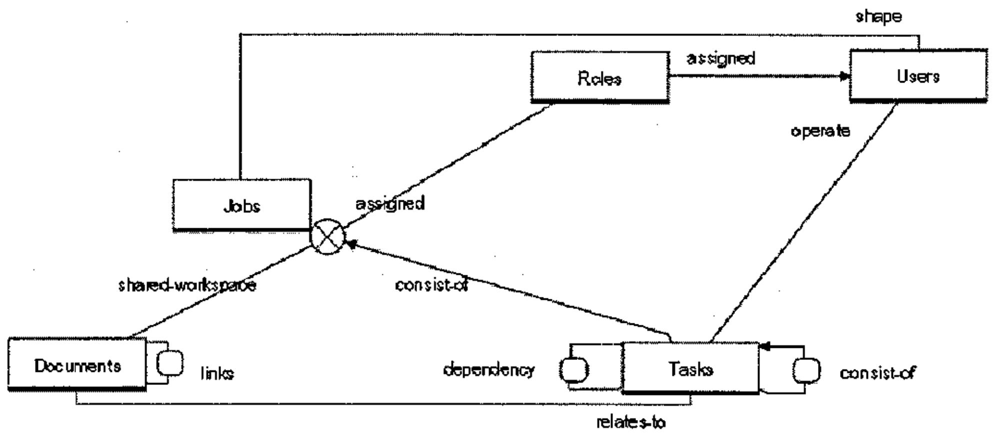


Figure 3-2. InConcert meta-model.

3.2.1.2 Flexibility Features

InConcert supports process change during process enactment where both job structure (task dependencies) and shared workspace (documents) may change at run-time. The process change is realised through graphical editors operating on process models according to a defined API, and this enable a flexible way for user to make changes.

The concept of process fragments (Miers, 1996) is supported through libraries of job templates that may be minimally structured in order to capture ad-hoc processes. The reuse in InConcert is realised based on "copy and paste", it is therefore difficult to harvest best practice from passed experience in finished process instances (Carlsen, 1997).

InConcert's object oriented design together with API and language bindings provide flexibility for system developers deploying workflow solutions regarding integration with other systems, extendibility and customisation. The user interface integration is also flexible because of the use of API to provide tailored desktop integration, or archived by customising particular tasks through a supplied task user interface design tool.

3.2.2 TeamWare Flow

TeamWare Flow is a commercial product that stem from the Regatta research project (Swenson, 1993b; Swenson et al., 1994b), which focused on workgroup support and help in reengineering of work processes. The aim of this research project is to develop a software which supports the coordination of activities among members of work group, providing support for individuals to gain better understanding of the processes, and support change through increased understanding (Swenson, 1993b).

3.2.2.1 Workflow Process

Following requirements were made for the process model (Swenson, 1993b).

- **Ease of process definition.** Every user is able to create and change process models.
- **Ease of monitoring process.** Visualisation of an instance's current state to help actors to understand the progress of process execution.
- **Dynamic modification.** Changes in process definition are allowed at run-time.
- **Partial definitions.** Details of process model may be added at run-time.
- **Individually tailorable.** Policies (process templates) for tasks and goals are maintained in multiple versions for individual groups.
- **Abstraction and decomposition.** The process model may be viewed at various levels of abstraction through hierarchically decomposition.
- **Control of plan.** Each person maintains control of tasks that are assigned to him, including the process model describing the task.
- **Authority.** Process owner may need authorisation to assign tasks to actors.
- **Negotiation.** Assignment of task is subject to negotiation between process owner and actors.
- **Delegation.** Delegating tasks to other persons is possible.
- **Open.** The use of external tools to perform tasks is supported.

The process model provides a shared collaboration space called a colloquy which coordinates the various tasks that make up a process with goals. The colloquy consists of a shared information space and a set of plans that are composed of stages and roles. A stage represents a task request, commitment or question as a specific process step. The stages include user options representing declarations the actor may make to represent the results of the task or a decision. Each option is associated with a particular action and a corresponding message describing it. Plan history is maintained as a set of executed actions. The stages represent not only tasks, but also communications needed for task coordination. A stage is a request from the plan owner to the actor that performs the role associated with stage. Events are associated with each action, and may result in activation or deactivation of other stages. Each stage has a start node for receiving initialisation events and an exit node for sending events to its parent plan. Various stage options represent exclusive alternatives, AND-nodes may be utilised to describe AND-fork. Each sub plan has its owner and may be associated with a particular stage. The meta-model for the concepts in the TeamWare Flow process model is illustrated in Figure 3-3.

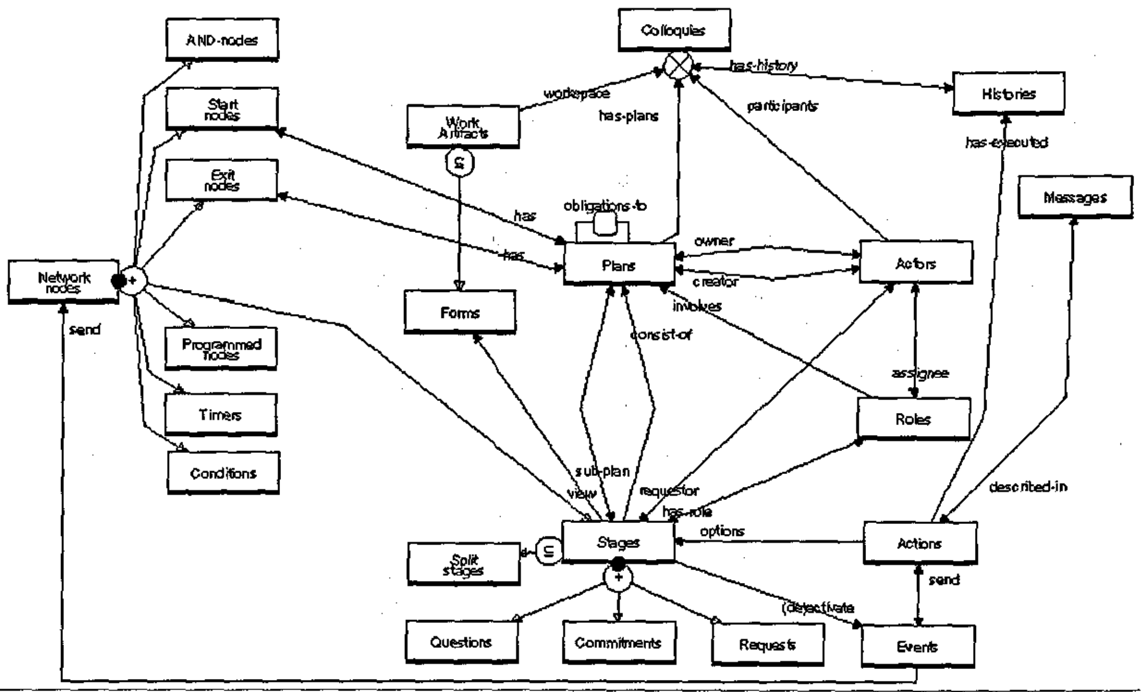


Figure 3-3. TeamWare Flow meta-model (Swenson, 1993a).

An example process model is depicted in Figure 3-4. A Quality Assurance Plan with a start and exit node is shown together with its stages. Stages are represented as ellipses and annotated with the stage role as well as stage options, depicted as small labelled circles. One of the stages is decomposed into a sub plan. The actor can hierarchically decompose the task at his discretion, what he has to do is to ensure the external interface is maintained.

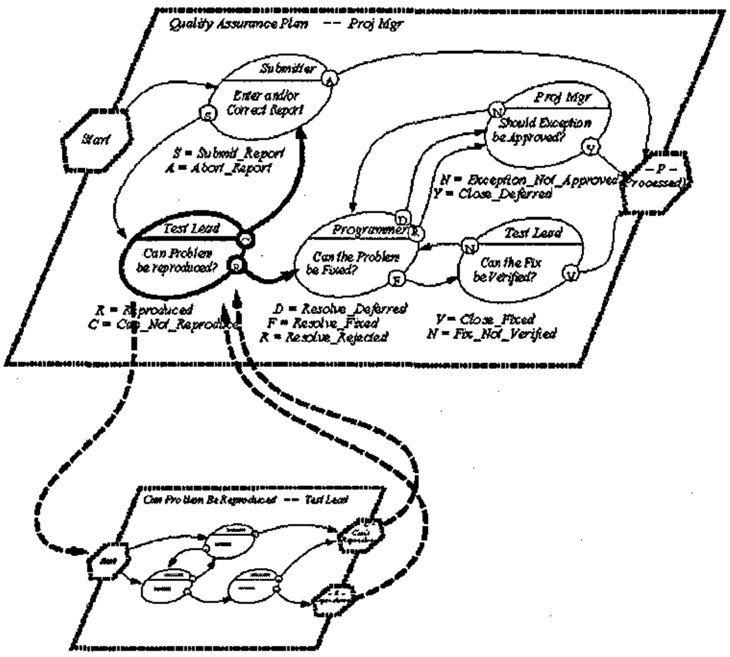


Figure 3-4. A sample process model of TeamWare Flow (Swenson, 1993b).

3.2.2.2 Flexible Features

Teamware Flow is flexible in the sense of providing support for dynamic modification and partial definition of work processes. It is claimed to support what is called collaborative planning (Swenson, 1994; Swenson et al., 1994a) that includes several aspects of flexibility.

- There is no sharp distinction between the work process planner and work process users. Users may construct sub-plans for their process in a visual language.
- There is also no sharp distinction between process build-time and process run-time. Partial definitions may be elaborated, refined and changed by users at run-time.
- Processes are defined by supporting the planning activity in a collaborative way, where process actors may have input to and control over different parts of the plan.
- Individuals or groups can control over their own process definition versions to achieve the same goals, so that they can build and collect their own libraries of suitable organisational actions.

The concept of collaborative planning is supported in the sense of allowing individuals to "plan their own parts". This is an important feature for processes which have to be executed by highly technical staffs, where their superiors raises only the requirements of tasks for which they have to detail, plan and perform themselves.

3.2.3 Task Based Process Management (TBPM)

Task Based Process Management (TBPM) (Stader et al., 2001) system is a research workflow prototype created jointly by Loughborough University and Edinburgh University in collaboration with several industry partners. The overall aim of TBPM project is to investigate the provision of knowledge-based support for workflow management in the area of new product development (NPD) within the chemicals industries. TBPM project takes a system-wide approach to consider how different techniques can be utilised to create a principled and flexible framework, which shall contribute to the speed and effectiveness of the product development process in the chemicals industries. These techniques include dynamic capability matching, the use of the knowledge of organisational structure and authority, and ontology.

An application of the TBPM system is its ability to manage scale-up processes that is part of NPD in the chemicals industries. Scale-up is a long-term process of experimentation and design, requiring a high degree of flexibility, and collaboration between specialists from many business and technical disciplines. The system architecture of the TBPM project consists of three main components (Stader et al., 2001):

- A modelling system that allows that the user to provide information about processes, agents and their organisational context.
- A planning system that uses knowledge-based planning techniques to assist in the planning of processes, using a library of templates for common processes.
- A process enactment system that uses a multi-agent paradigm for coordination of activities and dissemination of information.

The TBPM's knowledge-rich models that are based on a set of ontologies are developed in order to enable these components with the ability to reason about the domain in which they are deployed.

3.2.3.1 Workflow Process

TBPM's approach relies heavily on the use of knowledge about business processes and the context within which they occur (organisational structure, etc). Thus, a set of domain ontologies is used to ensure these knowledge-rich models being unambiguous, carrying the same semantics for the users, as well as the ability of a system to interpret and manipulate models (Peter et al., 1999). There are two distinct sets of ontologies employed in TBPM. The process ontologies defining the concepts central to the management of business process provide support for the task manager in the way of independent of the particular domain. The domain ontologies are a small number of interrelated ontologies that specialise the general concepts of the process ontologies for the particular application domain, for example, the scale up. The architecture of the TBPM system is illustrated in Figure 3-5.

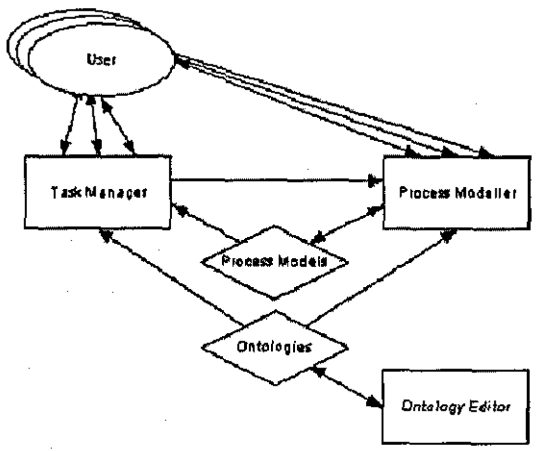


Figure 3-5. The architecture of the TBPM system (Peter et al., 1999)

A task in TBPM's process model (Moore et al., 2000a) is a basic unit of work. Tasks are represented through a simple hierarchically decomposed network of tasks and their dependencies. The dependencies related to the tasks include temporal precedence, pre and post conditions, and resource constraints.

Tasks can be hierarchically decomposed into subtasks if necessary. In this hierarchical task network (HTN), a set of tasks located on the same level to be executed toward a particular objective is called a process plan or method that is stored and managed in a process library. Process plans are indexed in the library by the name of task or high level actions which can be further refined. An objective of a task can be achieved by one or more process plans. Just like a task, each process plan may consist of sub-plans.

When planning a task, user can choose a suitable process plan from the process library and customising it as needed. Each activity in the process plan can also be refined by selecting suitable sub-plans from the library, and so forth. This new construction of activities can be saved as a new process plan, and can be reused in the future. Thus, the process libraries act just like a knowledge base that provides different approaches to achieve certain objectives.

When changes that lead to a task or a sub-task cannot be executed occur, the TBPM's workflow engine can base on this process knowledge to provide a list of suitable process plans in the library to achieve the same objective of the task, allowing user to make a selection for substituting the inevitable one. Figure 3-6 is an example of where two possible alternative plans are shown for achieving the "obtain" task.

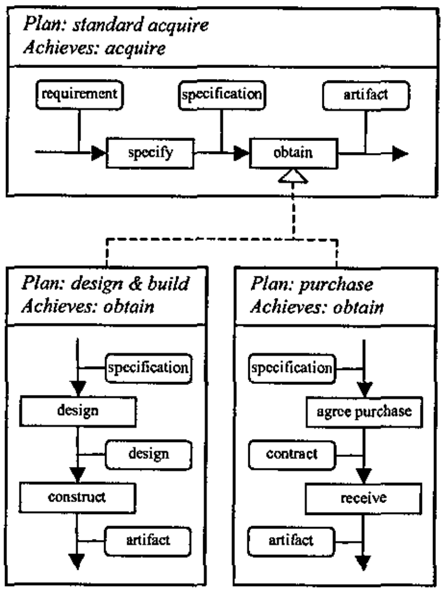


Figure 3-6. Examples of plans from the plan library (Moore, 2000a).

The organisational structure and authority context are modelled in the TBPM system, and agents together with their capabilities are mapped into this model (Peter et al., 1999). The level of satisfaction of an agent's capability for task execution can be achieved by matching the required capability of task execution and the capability that an agent has. Thus, the workflow engine can provide support for selecting the most suitable agent for each task through considering their authority in the organisation in performing the task and their technical capability.

3.2.3.2 Flexibility Features

TBPM allows interleaving between task specification, task planning and task execution, which enables parts of the plans to be specified while the overall process is in progress. Thus, the traditional task cycle in which any individual task must be specified before it is planned and planned before it is executed does not have to be strictly followed. This just-in-time feel in TBPM system provides an extension of flexibility that is particularly suitable for the scale-up processes such as NPD.

The plan library provides support in the form of pre-specified plans. These plans can be assembled and used within the interleaved task execution cycle, which enable the system with the adaptability to suit the specific needs of the current situation. Furthermore, these plans can be seen as the knowledge of providing different solutions to achieve particular objectives, and this knowledge can be enriched as time goes on because of past experiences.

The dynamic capability matching enables the system to deal with the changes occurring at agent level and infrastructure level (Jarvis et al., 2000a). Because of the use of power and finely-grained representation of capabilities, and infusing the system with the knowledge about organisational structure and authority context, the system is able to provide support, through dynamic capability matching, to select a suitable agent who has technical capability with required authorisation to perform the task.

3.2.4 Agent Enhanced Workflow

Agent enhanced workflow (Judge et al., 1998) is a technique that uses intelligent, distributed, autonomous software agents to improve the management of business processes under the control of a traditional workflow management system. A typical workflow management problem, a Correspondence Handling Centre (CHC), is used to demonstrate the feature of agent enhanced workflow. In this demonstrator, the software agents are used to negotiate the distribution of work items and to collaborate in performing real-time exception handling.

The CHC is composed of a Central Administration (CA) and several disparate Work Processing Centres (WPCs) as shown in Figure 3-7.

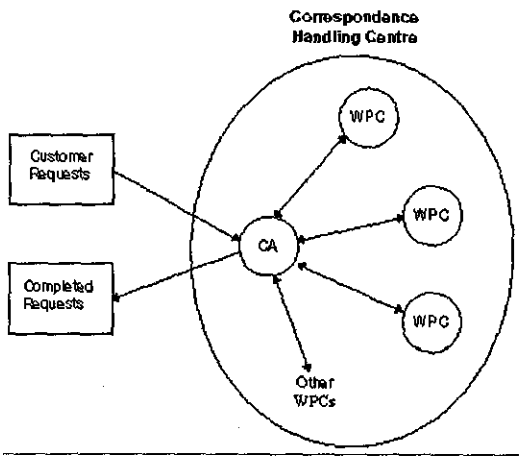


Figure 3-7. Correspondence Handling Centre context (Judge et al., 1998).

CHC may handle all or part of the correspondence received by the enterprise it serves. The correspondence could be many types, ranging from requests to quote for new business to complaints about goods or services, depending on the business in which the enterprise is running.

A simple demonstrative purpose business process in a typical CHC is shown in Figure 3-8. It comprise of six specific categories of activity, namely reception, classification, distribution, processing, inspection, and dispatch. Only processing activities are performed by the WPCs and the rest are the concern of CA.

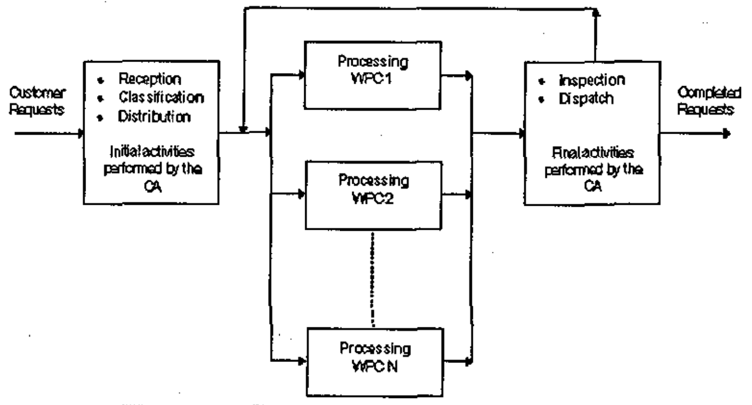


Figure 3-8. A sample business process of CHC (Judge et al., 1998).

3.2.4.1 Workflow Process

There are two types of software agents in the system. CA agent is responsible for managing the activities of the central administration. Work Process agents are responsible for managing the activities of the work-processing centres. Both CA and WPC agents implement the Agent Based Process Management System reference model architecture (O'Brien and Wiegand, 1996) that includes communication, environment and collaboration modules.

The basic construct in this demonstrator system is that the agent layer manages the overall distribution of work by establishing contracts between the CA and individual WPCs. The contracts, which are an extended form of the standard contract net protocol (Davis and Smith, 1983), are used to regulate the flow of work items within the workflow management system. Figure 3-9 shows the high-level system architecture.

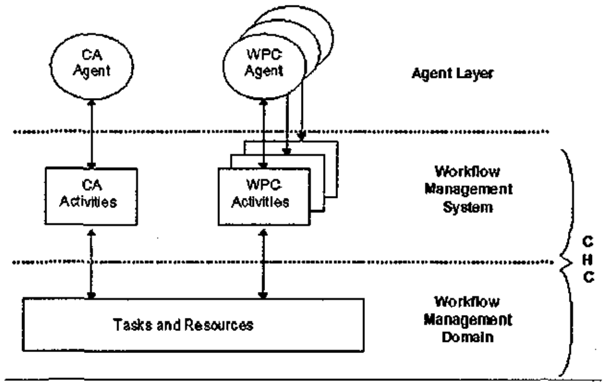
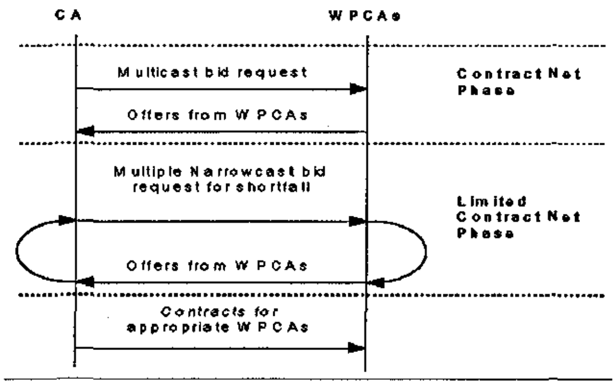


Figure 3-9. High level system architecture (Judge et al, 1998)

In the contract net terms, the CA agent plays the role of a manager, who divides the problem (work distribution) into sub problems (distributing work items), searches for contractors (WPC agents) to carry out the task and monitor the overall solution. In the bidding progress, as illustrated in Figure 3-10, it progressively excludes successful bidders of portions of the total work, which results in fair distribution of work items to a number of WPC agents, rather than a "winner takes all" scenario.



CA
WPCAs
Multicast bid request
Offers from WPCAs
Contract Net Phase
Multiple Narrowcast bid request for shortfall
Limited Contract Net Phase
Offers from WPCAs
Contracts for appropriate WPCAs

Figure 3-10. Software agent negotiation protocol (Judge et al., 1998).

3.2.4.2 Flexibility Features

The use of agent in this agent enhanced workflow demonstrator shows its ability to handle two classes of exception. The modelling of the processes of CHC is difficult to implement in the traditional streamline administration workflow systems; human intervention must be involved in decision making in exception handling.

There are two classes of exception in the CHC process, the external and the internal exceptions. The external exception is the positive or negative change in the amount and/or composition of work received from the outside world. The internal exception is the positive or negative change in the amount and/or composition of work that a WPC can currently process.

CA and WPC agents are autonomous entities. By making a contract that is based on a negotiation process between CA and WPC agents, such exceptions can be handled automatically. Thus, the human intervention is unnecessary.

3.3 Levels of Workflow Adaptation

Han et al. (1998) identified a conceptual framework for workflow adaptation which indicates the support of adaptation should be provided at various levels. The framework has been extended by Jarvis et al. (1999a) and is illustrated in Figure 3-11.

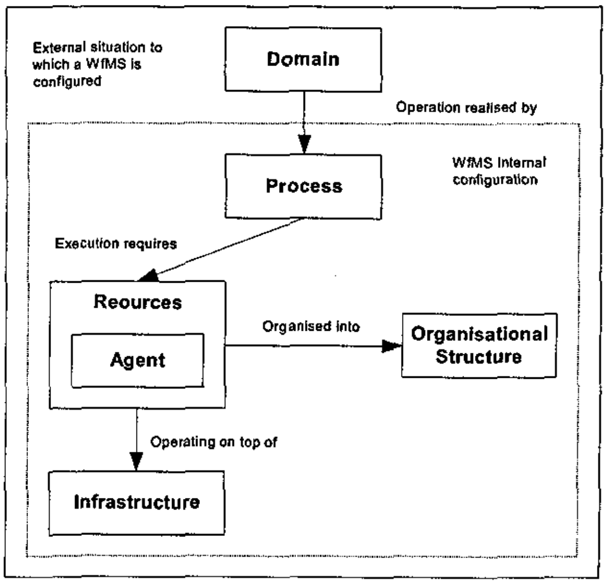


Figure 3-11. Levels of Workflow Adaptation (Jarvis et al., 1999a).

3.3.1 Domain Level Adaptation

Workflow systems usually exist with other business systems that are domain-specific. At the domain level, workflow system can be viewed as a single component that requires adaptation and reconciliation to be integrated into a specific business. The change of business context may have an impact on the application and embedding of workflow systems. Thus, workflow systems should be able to adapt themselves to a range of different business and organisation settings, and also to a changing context. The domain level adaptation may result in a series of adaptation taking place at different levels.

3.3.2 Process Level Adaptation

Process level adaptation deals with the changes to workflow models and their constituent workflow tasks. The changes of workflow process resulting adaptive issues can be grouped into three categories where each poses different challenges for WfMS (Jarvis et al., 2000)

Changes between executions of a process. These changes occur when the requirements of process varies during each execution. For example, the process of designing an artefact may differ from the process which is to be deployed in different countries because of the need for compliance with different regulations. To address changes of this type, workflow engines must support the tailoring of a general process that meets the requirements under particular situation.

Changes during the execution of a process. These changes occur when a process does not always proceed along the predicted path. To address changes of this types, workflow engines must be able to adapt an executing process to changes.

Business process changes. These changes occur when a business consciously changes the way in which it operates. To address changes of these types, workflow engines must support the identification of all instances within a process repository of the process logic that is to be changed. In addition, workflow engines must also support adaptation of processes that are currently executing.

3.3.3 Resource Level Adaptation

Resources include human and non-human resources. To perform process execution, normally at least one agent (person or software system) must be assigned to perform each activity in a process. The availability of a given agent is highly dynamic, such as staff turnover, vacations and work load. In addition, for certain processes, some data-related resources, such as documents, might be required during process execution. Therefore, the changes in resources may have direct impact on workflow process execution. WfMS should adapt to resource changes, assisting on each invocation of a process in the identification of agents that are capable and available to perform its constituent activities.

On the other hand, data and data structure may also change during the execution of a workflow process. Workflow relevant data can be managed by both workflow engines and applications; it can be changed independently and accessed by applications. Thus, the workflow engines should adapt to the data changes and react to these changes appropriately.

3.3.4 Organisational Structure Level Adaptation

Agents are arranged into an organisational structure. The changes in organisational structure not only have impact on selecting agent, but also on retrieving critical data for process execution. For example, if the data holders are moved to other departments, this may lead to a difficulty in assigning an updating request to them. In addition, in the context of an organisational structure, an agent may be technically proficient in performing an activity but not organisationally empowered to do so. Thus, workflow engines should account for these organisational norms when identifying appropriate agents for process execution.

3.3.5 Infrastructure Level Adaptation

Within a dynamic business world and fast growing computing industry, the supporting software systems may need to adapt quickly to a modified business environment or a technical setting, resulting in new system configurations. Thus, workflow engines must support flexible system infrastructures by being able to communicate with distributed and heterogeneous software systems, and therefore keep up with the changes.

3.4 Technologies for Building Adaptive Workflow

An adaptive workflow engine must be able to provide process management capabilities, allocation of resources in response to process requirements, and adaptation of process and allocations in response to changes in assigned tasks and dynamic operating environments. The work on reactive control, scheduling, planning, Software Agent, Dynamic Capability Matching and Ontology provide the underlying technologies meeting the requirements for adaptive workflow engines.

3.4.1 Reactive Control

Reactive control systems are a form of knowledge-based software controller that operates as an embedded system within highly dynamic environments. A reactive control system is typically organised around an interpreter that runs with an uninterrupted detection of key changes in the operating environment or sets of tasks, deliberating to determine how to respond to sensed changes, and acting to execute relevant responses.

Reactive control systems with procedural approaches in which predefined procedure libraries describe process that can be executed to achieve some goals, or that serve in responses to designed events (Musliner, 1993) (Pohl et al., 1999) (Tabbara et al., 2000), are particularly well-suited for the activity-based paradigm for workflow (Berry and Drabble, 1999). The bodies of these procedures employ rich operations and control constructs that provide a highly expressive framework for representing events and activities.

In activity-based paradigms, a process is decomposed into individual modules that provide small, coherent units of functionality. Each of the units generally consists of a description of the purpose of that unit together with conditions of applicability. These hierarchical representations enable the encoding of complex activities at multiple levels of abstraction. High-level activities can be initiated without low-level details, and low-level decisions are made on an as-needed basis when the time comes to execute those actions. For example, in TBPM (Stader, 2001) the high-level tasks can be initiated firstly to deal with the requirements of scale-up processes execution, and the details of low-level tasks can be defined when the critical figures are specified.

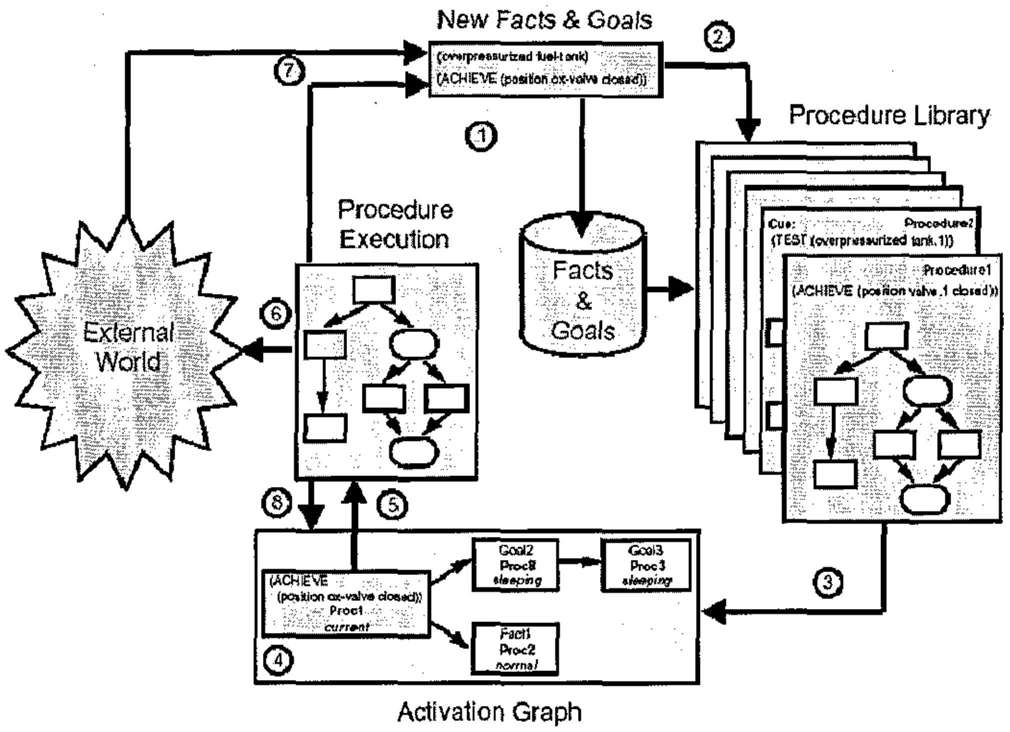


Figure 3-12. Operation of a Procedure-based Reactive Controller (Myers and Berry, 1999)

Figure 3-12 depicts a single execution cycle of a procedural reactive control system (Myers and Berry, 1999). (1) At a particular time, some goals are established and some events occur that change the beliefs held in an internal model of world. (2) The changes in beliefs trigger various procedures in response, (3) one or more of which will then be chosen for activation. (4) The interpreter then selects a task from the set of activated procedures and (5) executes portion of that task. (6) The result will be either the execution of a primitive action in the world, (7) or the establishment of a new sub-goal, (8) or a modification to the activated procedures. At this point, the execution cycle begins again.

Monitors in reactive control systems are used to check the status of activated processes, determining when to adapt or abandon activities, and to respond to key changes in the operating environment. The monitor is invoked by a set of pre-defined triggering conditions. To respond to such conditions, the monitor will invocate the pre-defined processes, adapt them to current activities, or abandon current activities that will be substituted by a new process that is suited to current environment.

Reactive control systems operate as embedded systems in dynamic environments in which the processes are executed in irrevocable ways. Thus, the recovery techniques grounded in checkpoint paradigms and rollback paradigms that rely on storing coherent state periodically to enable rollback to consistent state in case of unrecoverable failure, are impracticable. Instead, forward recovery methods are adopted for process recovery and repair that will perform further appropriate actions to transit a failed state into some safe state. For example, TBPM's "plan patch" approach (Moore et al., 2000a) that is used to identify the difference between the new and old methods, and to support user in moving them while the business processes are changed during processes execution.

More effort has been made to failure presentation within reactive control systems. Forward search methods have been defined, so-called safety rules, to determine the actions to take from various states in order to avoid failure states (Godefroid and Kabanza, 1991).

3.4.2 AI Scheduling

Workflow systems require efficient scheduling for enabling process execution during dynamic and uncertain environment. However, though traditional scheduling algorithms are powerful, they often failed to address the complexities of the domain or the dynamics of the environment. AI scheduling techniques provide a strong foundation for building such capabilities, particularly in the area of reactive scheduling, uncertainty management, and the propagation of capability and availability constraints. There are two main AI scheduling approaches, namely generative and stochastic.

Generative scheduling. This approach starts with an empty schedule and iteratively selects and assigns resources to each activity over time, and backtracks whenever a problem is found. The generative techniques have many advantages: the search strategy is complete and straightforward, solution quality can be controlled dynamically, and load-balancing techniques such as the use of capacity analysis can be easily incorporated. However, they provide little direct support for recovery while the schedules are disrupted, therefore unsuitable for the integration with reaction control systems.

Stochastic scheduling. This approach is based on the principle of iterative improvement. The idea starts with a completed scheduled containing unsatisfied (i.e. broken) constraints and iteratively selects and reassigns resources to an activity over time that satisfy as many unsatisfied constraints as possible. Although there have more techniques compare with generative scheduling of addressing schedule repair problem, they require additional machinery to accommodate schedule failures.

Both generative and stochastic methods have failed when applied to real time dynamic execution environments. The techniques in reactive scheduling that emphasises on building robust schedules, in which a valid schedule can be maintained while new activities are added, are helpful in addressing these problems. The ability to reschedule during execution is still an ongoing research topic. There are mainly two methods for addressing these problems, namely constraint-directed methods and constrained iterative repair methods.

Constraint-directed methods. These approaches use constraint analysis to prioritise outstanding problems in the current schedule, identify critical modification goals, and estimate the possibilities for modification with disrupting schedule. Possible modification actions include reordering the sequences of activities, re-sequencing the groups of activities on one resource, substituting resources, temporal shift activities, and pair wise exchange of resource or temporal assignments (Sycara and Miyashita, 1992).

Constrained iterative repair methods. These approaches are developed based on anytime algorithms in which a legal but low quality solution is generated quickly, which are then continuously updated to higher-quality solutions (Zweben et al., 1994). Stochastic techniques can be used to iteratively improve a schedule by making small changes, but must be constrained to repair only the disrupted part of the schedule in order to maintain stability.

3.4.3 AI Planning

Adaptive workflow systems have been involved in dealing with dynamic process execution in a rich domain where a priori process libraries that provide full spectrum of situations and conditions are difficult to be defined. When it is impractical for human to supply new process definitions, a process must either be created through synthesis form domain descriptions, or obtained by adapting previous defined processes to meet the new requirements, called plan repairing.

Automated planning techniques can be used to synthesise new processes from previously defined process templates that are guaranteed correct relative to current commitments and knowledge. Various planning systems have been explored; two of them provide synthesis of new processes from libraries of previous defined processes, namely hierarchical task network planning and case-based planning.

Hierarchical task network (HTN) planning. These systems synthesise plans using process libraries. The objective is usually specified as a high-level task to be performed. Planning proceeds by recursively expanding high-level tasks into networks of lower-level tasks, eventually bottoming out into a set of directly executable tasks. The expansions of high-level tasks into sets of lower-level tasks are described by transformation rules called methods. A method is a mapping from a task into partially ordered networks together with a set of constraints. Methods are stored and managed by process libraries in workflow systems. Through this decomposition architecture, the overall set of processes is feasible with respect to both stated applicability conditions or process definitions and any resource requirements.

HTN planning can be used to compose long-range processes from smaller units with the help of look-ahead analysis that ensures the viability of the constructed processes. The domains which successfully adopt HTN planning to synthesis processes attest to its values, include new product development within chemical industries (Stader et al., 2001), manufacturing process plans (Hebbar et al., 1996), commercial bridge playing (Smith et al., 1996), air campaign planning (Lee and David, 1996), crisis action planning (Wilkins and Desimone, 1994), oil-spill planning (Agosta and Wilkins, 1996), and antenna operations (Chien et al., 1997).

Case-based planning. These systems (Hammond, 1989) (Veloso, 1992) generate new plans for a given situation and task by retrieving solutions for similar problems from a previous defined case library, then adapting them to meet the new requirements. Case-based planning methods provide a way to build on experience with previous defined processes, usually with the help of knowledge based systems (KBS), providing adaptation to meet new situations.

HTN planning and case-based planning can be used to create processes when rich libraries of process are already defined. For the situation where current processes are inadequate, the machine learning methods may be used to generate new processes. For example, machine inductive approaches have been used to refine applicability conditions for process templates based on experiences (British Aerospace, 1996). Statistical pattern recognition approaches have been used to identify possible causes of failures by analysing execution traces, producing suggestions to improve the robustness of processes (Howe, 1995). Even though machine learning is an important area and the research on it has a long history, using learning methods for creating and improving process description is still in its preliminary stage.

Plan repair methods provide the basis for recovery from problems during process execution. Most plan repair methods start with analysis of the dependence structures that enable a plan to determine problems relevant to the current state and execution results (Kambhampati and Hendler, 1992). There are two main sources of such problems, precondition failure and execution failure. Precondition failure occurs when defined preconditions of a process cannot be satisfied at the time the process is to be executed. Execution failure occurs when a process execution does not obtain its expected result.

Several plan repair methods have been developed under different methodologies for different plan recoveries. The plan repair in generative planning systems focuses on replanning. New sub-plans are generated to substitute for problematic portions of the original plan that are identified by dependency analysis (Manuela, 1998) (Kambhampati and Hendler, 1992). The replanning methods in generative emphasise correctness preserving and minimal disturbance. Correctness preserving includes ensuring the modified plan that is relevant to the current state of knowledge, and minimal disturbance refers to the modification only performed on the portion that must be changed to ensure correctness. The redirecting processes execution potentially involves high cost, therefore keeping minimal changes is critical to ensure the continuity of the plan. In case-based approaches, the problematic portions of the plan are replaced by applying the libraries of predefined domain-independent adaptation methods. Thus, case-based approaches are generally correctness preserving but do not guarantee minimal disturbance.

The ideal plan repair methods should provide effective repair in problematic plans but costing minimum disturbance. On the other hand, the robust process plans that are less likely to fail in the first place, are more effective in maintaining smooth workflow executions. The work in this area is still in its preliminary stage but will become increasingly important as the need of adaptive workflows that are executed in dynamic, real-word environments. For example, the plans in contingency planning are conditioned with several sets of possible alternative situations, such as providing different sub-plans for good and bad weather. However, these methods are only suitable for the situations where the uncertainty is limited and well circumscribed. For rich domains where the uncertainties are ubiquity, more sophisticated methods are needed.

3.4.4 Software Agents

Software agents provide two facilities relevant to adaptive workflow. Firstly, they offer infrastructure that enable distributed and heterogeneous systems to communicate, based on heterogeneous agent architectures. This facility is directly relevant to infrastructure level adaptation. Secondly, they are compatible with the framework for dynamic capability matching. These facilities are directly relevant to agent level adaptation and will be discussed in the next section. This section focuses on infrastructure facilities.

To enable communication between distributed and heterogeneous software systems, these software systems need to be converted into software agents. There are three ways of doing the conversion (Nwana, 1996). First of all, is to rewrite the legacy systems to meet the criteria for agenthood, but it is the most costly approach. The second approach is to use a transducer that is a separate piece of software that receives messages from other agents and translates them into the legacy software's native communication protocol, and passes the message to the program. Similarly, it also translates the program's responses into Agent Communication Language (ACL) that is sent on to other agents. This approach is suitable for the situation where the code may be too delicate to tamper with or is unavailable. The final approach is the use of wrapper technique. In this approach, a software "wrapper" is developed and attached to software systems to allow it to communicate in ACL, as show in Figure 3-13. The wrapper translates outgoing messages of software systems into the ACL and incoming messages from the ACL into their native communication protocols. This approach brings about more intervention that requires the code to be available, but offers greater efficiency than the transduction approach.

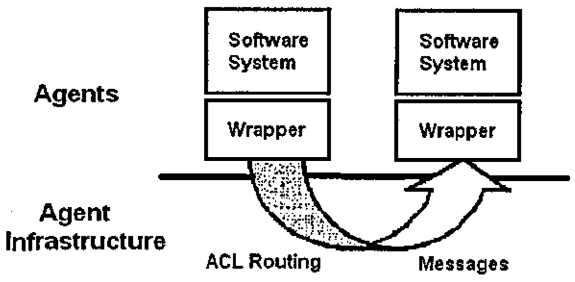


Figure 3-13. Agent Infrastructure Schematic (Jarvis, 2000).

When the agents are available, there are two possible architectures. In the first one, all the agents handle their own coordination. In another one, groups of agents can rely on special system programs to achieve coordination. The disadvantage of the first one is that the communication overhead does not ensure scalability that is a necessary requirement for the future of agents. As a result, the latter federated approach is preferred. Figure 3-14 shows an example of a federated system.

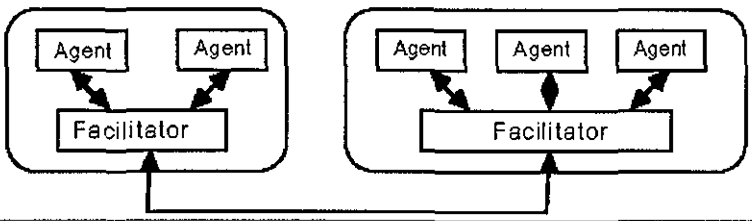


Figure 3-14. A Federated System (Genesereth and Ketchpel, 1994)

In Figure 3-14, there are five agents distributed into machines, one has two agents and the other has three. The agents do not communicate directly with one another but do so through intermediaries called facilitators. The facilitators are able to locate agents on the network to provide various services. They also establish the connection across the environments and ensure correct conversation among agents.

These agent based architectures and technologies endow with the workflow systems not only the capability to communicate with the distributed and heterogeneous software systems, but also the deployment in the distributed and heterogeneous environments. This agent based architecture enables workflow systems that are capable of dealing with the situation where the underlying infrastructures are frequently changed as a result of exploiting technological advances.

3.4.5 Dynamic Capability Matching

The sensitiveness of workflow systems to the organisational structure and authority context within which it operates has increasing importance (Joosten, 1996) (Tate, 1993) (Kappel, 1995) (Dellen, 1997) (Rupietta, 1997). Within the workflow, every task has to be assigned to one or more agents (humans or systems) for execution. Agents, particularly the human agents, are arranged into an organisational structure in which the authorities of agents are explicitly represented. Thus, the changes in organisational structure and authority may affect the workflow execution. Automatic selection of a suitable agent to instead of unavailable one may not be able to cope with all cases. In this situation, the workflow engine can assist planner with a prioritised list of agents based on their capabilities and authorities. The planner can then pick the most suitable agent to takeover the tasks or resolve the situation by alternatives like change the plan.

Endowing workflow systems with the knowledge of capability of agent, organisational structure and authority is an effective way in addressing the problems from the changes in agents and organisational structure while processes execution (Jarvis et al., 1999a; 2000) (Moore et al., 2000b). Using this knowledge with capability matching functions, workflow engines can identify suitable agents for performing the task by matching the capabilities required by the task against the capabilities held by available agents, within the organisational norms. For example, an engineer has been identified as the only suitable agent to perform a design task by capability matching. However, because he is not the member of the team, the workflow systems should notify the planner that the permission from the superior before assigning the task to the engineer is required.

A capability specification approach is proposed by Jarvis et al. (1999a). The terms used to describe a capability are drawn from a capability ontology that provides a hierarchy of capabilities. For example, database provides a Store capability and more specifically, a relational database provides a Store Relational capability. A capability specification is composed of two parts: the technical capability itself and the area in which it can be applied. For example, if a specific database application can store data about skills, it can apply its Store capability to Skills. Each of the parts uses its own hierarchy of terms.

Finding a perfect matching, however, is highly unlikely. The situations like that if an agent knows C++ and Java programming, how well he fits the requirement where the knowledge of LISP is required, are not tackled. Therefore, a more intelligent capability matching method is required.

3.4.6 Ontology

An ontology is a data model that "*consist of a representational vocabulary with precise definitions of the meanings of the terms of this vocabulary plus a set of formal axioms that constrain interpretation and well-formed use of these terms.*" (Campbell & Shapiron, 1995) An ontology is therefore an explicit representation of a "*...shared understanding of some domain of interest...*" (Uschold & Gruninger, 1996). It can be started and re-used by others in the same domain to minimise ambiguity. Key features of the ontology include:

- An ontology of some domain of interest identifies and precisely describes the important concepts and their relationships in the domain.
- The terms and their definitions are agreed between all participants within the domain, and form a basis for communication about the domain.
- An ontology can be specified independently from the intended particular application. This enables its reuse for other purposes and applications touching the same domain. Thus, different applications can use this ontology to communicate with each other in the matter of the same domain.
- An ontology can be formalised and thus support communication between computer systems.

Ontology is not a technique that is directly relevant to adaptive workflow, whereas it provides support for inter-operability where all the adaptation enabling techniques described can communicate with each other, so as to form an integration solution to deal with the requirements of different levels of adaptation as a whole.

Ontology is sets of vocabularies or terms used to refer to the shared understanding of some domain of interest, which may be used as a unifying framework to solve the communication problems among stakeholders (Uschold and Gruninger, 1996). To develop a flexible workflow system, it is necessary to employ ontology that enables not only the communication among constituent systems, but also covers the people and organisations. In TBPM project (Peter et al., 1999), a set of ontologies that includes tasks, plans, resources, organisations and application domains, is employed for providing support for communication among participants.

Normally, multiple tool sets, including workflow systems, are used in an organisation. However, due to the different needs and background contexts, there can be widely varying viewpoints and assumptions regarding what is essentially the same subject matter. For example, each of off-the-shelf software uses different jargon which may have differing, overlapping and/or mismatched concepts, structures and methods. The consequent lack of a shared understanding leads to poor communication within and between these systems. In this situation, the ontology can be used to support the translation among the different constituent systems that use different languages and representations. Figure 3-15 illustrates the use of ontology as an inter-lingua to integrate different software tools. The term "procedure" that is used by one tool is translated into the term "method" that is used by another tool through the ontology, whose terms for the same underlying concept is "process".

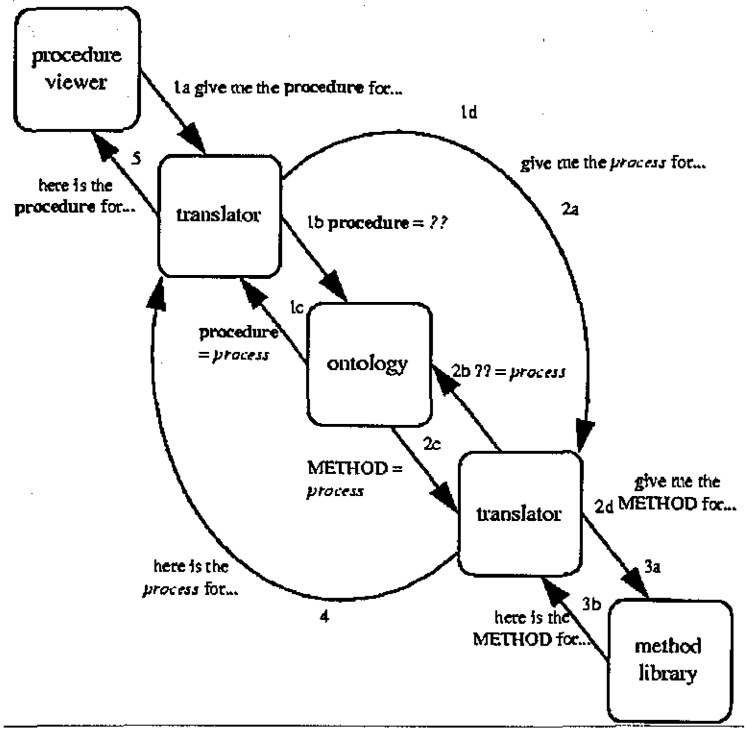

procedure viewer
1a give me the procedure for...
1d
give me the *process* for...
5
here is the procedure for...
translator
1b procedure = ??
2a
1c
procedure = *process*
ontology
2b ?? = *process*
2c
METHOD = *process*
translator
give me the METHOD for...
2d
3a
here is the *process* for...
4
3b
here is the METHOD for...
method library

Figure 3-15. Ontology as Inter-Lingua: an example (Uschold and Gruninger, 1996).

3.5 Chapter Summary

This chapter gives an introduction to the workflow adaptation and several enabling technologies from the AI community. A survey of two classic commercial products and two research prototypes which demonstrate various flexible features of workflow is presented. While commercial products focus on providing advanced process models, current research prototypes emphasise on using innovative technologies, particularly in AI aspect, to overcome some limitations of current WfMS.

Han's conceptual framework for workflow adaptation is employed for understanding the different levels of changes that may have impact on workflow execution. The framework categorises the adaptations of workflow systems to changes into five levels, and the requirements of handling workflow adaptation for each are identified.

This chapter than introduces various AI technologies for enabling adaptive workflow. The application of each at different levels of adaptation is considered. Planning, scheduling and reactive control are useful in solving the changes at process level. Dynamic capability matching helps in identifying agents in a dynamic environment. Software agent technology provides support for adapting infrastructure changes. Ontologies work as a common language that enabled communication among modules.

Part 2

Supporting Engineering Process with a Novel Framework

Chapter 4

Understanding Engineering Process

"If you try to catch the tail of your leading competitors,
you will always remain behind them.
Instead, you have to focus on the direction in which their head is turning.
Only then can you ever think of overtaking them."
- Edward Biernat, Bausch and Lomb

4.1 Introduction

Engineering process refers to the governing of technical management process that manages and control product development. It addresses all aspects of total system performance and provides technical interface with other key processes. It defines the technical processes and interfaces and provides the technical baseline for the integrated master plan for development and production. The primary function of an engineering process is to ensure that the product meets the cost, schedule, and performance needs encompassing the product development life cycle.

Engineering process differs from general business processes. It is highly technical, dynamic, ad-hoc, collaborative and involves vast amount of information interchange of which current WfMS lack the ability to support.

This chapter introduces the concept of an engineering process by means of a product development process. The requirements necessary of a successful supporting system are identified. It is organised as follows: §4.2 gives a general introduction to a new product development process; §4.3 discusses the overlapping of development activities, particularly in the engineering aspects; §4.4 introduces the role of industry standards in a product development process; §4.5 identifies and discusses the characteristics of tasks in an engineering process; §4.6 outlines the challenges that have to be addressed by a system which is to succeed in supporting engineering processes; a summary of this chapter is given in §4.7.

4.2 Introduction to Product Development Process

The idea of a new product normally comes from either an external or internal desired need. External causes for a new product include, for example, the direct order from a customer, the obsolescence of an existing product, or a change in market demands. On the other hand, it may come from new discoveries and development of a company or the need for a product identified by the marketing department.

Once the need has been established, the product has to be designed and manufactured. To understand the product development process, it is necessary to review the product life cycle that is shown in Figure 4-1. The life of a product begins with its planning and ends with its disposal. The product development process refers to the steps in the product life cycle, from its go-ahead to when the product physically exists. In general, the product development process consists of four steps, namely product planning, design, manufacturing planning and manufacturing.

4.2.1 Product Planning

Product planning is the process of search, selection and development of ideas for new products. A systematic product planning process will lead to a more successful project in meeting the constraints of cost and time. A product planning process include following tasks:

- Establishing product goals.

- Conducting market analyses.
- Detailing the benefits the product will provide the customer.
- Deciding on the features the product will have.
- Establishing product performance.
- Conducting an economic analysis and setting the cost target.
- Establishing the expected sales volume.
- Setting deadlines for completion of tasks and setting up the manufacturing line.

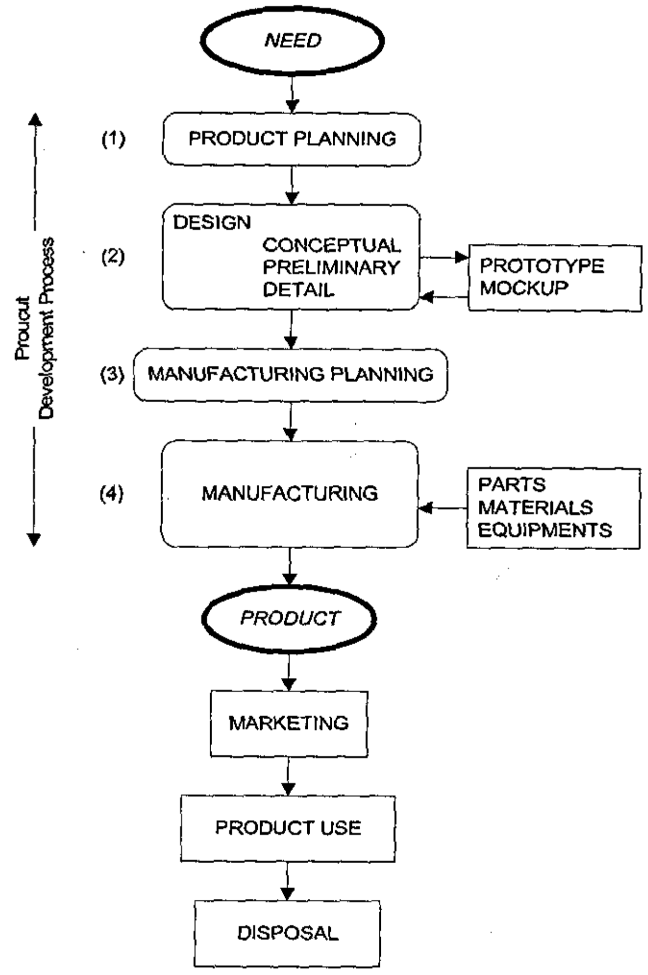

NEED
(1) PRODUCT PLANNING
(2) DESIGN CONCEPTUAL PRELIMINARY DETAIL
PROTOTYPE MOCKUP
(3) MANUFACTURING PLANNING
(4) MANUFACTURING
PARTS MATERIALS EQUIPMENTS
Product Development Process
PRODUCT
MARKETING
PRODUCT USE
DISPOSAL

Figure 4-1. Life phases of a product.

4.2.2 Design

Design is a process of devising the product to meet desired needs. It is a decision-making process, in which a number of technologies are applied to convert resource optimally to meet a stated objective. The design process starts from the preparation of the requirements or specification list. The list includes the overall function of the product and any sub-functions foreseen by the designer. The requirements are classified according to (a) life phases of the product (refer to Figure 4-1) and (b) types of requirements (e.g., technical, economic, ergonomic, and legal) in which the technical requirements are the most important.

The design process consists of three sub-processes: conceptual design, preliminary design and detail design. They are normally performed sequentially, proceeding from a more abstract level and takes on a more concrete form at the end (Pahl and Beitz, 1996).

Conceptual design is a very important process which looks for different concepts that can be used to achieve the requirements. Each concept is an outline solution to a design problem, identifying the spatial and structural relationship of the principle components in achieving each major function of the product. Enough details have been worked out so that the cost, weight, and overall dimensions can be estimated.

The starting point of the concept development process is the determination of the functions that must be fulfilled. Each function can be realised by one or more methods, and therefore a functional analysis of the requirements is needed, in which a complex function may breakdown into simpler sub-functions. In the next step, the physical effects on determining how the functions are realised are considered. By considering different solutions for the functions, a number of different concepts can be generated, in which the one or more that best satisfies the specification are chosen.

The second step of the design process is preliminary design which is more concrete and technical in nature. Engineers begin to select and size the major sub-systems, based on lower-level concerns that take into account the performance specifications and requirements. The final choice from among the proposed concepts is made in this phase, including part shapes and size, fastening methods and materials.

The final step of the design process is detail design where engineers turn to refining the choices made in the preliminary design phase, articulating early choices in much greater detail, down to specific part types and dimensions. The detail design process typically consists of standard procedures and techniques. Relevant knowledge is found in design codes, handbooks, databases and catalogs, and is normally performed by component specialists because the design is now close to being assembled from a library of standard pieces. The final decisions on dimensions, arrangement, shapes of components, and material are made.

4.2.3 Manufacturing Planning

The process of manufacturing planning involves decisions on how the product is to be manufactured. A plan of the steps required to manufacture the product, the manufacturing processes, machines, tools required, how the parts are to be assembled and so forth is developed. The manufacturing planning process includes productivity analysis, initial process design, vendors or sourcing selection, and tools design.

4.2.4 Manufacturing

The product is produced in manufacturing process which includes materials handling, production of parts, assembly, quality control, and related activities. Many of the decisions regarding manufacturing have already been made during the design stage, knowingly and unknowingly. Some of them are inappropriate and therefore have to be changed in this stage. The following decisions made in the design process have the largest influence on manufacturing process (Ehrlenspiel, 1985):

- The nature and number of sub-assemblies and components defined in overall design arrangement.
- The methods used and their quality in the design of components.
- The material selected for components.
- The number and type of standards to which the product has to conform.

4.3 Concurrent Engineering

In a traditional product development process, the activities of product planning, design, manufacturing planning and manufacturing are carried out in a relatively independent and sequential way, as shown in Figure 4-2 (a). A process cannot start until its ancestor process has been completed and signed off. In this approach, product design and manufacturing processes involved in development and production of new products are compartmentalised with different specialists. The collaboration between the different functional areas is limited to a series of standard engineering change orders, and information is transferred in batches at the end of stages, via documents and computer networks (Clark and Fujimoto, 1991). New product development, however, suffers from this sequential engineering.

Development time is relatively long because of the sequence of different activities and the problems that occur at the interface between two different stages of product realisation process (Azzone and Bertele, 1994).

The poor product quality and the higher life-cycle costs is a consequence of the sluggishness of dealing with last-minutes changes that must be made at manufacturing stage (Hundal, 1998).

This single direction batch information transfer does not encourage marketing people to take into account engineering considerations (Wind and Robertson, 1983) or product design engineers to take into account manufacturability considerations (Clark and Fujimoto, 1991), leading to problems of marketability and manufacturability.

This single direction batch communication often leads to subtle inconsistencies in the information transmitted downstream, creating further problems for downstream functions (Clark and Fujimoto, 1991).

Smith and Reinertsen (1991) identify the key ingredients to keep a manufacturing company alive are (1) product innovation and (2) rapid development and marketing of new products. As manufacturing has become more globally competitive in recent years, some techniques of the time-driven development (Hundal, 1998) have been used to shorten the product development cycle. Time driven development can also be called rapid product development, accelerated product development or integrated product development. Use of these techniques allows a certain level of coordination and overlap of product development process, such as the concurrency of product and process design, as shown in Figure 4-2 (b).

Figure 4-2 (b) illustrates the time-driven product development process. The development activities are overlapped in order to reduce the development time and cost and to increase quality (Hayes et al, 1988). The overlapping of these activities implies that a process might be started on the basis of incomplete information. To deal with that, a continual sharing of information take place between teams engaged in the activities and their successors. The downstream team always makes quick changes in its processes in response to revised information from the upstream team. This two way flow of information also leads to fewer misunderstandings and errors (Dean and Susman, 1989), and early resolution of conflicts (Hundal, 1998).

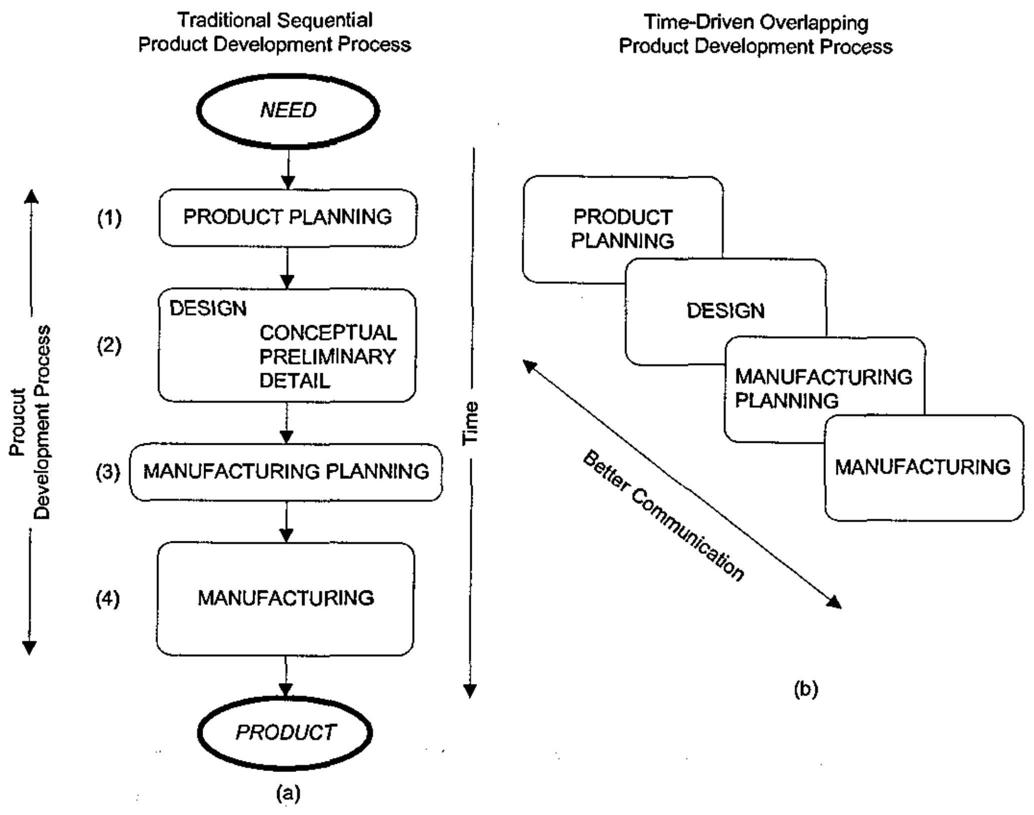


Figure 4-2. Product Development Processes.

The benefits obtained from concurrent product and process design can be extended by overlapping of the activities in the development cycle. Concurrent engineering may also include interplay with marketing, sales, field service and even suppliers. Figure 4-3 illustrates a high degree of interplay among all three key activities in product development cycle. Usually, sufficient information is available at appropriate stages for each activity to enable some operations to begin in the next stage. For a larger project, this interplay can be decomposed into smaller units if necessary.

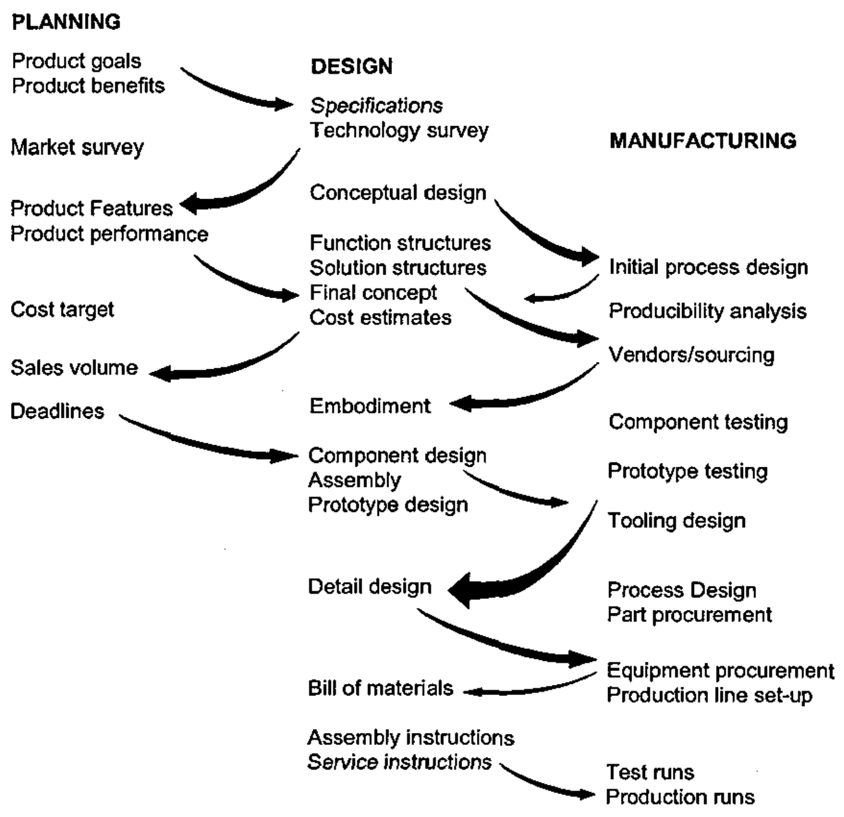


Figure 4-3. Overlapping of activities in planning, design, and manufacture (Hundal, 1998).

The sequence of overlapped activities in Figure 4-3 is as follows:

1. After the planning has identified the market position, price range etc. of the product, the design team can begin preparing the requirements list and performing the technology survey.
2. The concept can be developed based on the identified product features, performance and price range.
3. Perform sales forecasting and set deadlines for subsequent activities concurrently with, or following, the concept development.
4. Manufacturing begins to look at the production process design, analyses producibility, and begins selecting vendors. This is performed in parallel with conceptual design after the product structure is finalised.
5. At this time, the embodiment has started, and specific components are shaped and assembled and then to the initial prototype design.
6. Build and test prototypes to validate the component production and procurement.

7. The details of the design are finalised after the final tooling and process design are implemented.
8. The final activities are the procurement of materials or special production equipment and setup of the production line.
9. Design prepares the final documents on bills of material and assembly and service instructions, while manufacturing setup up the test run. This is followed by actual production.

4.4 Quality Control through Industry Standards

> *"Standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose."* (ISO, 1997)

In order to achieve the required quality for a product, a number of industry standards or regulations have to be used. The standards that have to be used are subject to the country where the product will be produced or consumed, and the function of the product. Well established engineering standards, such as IEC61508 (IEC, 1997), set down the properties that both the development process and its product have to possess. These properties are documented at a particular point or at the end of the development process. Once a standard has been adopted by an engineering project, it is important to manage compliance with that standard.

Standards constrain the product development process, resulting in a number of activities that must be included in a development process and have to be performed in the suggested sequence by qualified staff using appropriate techniques and methods. These standards are generic; their every application, however, is different because of the differences in project details. Thus, no canonical process structure can be applied to all engineering projects where the processes and products have to be compliant with a particular standard. The current best practice of compliance checking is to compare the development documents with the standard at the end of every development stage. If incompliance is found, the relevant development processes may need to be rewound to a suitable state and re-processed again.

4.5 Characteristics

Paashuis (1998) summarised the characteristics of product development process and its organisation. The summary is drawn upon the work of Burns and Stalker (1961), Thompson (1967), Galbraith (1973), Van de Ven et al (1976) and Pelled and Adler (1994). They identify some fundamental factors which in a sense explain the need for coordination and overlapping of development activities. The summary is extended based on our observation on product development processes in general.

4.5.1 Task Scale-Up

Scale-up typically occurs at a point during a product development process when a promising product has been identified, preliminary marketing investigation has been done, and a potential process for design and manufacturing of the product has been proposed, but not fully investigated. The intention of scale-up is to investigate the behaviour of the proposed process. During the scale-up process, a series of experiments are performed at a gradually increasing scale, starting in the laboratory and ending (if all proves satisfactory) with a working pilot plant. As the scale of the development task increases, an increasing number of different disciplines and departments will get involved, and interaction and ad-hoc processes will also increase.

Many organisations adopt process models to organise and control various activities involved in the development of a new product. A canonical process model provides a skeleton by which each project manager can build his/her own program evaluation and review technique (PERT) or network diagram specific to any one project. These process models consist of a series of "go/kill" decisions spanning the whole product development life cycles, particularly in the early stages (Cooper, 1983). This situation causes the design and manufacturing plans to be devised only at a higher level, the details of which will then emerge as some tasks have been done. Therefore, the planning in a scale-up project is relatively dynamic. The results from former tasks are used in planning the details of sub-sequential activities. The process plan grows along with its tasks execution.

4.5.2 Task Uncertainty

Task uncertainty refers to the difficulty and variety of the work in new product development process (Van de Ven et al, 1976), in which the information is difficult to understand, and changes are frequent. People need to apply more knowledge and skills to use the information that is difficult to understand. The information can be explained and changes can be communicated through coordination. Thus, the more complex a product and its process technologies, a higher level of expert knowledge and skills is required. The more tasks that are carried out, the more it is likely that these tasks will have to be coordinated.

If the process is analysable and does not have variables, most of its activities can be standardised and programmed (Perrow, 1970). However, as more and more complex technologies are applied and developed, the development process becomes increasingly difficult to standardise. This is because a greater number of exceptions are likely to arise (March and Simon, 1958) and technologies are more difficult to analyse (Van de Ven et al, 1976). If a product or process is not well understood, it will result in changes in role allocations, schedules and priorities (Perrow, 1967) (Galbraith, 1973). And thus, it is necessary to have group judgements in dealing with complex situations, in a process of collaboration between the different function teams involved in product development process.

4.5.3 Task Interdependence

Task interdependence refers to the degree of dependence of people on the information, resources and materials from other functions in accomplishing their individual tasks. The greater the dependences, the greater the need of coordination between participants will be.

There are three kinds of interdependence in terms of task flow in product development process, namely pooled, sequential, and mutual. A hierarchy of increasing levels of task interdependence can be determined by observing these task interdependences (Thompson, 1967) (Van de Ven et al, 1976). Pooled interdependence implies that each task is part of the process and contributes to the common goal, but each function is also relatively independent because work does not often flow between tasks. Sequential interdependence implies that the output of a task become the input of another tasks in a serial fashion. The first task must perform correctly so that the second task can performed without any problem. Mutual interdependence implies that the output of task T1 is the input of another task T2 and the output of task T2 is the input back into task T1 (Daft, 1994).

Pooled interdependence concerns with standardisation of task executions, rules and regulations.

Sequential interdependence requires further planning and scheduling so that the flow of information and resources are coordinated. Scheduled meetings and face-to-face discussions take place through day-to-day coordination between teams. Mutual interdependence is most complicated because integrated collaboration is required which enable high capacity of information processing (Galbraith, 1973).

In a product development process, the tasks are likely to be more interdependent when more complex product and process technologies are applied and developed. Complex technologies are likely to have complex interfaces and the processes in which will require more attention, thereby increasing task interdependence. Such situations lead to a higher demand for coordination and overlapping of the development activities.

4.5.4 Unit Size

Unit size refers to the total number of people involved in a product development project (van de Ven et al, 1976). In general, the greater the number of people participating in a project, the greater the need for the division of labour, differentiation of functions, and growth of professional staffs will be, which will eventually increase the need for a high degree of coordination of production development process. When a team grows larger (10 or more people), it requires more rules, policies and procedures and increasingly uses hierarchical leaders to control the team members. On the other hand, a relatively flat authority system is used in small teams, even greater use is made of rules, policies and procedures to coordinate work activities (Van de Ven, 1976). In these cases, an increase can be distinguished from the use of rules and regulations, and increase the use of internal systems for control, reward, and innovation (Draft, 1994).

4.5.5 Compliance with Industry Standards

The products and their development processes are required to comply with industry standards. These standards are often involved in an engineering project in order to ensure the required quality or safety of the product can be achieved, such as IEC6108 for safety and ISO9001 for quality. Many standards propose a development lifecycle to deal with all the activities necessary to achieve the required quality. The development process is constrained by these standards in many aspects. First, the objectives of the activities in the framework corresponding to the nature of product must be achieved in its development process. Second, the sequence of achieving these objectives must follow the guideline in the standards. Third, the identified methods or techniques have to be used in achieving these objectives. Forth, the deliverables of each activity are defined and have to be achieved. Finally, the task performers have to possess required skills, qualifications and experiences.

4.5.6 Mechanistic and Organic Systems

There are two extreme management systems that can be used to adapt to a specific technical and commercial change. Stalker (1994) characterise two systems as follows:

"In mechanistic systems the problems and tasks facing the concern as a whole are broken down into specialisms. Each individual pursues his task as something distinct from the real task of the concern as a whole, as if it were the subject of a subcontract. 'Somebody at the top' is responsible for seeing to its relevance. The technical methods, duties, and powers attached to each functional role are precisely defined. Interaction with management tends to be vertical, i.e., between superior and subordinate. Operations and working behaviour are governed by instructions and decisions by superiors. This command hierarchy is maintained by the implicit assumption that all knowledge about the situation of the firm is and its tasks is, or should be, available only to the head of the firm. Management, often visualised as the complex hierarchy familiar in organisation charts, operates a simple control system, with information flowing up to a succession of filters, and decisions and instructions flowing downwards to a succession of amplifiers.

Organic systems are adapted to unstable conditions, when problems and requirements for action arise with cannot be broken down and distributed among specialist roles within a clearly defined hierarchy. Individuals have to perform their special tasks in the light of their knowledge of the task and the firm as a whole. Jobs lose much of their formal definition in terms of methods, duties, and powers, which have to be redefined continually by interaction with others participating in a task. Interaction runs laterally as much as vertically. Communication between people of different ranks tends to resemble lateral consultation rather than vertical demand. Omniscience can no longer be imputed to the head of the concern."

The organic system is appropriate for dealing with a changing environment where the unforeseen requirement cannot be broken down automatically on the basis of the function role defined. On the other hand, the mechanistic form is suitable for stable conditions. In addition, the organic system is inappropriate for in a large project as a large team is difficult to manage. The two systems are sometimes mixed in order to utilise the human resources of an organisation in the most flexible manner.

4.6 Requirements of an Ideal Support System

Engineering processes are relatively complex, dynamic changing, collaborative and unpredictable. A number of challenges are identified which are not addressed by current workflow systems, and which must be addressed by a system if it is to succeed in supporting complex, dynamic changing, collaborative and unpredictable engineering process. These requirements are discussed below.

4.6.1 Compliance Management

The current best practice of project compliance check takes place at only documentation level where the processes in the project have been done and that their results have been documented. If any incompliance output is discovered, its relevant processes may need to be rewound to a suitable state and re-perform again. Currently, a large amount of human effort is consumed in managing project compliance.

An intelligent support system should be able to provide support for managing project compliance at process level. If the incompliance errors can be identified and removed at the time of process planning and execution, the efforts wasted in the re-processing can be dramatically reduced, and thus shorten the development time.

4.6.2 Traceability

Engineering design should be carried out in a manner that shows explicitly how the required requirements are implemented. Therefore, the process rationale (information used to make the decisions during the process) should be recorded so that the members of an engineering team, particularly the assessors, can review the decision path to understand how the requirements are implemented at any time during the development process as required.

4.6.3 Selection of Agent

An engineering process is highly complex and technical. Many standards emphasise that all persons involved in any activity of the development lifecycle, including management activities, should have the appropriate training, technical knowledge, experience and qualifications relevant to the specific duties they have to perform. An engineering project usually spans multiple departments where managers may not have a clear picture of every available agent. In order to be able to assist with the selection of appropriate agents and delegation of tasks to them, the system needs to have access to the knowledge about the capabilities of those required by the process and those possessed by the agents. Together with the availability information, the system can rank the available agents according their degree of fitness so that the most suitable agent can be identified.

4.6.4 Flexibility

No single process model can fit all engineering processes. Each project typically takes a unique form, depending on many factors. While certain characteristics and development processes can be defined, much of the activity details cannot be fully specified at the beginning, since it requires information that only becomes available some way into the project. For example, before the safety integrity level (SIL) of a safety system can be identified, it is not possible to specify the detail of its design process, as different levels of SIL requires different techniques, methods and technical staffs in the development process.

On the other hand, guidelines and norms (which may come from industry standards or organisational culture) are established to deal with the increasing complexity and difficulties of engineering projects. These guidelines and norms are open to interpretation by the project's management team: people trusted with managing projects are experienced and expert in the field; and much is left to their discretion. While expected norms can be stated, such as "never do any work without an approved budget for it", these are open to interpretation and variation in particular circumstances. For example, a project manager might choose to undertake a few days of work without a specific budget in an initial feasibility study for a regular customer where it seems likely that the work will lead to the placing of a valuable contract.

To take account of these situations, the system should allow the process models to be expressed at an appropriate abstract level in the beginning, and details added when they become available. Therefore, it is necessary to interleave the planning and execution of processes, deferring the complete specification of later stages while earlier stages are being executed. In addition, the system should not impose fixed constraints. Instead, constraints should be advisory, so that the manager is aware of situations when he is breaking them, but is still at liberty to do so.

4.6.5 Common Process

There are commonalities between parts of the design process at different levels of detail. As the process model of product development described before, most engineering activities have very similar structures at high level with a consistent breakdown into several design stages, particularly among similar type of projects. At a more detailed level, many of the engineering activities follow some identifiable type of basic processes that reflect common best practice and experience within the industry. These common process structures should be maintained as a resource for those setting up a specific process. The task of process planning is therefore accelerated by selecting the template process that best matches current situation and customising it as required.

4.6.6 Management at Different Levels

Different stakeholders in the process may work at different levels of detail for both process specification and execution. For example, a project manager is interested in the overall project structure, but may delegate management of the details of the engineering activities to other experienced people. Each of them may then further subdivide their respective responsibilities to relevant specialists. Therefore, the system must support this kind of hierarchically structured distribution of management responsibility.

4.6.7 Process and Information Management

Different technical disciplines are involved in different parts of the activities in an engineering project. Many activities are performed in parallel in order to minimise the time needed for development process. These disciplines have to communicate effectively with one another since the work of one discipline may impact on another. The delay, or absence, of information that is crucial for the activities may lead to effort being wasted. Thus, there is a need of a flexible approach where information is mapped to the interested parties so that they can obtain the required information in the first instance.

4.7 Chapter Summary

This chapter introduced the general product development process, and discussed concurrent engineering activities. Engineering processes are different from business processes. The needs for flexibility and uncertainties lead to frequent changes being made to the process plan. The interdependence, unit size and different management styles further increase the difficulty of information management. Compliance management is another challenge for which current systems can only provide support at the documentation level, taking place normally when relevant processes have completed. As such, a number of requirements that have to be addressed by a support system were identified.

The most important requirement identified is the flexibility requirement, emphasising the interleaving between process build time and run time. One serious limitation is that current systems provide no support to ensure a process is planned and performed in accordance with a standard.

Chapter 5

A Flexible Framework to Support Dynamic Process Management

"By seeing the seed of failure in every success, we remain humble.
By seeing the seed of success in every failure we remain hopeful."
- Mel Ziegler

5.1 Introduction

Flexibility is one of the most important requirements for a system that is to succeed in supporting engineering processes. While workflow management systems (WfMS) are widely used in providing support for well-defined and predictable "administrative" processes, current workflow products are not flexible enough to handle complex, dynamic changing, collaborative engineering processes and to ensure their compliance with a selected standard. More seriously, they lack the ability to ensure the planning and the execution of a process is compliant with an industry standard.

This chapter presents the novel framework of Compliance Flow system which provides a platform to enable the integration of various technologies to deal with more complex and flexible engineering processes. It is organised as follows: §5.2 presents the novel framework and describes the functions of its components; §5.3 provides a comprehensive view of process management in the system; §5.4 concludes the advantage of this framework against the challenges identified before; a summary is given in §5.5. Some examples drawing from a draft version of IEC61508 safety standard are used.

5.2 Compliance Flow Framework

Compliance Flow is designed as a process management tool, rather than a task specific tool for performing specific computation, such as risk calculation. Instead of a single, fully integrated and monolithic IT system, its approach is to provide a process management framework that integrates existing tools. This is particularly important in highly technical fields where the use of specialist third-party software, such as simulation and analysis tool, is essential (Stader et al., 2000).

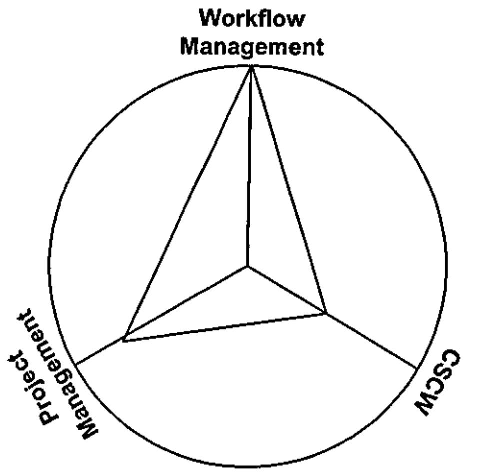


Figure 5-1. The consolidation of three types of support systems in Compliance Flow.

Compliance Flow consolidates three types of system function together, as depicted in Figure 5-1. It serves three types of stakeholders involved in an engineering project:(1) the people who manage the project, such as project managers and team leaders; (2) the people who perform the tasks in the project, such as technical engineers; and (3) the people who are responsible for quality assurance, such as quality assessors.

The project management function enables managers to plan the project and track its progress using workflow technology. Unlike the typical project management tools which only serve the managers' needs, Compliance Flow is intended to be used by all stakeholders of a project where the tasks are performed under the control of the workflow engine according to the specified process. As the results are updated quickly after an individual task has been performed, the latest progress of the project can be shown instantly at any time by request. The project compliance is managed at the process level, which enables the identification of compliance errors and concerns in advance of task execution. The integration of CSCW enables the sharing of information between stakeholders, assisting, for example, a quality assessor to easily retrieve the required information. A framework developed to support the proposed approach is depicted in Figure 5-2.

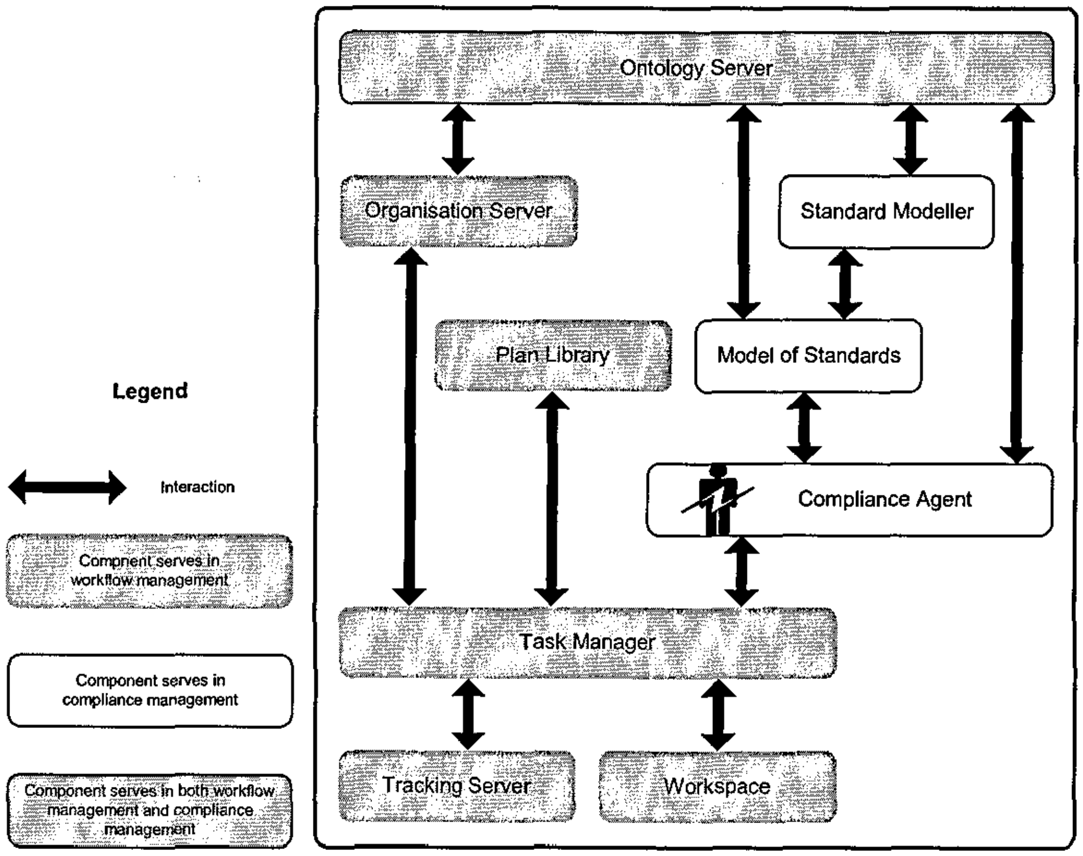


Figure 5-2. Framework of Compliance Flow.

5.2.1 Ontology Server

As communication will take place among components as well as between systems and users, the terms used in describing a concept of interest must be consistent. A set of ontologies is adopted to enable communication. These ontologies are available within the system and are maintained by the ontology server. Users are allowed to add, change, remove or extend an ontology to adapt to a particular environment where the system is running. It is noted that the ontologies provided by Compliance Flow do not follow a particular industry standard in the context of workflow management. Other works like Workflow Management Coalition Terminology & Glossary (WfMC, 2000a) and Enterprise Ontology (Uschold et al., 1998) provide ontologies with more rigorous definitions. The following ontologies are part of Compliance Flow:

Process Ontology. This ontology describes the processes and their activities in the context of development process, including tasks, pre-conditions and post-conditions.

Capability Ontology. This ontology describes the domain specific skills possessed by agents and the skills that are required to perform particular tasks.

Application Ontology. This ontology describes the application areas of the domain specific skills defined in Capability Ontology.

Technique Ontology. This ontology describes the techniques or methods that are used to perform tasks.

Artifact Ontology. This ontology describes the physical equipment, or tools used in performing particular tasks.

Document Ontology. This ontology describes the information required for performing a task or the information created during a task, such information is normally stored in a document.

The terms of an ontology in Compliance Flow are organised into a hierarchy in which a term located in a higher level implies a higher level of abstraction, while a lower level term represents a more concrete concept or object. A term can be changed, removed or extended if necessary. Therefore, a concept can be refined by decomposing it into a set of such concepts. An example hierarchy is given in Figure 5-3.

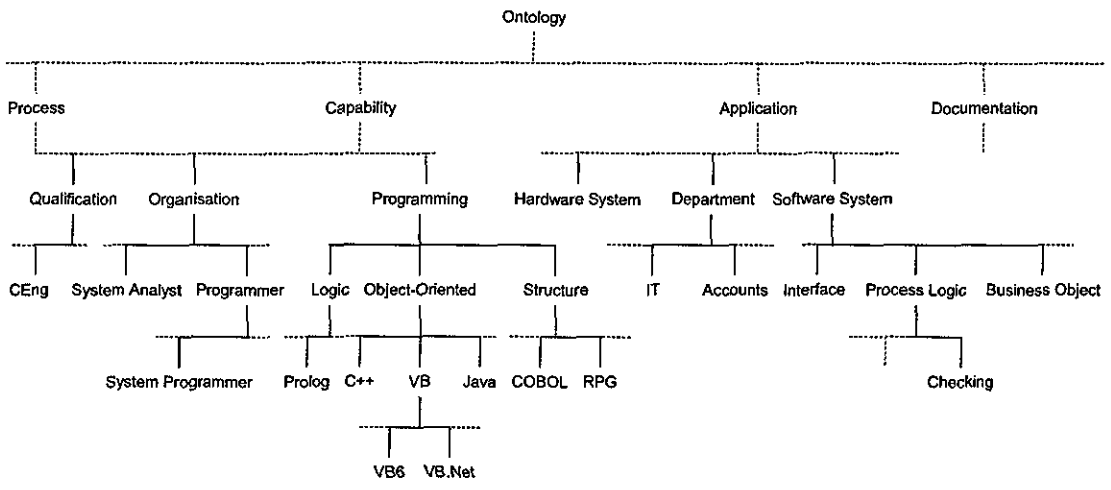


Figure 5-3. An example of ontology hierarchy.

The terms at any level in the hierarchy can be used in naming an appropriate object in the system. A term may become outdated or may need to be changed. An example is that the term Electronic Data Processing (EDP) is replaced by Information Technology (IT). This change can be performed at any time. Once it has been updated, all its instances in the system will be automatically changed to maintain consistency.

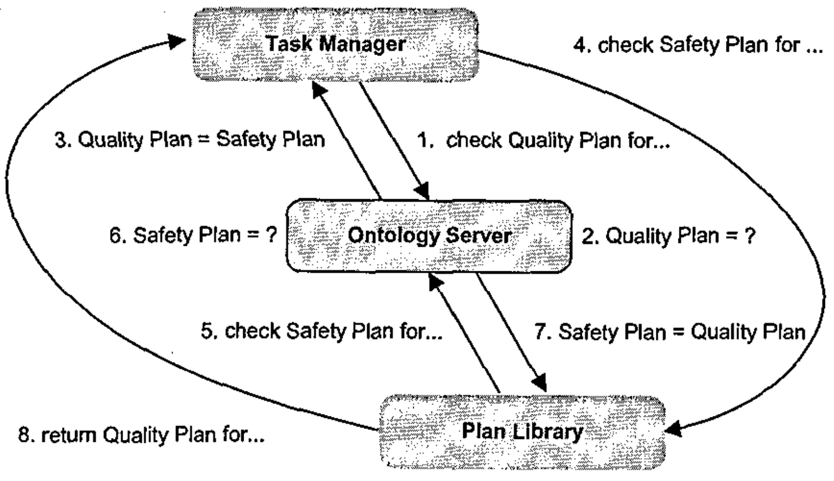

Task Manager
4. check Safety Plan for ...
3. Quality Plan = Safety Plan
1. check Quality Plan for...
6. Safety Plan = ?
Ontology Server
2. Quality Plan = ?
5. check Safety Plan for...
7. Safety Plan = Quality Plan
8. return Quality Plan for...
Plan Library

Figure 5-4. Ontology Server as a translator.

Occasionally, two terms that represents the same domain of interest will coexist in an organisation, particularly when two different domain teams are working together. For example, Quality Plan has the same meaning as Safety Plan in some way. To deal with this, a synonym can be rewritten to the ontology server if required at any time. The only constraint is that the synonym cannot be a duplicate of other terms used in the system. Ontology server will perform the translation between the terms and their synonyms if necessary. Figure 5-4 illustrates the use of Ontology Server as a translator. The synonym "Quality Plan" that is used by Task Manager is translated into the original term "Safety Plan" that is used by the Plan Library.

5.2.2 Organisation Server

It is critical in an engineering project that all persons involved have the appropriate qualifications to perform the tasks that they are assigned to. Thus, the selection of agents to perform specific tasks based on the knowledge about their abilities and roles within the organisation is a key feature that an intelligent workflow management system needs to offer (Moore et al. 2000). An agent here refers to a person, a software system or a machine which is involved in executing a process. To provide decision support in this area, the knowledge of the capabilities required of agents to perform certain tasks and the capabilities possessed by the agents are captured and maintained in the Organisation Server. Capability matching can be performed to identify the most appropriate agent for performing a task.

Agent selection in current workflow systems is mainly based on some pre-defined agent selection polices (Rupietta, 1997) (Kappel et al., 2000). There are mainly three kinds of agent selections policies: (1) agent related selection – select an agent by means of its identifier, (2) role related selection – select an agent who plays a certain role, and (3) workflow related selection – select an agent based on various data about actual and previous workflows. The initial concept of using dynamic capability matching to select a suitable agent for a task was proposed by Moore et al. (2000). A more flexible approach is proposed which extends the capability matching to not only considering technical capabilities but also organisation knowledge and authority. The details of capability matching are described in Chapter 8.

The reason for using dynamic capability matching function in a workflow context is that it is impossible to predict the exact environment where a process is executed, as a consequence of the uncertainties and changes in a process or an organisation. For example, specific agents may not be available at the time of task execution (people take holidays or leave the organisation), or more suitable agents may have become available (new people are employed or new systems are developed). A system that can instantly identifies the most suitable agent for task execution is critical, particularly for engineering projects where the processes span multiple departments and penetrate various management levels.

The Organisation Server also provides support for handling resource conflict, e.g. the most capable staff for performing a particular task is not available at the time of process execution. Availability of task agents not only has an impact on assigning tasks, but also on the decision on which method to use to achieve a given task. If a method for carrying out a task requires a particular capability but currently there are no available staff with that capability, then an alternative method for achieving the task is sought.

5.2.3 Plan Library

Engineering processes have a similar structure at a high level as they are planned based on similar type of canonical process models. At a more detailed level, many activities follow some identifiable type of basic process that reflects common best practice and experience. These process structures at any level of a hierarchical task network (HTN) can be saved as a process template called a Plan. A plan can be a set of tasks representing a process at an abstract level, or with the detail as a single HTN. Thus, a Plan represents one possible way of achieving a given type of task by breaking it down into a particular structure of sub-tasks.

The Plan Library maintains a database of plans. A hierarchical folder structure is provided, with each folder containing solution plans for a task at a specific level. As more plans are stored in the library, automatic plan selection to meet the requirements of a given task is possible.

An engineering process is usually required to be refined by breaking it down into further set of sub-processes. Its decomposition continues until the required detail is reached. For each sub-process, a plan may exist in the library which can be selected to specialise the sub-process so that a multi-level hierarchical task network can be generated. If necessary, users can modify it to adapt to a particular environment or situation.

5.2.4 Workspace

In an engineering project, many engineers are required to work together where each person is responsible for a part of the design, sharing information among them as they are created during the design process. The final solution will emerge as different pieces of information are combined. Hence the concept of workspace is integrated into process management to provide a highly transparent environment to support collaborative design work.

Each task in the task hierarchy is associated with a workspace in which the task related information, such as the design requirements and specifications, are linked. Once an agent is assigned to a task, he/she will become the owner of the associated workspace and is able to manipulate all the objects in the workspace. The key concept of workspace is shown in Figure 5-5.

In Figure 5-5 and Figure 5-6, the rounded rectangles with identifiers beginning with the letter T represent the tasks of a process and the rectangles with identifiers beginning with the letters WT represent the workspaces. A document is represented using a document icon.

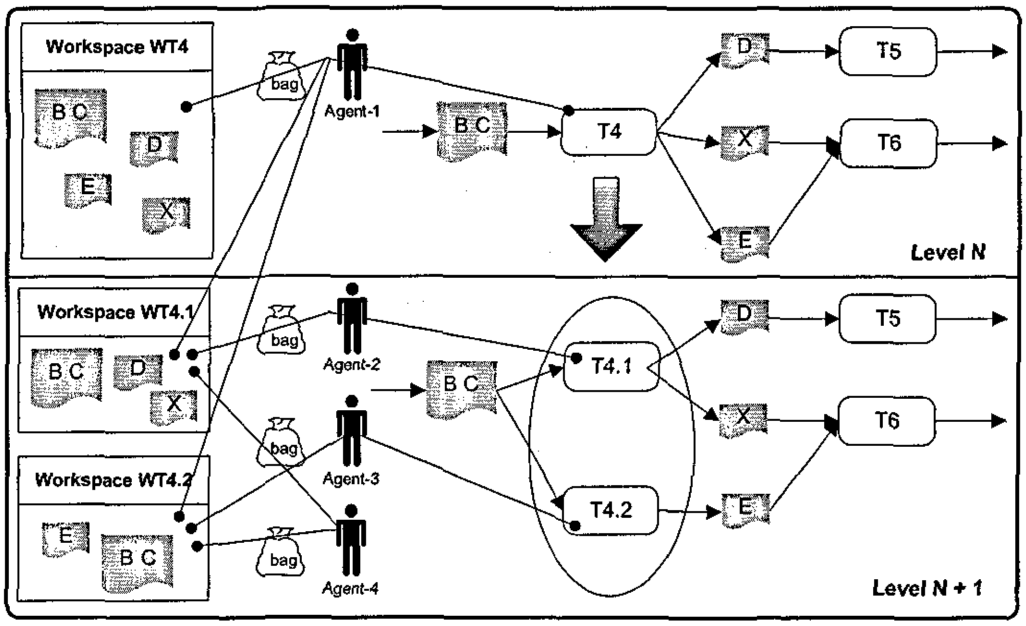


Figure 5-5. Tasks and their workspaces.

In Figure 5-5, the level N workflow represents a higher level of abstraction of a process than level N + 1. Agent-1 is appointed as responsible for task T4 and becomes the owner of the associated workspace for task T4, denoted as WT4. Agent-1 subdivides task T4 into sub-tasks T4.1 and T4.2, which are then assigned to engineers Agent-2 and Agent-3 respectively. Therefore, Agent-2 becomes the owner of the workspace WT4.1 and Agent-3 becomes the owner of the workspace WT4.2.

When a task is decomposed into a set of sub-tasks, the parent task owner can access the workspaces associated with the child tasks, but the child task owners are not allowed to access to the higher-level workspace. Because Agent-1 is the owner of the parent task of T4.1 and T4.2, he, therefore, becomes a member of the workspaces WT4.1 and WT4.2. However, Agent-2 and Agent-3 both are not allowed to access the workspace WT4.

A number of staff can be a member of any workspace by making an application to the workspace owner. Once the application has been accepted, the staff will then gain the access rights granted by the owner. In the example, Agent-4 is a member of the quality assurance team who is required to frequently access the documents created in T4.2.

In Figure 5-5, the same document BC exists in the workspaces WT4, WT4.1 and WT4.2. There is, however, only one copy of BC which is stored in a secure place, i.e. an FTP server where hyperlinks are used to link the document to the appropriate workspaces. Other document management features, like version control, can be achieved by integrating workspace with third party systems. In addition, objects stored in a workspace can be any information represented in a digital form, such as Computer Aided Design (CAD) diagram or video clips.

The concept of workspace also contributes towards information transmission between tasks within a project. In an engineering project, an output of a task usually will be used as an input to one or more subsequent tasks. When this happens, the workflow engine will link the output to the workspaces where it will be used as an input. An example of the information transmission is given in Figure 5-6.

In Figure 5-6, because document B is the output of task T1 and is required for performing the tasks T2 and T3, it is defined as a post-condition of the task T1 and a pre-condition of the tasks T2 and T3. When the task T1 is completed and the document B is uploaded to the workspace WT1, a link to the document will appear in the workspaces WT2 and WT3 for downloading and therefore the pre-conditions of tasks T2 and T3 will be fulfilled automatically. Similarly, when the tasks T2 and T3 are completed, links to the documents C and D will appear in the workspace WT4 automatically.

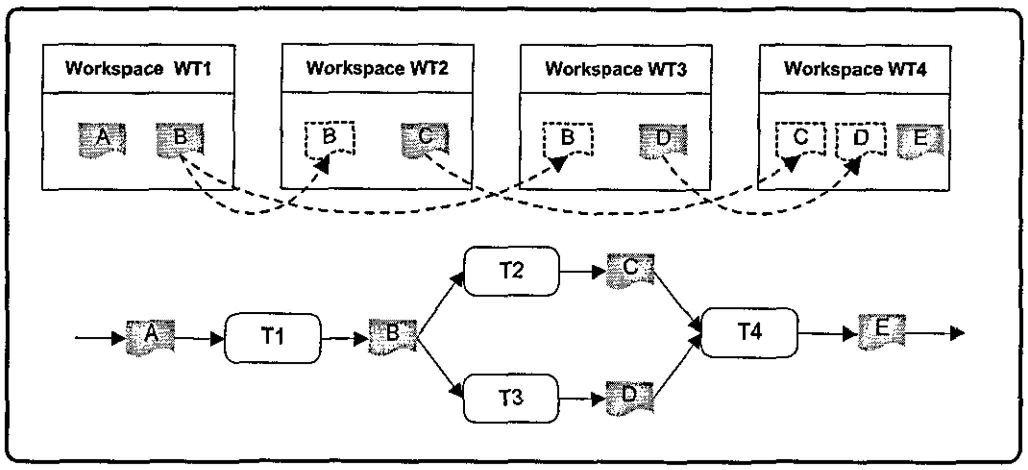


Figure 5-6. Information transmission across workspaces.

Workspaces provide an all-in-one view of the processes at any level of abstraction. While high-level workspaces provide a broader view, the leaf workspaces focus on the required information for particular tasks. However, this kind of information sharing may cause inconvenience. For example, an agent may be reluctant to expose a document which is still a draft. Compliance Flow provides every agent a private space called a 'Bag'. The objects in a bag can only be accessed by the owner. Thus, Agent-1 is prohibited from accessing other agents' Bags, including the ones belonging to his subordinates, even though he is the owner of the parent task.

5.2.5 Tracking Server

The Tracking Server records the rationale of process planning and execution by logging the transactions of the objects in a workspace and capturing the cause of abnormal decisions. During the development process, information is moved from one workspace to another for further processing until the expected deliverables are achieved. The transactions of information objects are recorded by the Tracking Server. Once an abnormal decision is made, the rationale of that decision has to be provided by the user and is stored in the Tracking Server. The following actions will be treated as abnormal operations and each would lead to a request from the Tracking Server for an explanation:

- A structure change occurring after a process has started. Structure changes include addition, insertion or removal of tasks in a process.
- The rewinding of the status of a task. For example, rewind from 'Completed' to 'Active'.
- Adoption of a non-recommended technique to perform a task.

- Employment of a non-qualified person to perform a task.
- Forced to perform a process that does not fully comply with the required standards.

The decision rationale is recorded by the Tracking Server to form a process rationale database so that the members of the engineering team can review the decision path to understand how the design requirements are implemented and at any time during the development process if necessary.

5.2.6 Task Manager

Task Manager is the presentation layer of a workflow engine through which users model and manage their tasks. Task Manager integrates two key workflow system functions in its interface: process planning and user work-list management, and allows interleaving between them.

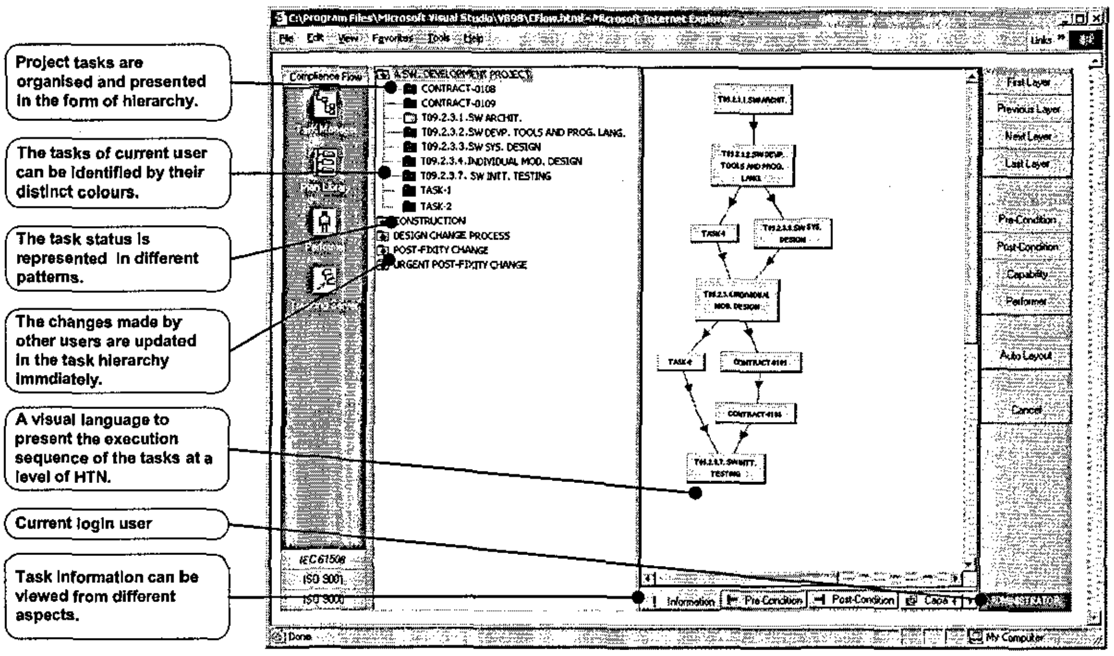

Project tasks are organised and presented in the form of hierarchy.
The tasks of current user can be identified by their distinct colours.
The task status is represented in different patterns.
The changes made by other users are updated in the task hierarchy immdiately.
A visual language to present the execution sequence of the tasks at a level of HTN.
Current login user
Task information can be viewed from different aspects.

Figure 5-7. A screen shot of the Task Manager.

A screen shot of the organisation of Task Manager is given in Figure 5-7. The task hierarchies are presented on the left hand side where the tasks belong to the current user together with their status can be identified by their distinct colours and patterns. Process planning is performed using the Process Planner located at the right hand side. The task flow is modelled and presented graphically. Planning starts from the high level by selecting appropriate sub-plans in the library and adapting them to the current situation if necessary. Plan refinement is done level by level until a sufficient level of detail is reached. Once a plan is updated (whether by the current user or others), the changes made are updated to the task hierarchy inside the Task Manager of every user to keep the information up to date. In addition, the management of pre- and post-conditions, task capability, and workspace, is performed by the Task Manager.

For the normal users (beside the system administrator), the Task Manager is the only tool to be used to manage their activities. Task Manager is able to provide users with a clear picture of all their tasks with the interdependence among all other tasks in HTN, including all projects, in a project-based perspective.

The interleaving between process planning and execution becomes possible as a consequence of their integration. The interleaving implies that a portion (normally the earlier part in the process flow) of a process plan can be performed while the rest of plan is not complete, and the planning is on going. This situation often occurs in an engineering process where the planning of further tasks is dependent on the execution result of some previous one. As the results of some tasks become available, the scope of the process, including the completed part, has to be re-planned to better match the project objectives. This iterative process of planning and execution is handled by Task Manager without any problem.

Collaborative planning is also possible in the Task Manager. Collaborative planning occurs in an engineering project in both horizontal (people with similar authority but in different disciplines) and vertical (people with different authorities but in the same professional discipline) organisation directions. As all users use the Task Manager for process planning and work-list management, any changes to tasks done by a user will instantly be reflected on the Task Manager's interface. All team members will be made aware of them without delay.

The task flow cannot be obtained from the hierarchy provided in the Task Manager, and the process planner can only display the relationship of tasks that are at the same level in HTN with the same parent task. The "penetration" feature provides a more transparent view of the task hierarchy. The penetration function allows a user to view a hierarchical process model visually, starting at any level, moving upward or downward level by level, with the ability to add or remove any task attributes, such as agent information, in the visual interface at any time. An example of a penetration view is given in Figure 5-8.

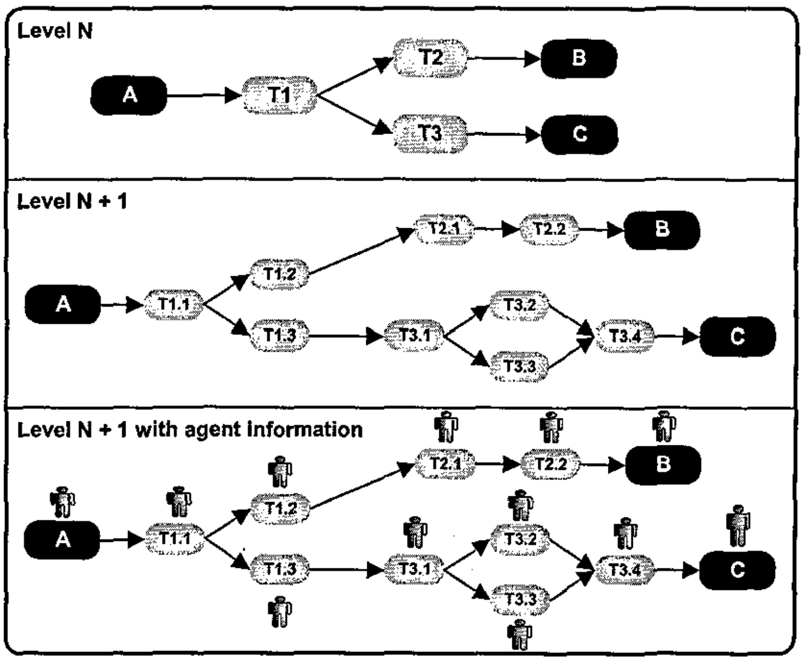


Figure 5-8. An example of penetrative view between two levels in a HTN.

5.2.7 Model of Standards

The Model of Standards acts as a knowledge database system providing information about the standards for process management. The information is used in the compliance checking performed by Compliance Agent. Four types of information about a standard are modelled:

- The tasks framework of the standard development lifecycle that is used to deal with all the necessary activities to achieve the acceptable quality of products or services.
- The requirements and deliverables of every task in the lifecycle.
- The techniques, measures, tools or methods that are recommended to be used to achieve specific objectives or requirements.
- The required capability of task agents. Capability refers to qualifications, roles, experiences or other attributes identified by a standard, which a staff must possess in order to be qualified to perform a specific task.

The Model of Standards that comes with the system is capable of providing information on a number of standards. It allows users to change the information as well as adding new standards. Therefore, the standards modelled in the system can fully represent their actual applications in an organisation. For example, suppose the full version of IEC61508 safety standard comes within the Model of Standards. If the business of a company only concerns safety software development, the users may consider removing the hardware development section in the Model of Standards. In addition, users can amend it to the required details for which the organisation is using. Changes to the Model of Standards are made using the Standard Modeller.

5.2.8 Standard Modeller

Standard Modeller provides a visual interface for the user to model a new standard or amending an existing standard in the Model of Standards. The modelling is performed using a visual process modelling language. The design of the Standard Modeller is similar to that of the Task Manager. The left side of the Standard Modeller is the task hierarchy; the right hand side provides further information about the tasks, which includes task flow, pre- and post-conditions, capabilities and recommendations. To enable communication and prevent misunderstanding between different stakeholders, all the terms used in naming the objects of a standard have to be chosen from the Ontology.

5.2.9 Compliance Agent

Compliance Agent is a software agent that is responsible for ensuring the planning and the execution of a process is compliant with a selected standard. It will continually monitor the planning and execution of the engineering activities, providing proactive assistance in managing the project compliance. Compliance Agent performs the following duties:

Error identification and prevention:

- Correctness Check – to ensure the sequence of tasks specified in a user-defined process is in accordance with a selected standard.
- Completeness Check – to ensure all the required tasks within a standard are defined in the user-defined process.
- Capability Check – to ensure the required capabilities of an agent are specified according to a selected standard.
- Recommendation Check – to ensure the recommended techniques, measures, tools or methods for performing a particular task are fully considered.

Process management assistance:

- Planning Assistant – to remind the user of possible pre-conditions, post-conditions, recommendations, and required capability of a particular task while it is being planned.
- Information Navigation – to transfer the required information, once it is available, to the tasks where the information may be used for their execution.
- Cross-Referencing – to refer a user-defined task to the relevant part of a standard or vice versa.

5.3 Process Management

The general lifecycle of a workflow comprises two stages:

1. Planning: decide what and how it is to be done.
2. Running: do it.

In conventional workflow systems, these two stages must be performed in sequence. This restriction is not suitable for an engineering process where many tasks may be too complex or unpredictable to be fully specified in advance. Furthermore, some other tasks may have to be performed in a dynamic environment that leads to frequent changes to the original plan.

The Compliance Flow's process management approach allows the interleaving between the two stages as long as any individual task execution follows the associated constraints. Each of the two stages is outlined below.

5.3.1 Planning

Planning is brought into the system either as a new top-level activity, such as for a completely new project, or the need to create part of a plan to implement another process specification. Initial planning support is provided by the process planner in the Task Manager, which can be used to edit the structure and detail of sub-plans in the Plan Library, enabling the owner of a process to adapt the plan to the current situation. Such changes can be made in advance of execution, but they can also be made when the overall process has already started. Collaborative planning is supported where agents can plan their tasks concurrently; a complete plan can be delivered by integrating all these partial plans.

A new method (a sub-process) of the task execution can be saved as a Plan if an agent believes that it may be possible to be reused in the future or may be useful to others. While a process is decomposed into sets of more concrete sub-processes, the details of a sub-process may be found in the Plan Library. Therefore, a multi-level hierarchical task network may be generated by bringing together of many Plans. The necessary changes can be made to the process plan to adapt it to a particular situation.

When a task agent decides to delegate a task to another more appropriate agent instead of performing it himself, he can directly appoint another agent or use the capability matching facility to identify the most suitable available agent before assigning the task.

During process planning, the Compliance Agent provide pro-active assistant in ensuring that the process plan complies with the standards by performing the compliance checks where the user will be notified if any incompliance error in the plan is discovered and reminded if any possible solution for specifying a current task is found.

5.3.2 Running

A workspace can be viewed as a virtual information container of a process at any level of the hierarchy task network. Once a process is initiated, its associated workspace is created. From that moment onwards, its members can upload or download information. Uploading information takes place either when a task has been completed with the deliverables being uploaded to the workspace or when a document requires sharing with others. When a deliverable of a task is uploaded, a link to that deliverable will be sent to the workspaces where it is required as a pre-condition.

During run-time, the Tracking Server registers many types of operations carried out by the task agents on the objects inside a workspace. If a process needs to be undone, the Tracking Server will provide the history and guide the user to undo the process, returning it to a suitable state.

When an agent starts execution of a task, the Compliance Agent will perform compliance checks and the task will be prohibited if any error is discovered until an explanation is provided or the error is eliminated.

5.4 Chapter Summary

This chapter presented the novel Compliance Flow framework, which is able to deal with the identified requirements for supporting engineering process.

The hierarchical task-based process model is capable of modelling and managing complex engineering project. The interleaving between process build and run time allows an abstract process plan to be performed in the beginning and the details added when they are available. It forms an essential part of a system to support engineering processes.

The use of ontology facilitates communication between the system stakeholders, including system components and users. It is also the key element to enable the compliance checks. By using dynamic capability matching, the Organisation Server can assist in identifying the most suitable task agent for a specific task. As the Plan Library becomes rich along with the use of the system, the hierarchical task network planning and case-based planning can make process planning easier.

The workspace provides a base to allow collaboration among task agents. The rationale for engineering process is captured by the Tracking Server so that engineers can trace back the decision path at any time if necessary. Task Manager gives an integrated working environment, allowing users to manage their tasks and keeping updated with project progress.

A Compliance Agent is used to provide support for compliance management. By performing a number of compliance checks, it can effectively identify the compliance errors and proactively prevent unqualified products from being produced.

In conclusion, the requirements for a supporting system identified are dealt with by the Compliance Flow approach in the following ways:

- Compliance with Standards: A Compliance Agent performs a number of compliance checks between the Model of Standards and the user-defined processes during process build and run time. As a result, compliance errors are identified during process planning, and tasks with compliance problems are prohibited from execution during run time.
- Traceability: A Tracking Server records decision rationale so that the implementation and decision paths can be traced.

- Selection of Agent: An Organisation Server provides a fuzzy capability matching mechanism for this purpose.
- Flexibility: The interleaving between process build and run time allows parts of a process to be executed while other parts are being refined.
- Common Process: A Plan Library provides support in the form of pre-specified process structure that can be assembled and adapted to suit the specific needs of the current situation.
- Process and Information Management: The process management facilities together with the provision of workspaces provide a collaboration environment for engineering processes. The information will be transferred to the relevant tasks once they are ready. The personal 'Bag' can effectively maintain the necessary privacy for each agent.

Part 3

Managing Process Compliance

Chapter 6

Standards Modelling

"What we observe is not nature itself,
but nature exposed to our method of questioning."
- Werner Heisenberg

6.1 Introduction

To automatically perform the process compliance evaluation in a software system, an essential requirement is the ability to retrieve the information about a standard. In Compliance Flow, the Model of Standards is responsible for providing such information. It describes the processes that have to be followed to develop a standards compliance product. The traditional process modelling concepts are insufficient in describing the processes recommended by standards. To develop an approach to modelling a standard, it is necessary to understand the standard and analyse the key requirements that have to be fulfilled to achieve the required level of compliance. The IEC61508 international standard is used to explain the proposed approach.

This chapter describes and discusses the standard modelling in Compliance Flow. It is organised as follows: §6.2 gives a general introduction to IEC61508 international standard and identifies the key requirements for developing a standard compliance product; §6.3 discusses the perspectives of process modelling and points out that an extra technological perspective is necessary for modelling a standard; §6.4 presents Compliance Flow's approach to model a standard; §6.5 is the summary of this chapter.

6.2 IEC61508 International Standard

IEC61508 is an international standard for the development of Electrical/ Electronic/ Programmable Electronic Systems (E/E/PEs) that are used to perform safety functions. There are two different groups of sub-systems in IEC61508: the Equipment Under Control (EUC) and safety-related system. The former performs the required manufacturing, process, transportation, medical or other activities, while the later provides the safety functions required to ensure that the EUC is suitably safe. The scope of IEC61508 is limited to the functional safety of the E/E/PEs and is concerned with hazard analysis on the EUC in order to identify requirements for safety-related system. The design issues relating to the EUC is not included.

6.2.1 Structure of IEC61508

IEC61508 consists of three main parts, together with definitions and guidance. Part 1 specifies the general requirements that are applicable to all parts of IEC61508. In addition, it provides an introduction to the safety lifecycle and the overall structure for the technical requirements for safety-related E/E/PES. Supplementary information on concepts and suggested methods is in Part 5. Part 1 sets out requirements for:

- Conformance (Clause 4)
- Competence of persons (Clause 5)
- Safety management (Clause 6)
- Overall safety lifecycle (Clause 7)
- Functional safety assessment (Clause 8)
- Documentation (Clause 9)

Part 2 presents a safety lifecycle for the hardware aspects of E/E/PEs. The requirements for the design, test and validation of the hardware used in safety-related E/E/PEs are presented in Clause 7 of Part 2. In particular, there is useful detail on:

- The architecture of E/E/PEs.
- The guidance on architectures encourages consideration of the sensors and actuators that form part of the safety function.
- The detection and protection against component faults for the components of hardware architectures, including measures to detect and protect against random failures in hardware which relate to Safety Integrity Level (SIL)
- The avoidance of design errors for hardware related to SIL.

Part 3 specifies the requirements for software, with some additional guidance in Part 6. It also identifies the measures and techniques for specific software development activities within the lifecycle. These techniques are listed in Annex A of Part 3 and supplemented by an example of their application in Part 6. These requirements are particularly for:

- The selection of requirements and design methods and notations related to SIL.
- The levels of testing and the types of test related to SIL.
- The choice of programming language related to SIL.
- The hazard analysis of software designs related to SIL.
- The software validation techniques and level of independence of validation related to SIL.

IEC61508 has two important concepts which are fundamental to its application – namely, Safety Lifecycle and Safety Integrity Level. Safety Lifecycle forms the central framework which links together most of other concepts introduced in this section.

6.2.2 Safety Lifecycles

A Safety Lifecycle is used, as the key framework, to deal in a systematic manner with all the activities necessary to achieve the required Safety Integrity Level. The Safety Lifecycle encompasses the risk reduction measures that include E/E/PES safety-related systems, "other technology" safety-related systems, and external risk reduction facilities.

The Overall Safety Lifecycle proposed by IEC61508 is shown in Figure 6-1 with simplified views of reality and as such do not show all the iterations relating to specific phases or between phases. However, the iterative process is an essential and vital part of development throughout the Safety Lifecycles.

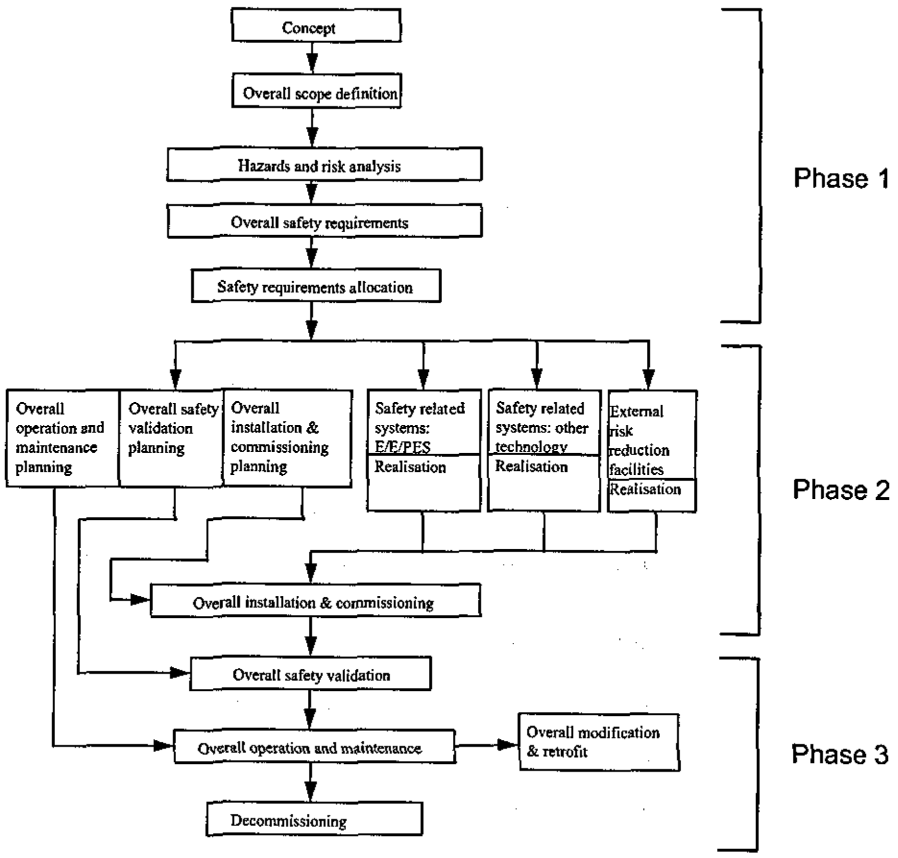

Concept
Overall scope definition
Hazards and risk analysis
Overall safety requirements
Safety requirements allocation
Overall operation and maintenance planning
Overall safety validation planning
Overall installation & commissioning planning
Safety related systems: E/E/PES
Realisation
Safety related systems: other technology
Realisation
External risk reduction facilities
Realisation
Overall installation & commissioning
Overall safety validation
Overall operation and maintenance
Overall modification & retrofit
Decommissioning
Phase 1
Phase 2
Phase 3

Figure 6-1. IEC61508 Overall Safety Lifecycle

IEC61508 requires that the Overall Safety Lifecycle shall be followed (or concluded) by a verification activity, and that this verification shall be planned. The verification techniques to be applied will depend upon the specific phase but shall include such things as document and design reviews, testing and analysis of results.

6.2.3 Safety Integrity and Safety Integrity Level

In IEC61508, safety requirements are defined as safety functions together with a safety integrity level for each function. The safety integrity is the probability of a safety-related system performing the required safety function under all stated conditions within a stated period of time. The Safety Integrity Level (SIL) is one of four discrete levels for specifying the safety integrity requirements. Safety integrity levels are numbered from 1 to 4 with 4 having the highest level of safety integrity. Table 6-1 shows the safety integrity levels and relates them to quantified probabilities of failure for systems. The probability figures are noted as indicative in IEC61508 and could be altered to suit different industries.

Safety Integrity Levels: Target Failure Measures		
Safety Integrity Level	Demand Mode of Operation (Probability of failure to perform its design function on demand)	Continuous / High Demand Mode of Operation (Probability of a dangerous failure per hour)
4	$\geq 10^{-5}$ to $< 10^{-4}$	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-4}$ to $< 10^{-3}$	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-3}$ to $< 10^{-2}$	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-2}$ to $< 10^{-1}$	$\geq 10^{-6}$ to $< 10^{-5}$

Table 6-1. Safety Integrity Levels

6.2.4 Requirements for Development Safety Equipments

Similar to other safety standards relating to safety-related systems, IEC61508 defines a set of requirements for developing E/E/PES for use in safety-related systems. Implementing such requirements is the current best practice for developing high quality product. The key requirements (ERA Technology, 2000) are:

Quality and Safety Management

Requirement 1: The development shall be performed in accordance with a defined quality and safety plan.

A clearly defined quality plan and a separate safety plan for complex safety-related system that identify all development activities, the methods and associated procedures to be followed in performing the identified activities, and the allocation of responsibilities for performing the identified activities is necessary. The purpose of the quality plan is to facilitate the overall development process management. The process of overall development can be reviewed at any point during the development and the misunderstandings of identified activity among performers can be avoided.

Requirement 2: Suitable techniques and measures shall be selected.

Suitable techniques, measures and supporting tools shall be selected for each of the identified activities. IEC61508 identifies possible techniques and measures that should be consider for developing E/E/PES with different safety integrity level targets. The basic principle in making a selection is that the use of the selected techniques, measures and tools should reduce the risk to a level that is As Low As Reasonably Practicable (ALARP).

Requirement 3: The project organisation and allocated responsibilities shall be defined.

The project organisation should be clearly defined in terms of the process of interactions between the individual members of the organisation and the allocation of responsibilities. The persons who have responsibilities for any safety lifecycle activities should be competent to discharge those responsibilities. In particular, it should be clear on how the responsibilities for addressing the safety of the equipment will be shared amongst the members of the organisation.

Requirement 4: Configuration management and change control procedures shall be defined.

Procedures for managing changes, both during the initial development of the equipment and subsequent design corrections and enhancements, should be clearly defined and understood by all staff involved in the development process.

Requirement 5: Design reviews shall be planned and carried out.

The activities of design review should be included in the quantity plan.

Requirement 6: Document shall be produced.

The results of all development and review activities should be documented and presented in an auditable form.

Safety Requirements

Requirement 7: Hazard analysis and risk assessment shall be performed.

The hazards associated with the safety-related system application and the associated risks should first be identified in order to derive safety requirements. The resulting requirements should then be analysed in the context of the identified hazards and risks to ensure that they are correct and complete with regard to all possible eventualities.

Requirement 8: The specification of safety requirements shall be documented.

The safety requirement should be clearly specified, using an appropriate notation, in terms of the required functionalities. The safety requirements specification should also include a definition of a required safety integrity level, which is based on the results of the hazard and risk analysis activities.

Requirement 9: There shall be clear tractability from requirements through design.

The design of a safety-related system should be carried out in a manner that shows explicitly how the safety requirements are implemented.

Architecture

Requirement 10: Appropriate programmable electronics architecture shall be selected.

The suitable E/E/PES architecture should be designed based on the required safety integrity level of the design.

Design

Requirement 11: Suitable design techniques shall be employed depending on the required safety integrity level.

The design of the E/E/PES should be performed in accordance with the quality and safety plan in which the methods, techniques and measures to be adopted are specified.

Test and Analysis

Requirement 12: Test specification shall be prepared prior to testing.

The test to be performed on the design should be specified in parallel with the actual design activities, and should be clearly documented.

Requirement 13: The results of test and analysis activities shall be recorded.

The results of test and analysis activities shall be documented for subsequent review.

Independent Safety Assessment

Requirement 14: The development shall be subjected to independent safety validation and assessment.

The development process, and the results of the process, should be subjected to review that is performed by a suitable competent individual or organisation that is independent of the development team.

6.3 Standard Modelling Consideration

To describe a process related to a standard, several concepts called process perspectives (Curtis et al., 1992) are necessary.

6.3.1 Functional and Behavioral Perspective

First of all, processes contain functions to be executed. Examples of such functions in IEC61508 are hazard and risk analysis, overall safety requirement or overall safety validation. These functions are examples of tasks. Every task has a particular objective. If a task is too complicated, it can be decomposed into sub-tasks. For example, overall safety validation can be decomposed into hardware validation or software validation. The task decomposition can be performed continually if necessary, forming a hierarchical task network (HTN). The lowest level tasks in the HTN are called leaf tasks. A leaf task is performed normally by a human or a machine and produces a single deliverable. For example, designing a car is a complicated task which consists of a set of sub-tasks each of which further consists of other sub-tasks and so on. Finally, hundreds of functions are executed to build a car.

The behavioral aspect of a task defines in which order its sub-tasks are performed. The order is specified through control constructs such as sequence or parallel branching, and is constrained by the pre-conditions of tasks. A task can only be performed when its previous tasks have been completed and its pre-conditions have been fulfilled.

Applying to the recommended processes of IEC61508, functional perspective concerns what are the required processes and behavioral perspective defines in which order they are performed. Following are the key requirements of applying IEC61508. concerned by these two perspectives:

- Requirement 1: The development shall be performed in accordance with a defined quality and safety plan.
- Requirement 4: Configuration management and change control procedures shall be defined.
- Requirement 5: Design reviews shall be planned and carried out.
- Requirement 7: Hazard analysis and risk assessment shall be performed.
- Requirement 10: Appropriate programmable electronics architecture shall be selected.
- Requirement 12: Test specification shall be prepared prior to testing.

6.3.2 Information Perspective

In general, a task requires data as input and produces data as output. The input data may come from the output of other tasks. Similarly, the output data may become the input of other tasks. The information might be moved or transformed. For example, software safety validation requires a software validation plan from another task as an input data, and will produce a software safety validation report after the task has completed successfully. In regard to the recommended processes of IEC61508, the input data and output data of a task are recorded and presented in a document form. As task can be decomposed to sub-tasks, the input data and output data will also be decomposed to granulated pieces each of which will be assigned to relevant sub-tasks. For example, software validation can be decomposed into sub-tasks where each output data is a sub-section of the software safety validation report. A completed report can be obtained when all sub-tasks have been completed successfully. Following are the key requirements of applying IEC61508 concerned by information perspective:

- Requirement 6: Document shall be produced.
- Requirement 8: The specification of safety requirements shall be documented.
- Requirement 13: The results of test and analysis activities shall be recorded.

6.3.3 Organisational Perspective

The organisational perspective of a process is concerned with who is responsible for performing a task. As soon as a task is chosen to be executed through the behavioural perspective, it has to be decided which agent is responsible for performing this task. This perspective uses the concept of agent selection polices, such as (Kappel et al., 2000), (Rupietta, 1997) and (Moore et al., 2000). There are mainly four kinds of agent selection polices: (1) agent related selection – select an agent by means of its identifier, (2) role related selection – select an agent playing a certain role from an organisation structure, (3) workflow related selection – select an agent based on various data about actual and previous workflows, and (4) capability matching – select an agent base on the satisfaction of its capability against the required capability for performing the task. Following are the key requirements of applying IEC61508 concerned by the organisational perspective:

- Requirement 3: The project organisation and allocated responsibilities shall be defined.
- Requirement 14: The development shall be subjected to independent safety validation and assessment.

6.3.4 Technological Perspective

The three perspectives from traditional process modelling are not complete. There are further important perspectives in the context of process modelling. The technological perspective is an example which defines how a task is performed. The technology perspective captures the IEC61508 recommended techniques which have to be used to perform identified tasks. Following are the key requirements of applying IEC61508 concerned by this perspective:

- Requirement 2: Suitable techniques and measures shall be selected.
- Requirement 11: Suitable design techniques shall be employed depending on the required Safety Integrity Level.

Out of the fourteen key requirements only requirement 9, which emphasises the importance of tractability of the design process, is not covered by the perspectives discussed. Tractability is concerned with the follow-up of tasks execution which belongs to the scope of workflow engine. The tractability issue has clearly been discussed in Chapter 5.

6.4 Model of Standards

The Model of Standards is a process model which acts like a knowledge database providing information about a standard in terms of process management. In Compliance Flow, such information will be used as a check spelling process in which a number of matching mechanisms will be performed to assess the degree of compliance of a user-defined process with a standard. The standard modelling is based on the four perspectives discussed above, with capability of modelling a wide range of standards. The approach has successfully modelled IEC61508 with its two important concepts: the Safety Lifecycle and Safety Integrity Levels (SIL). Safety Lifecycle is the proposed development process necessary to achieve the required SIL. SIL, a number between 1 and 4 is an indicator for specifying the safety integrity requirements with 4 having the highest level of safety integrity. For different level SIL, the detail of development process may vary. The meta-model of standard modelling presented using Object Modelling Technique (OMT) is illustrated in Figure 6-2.

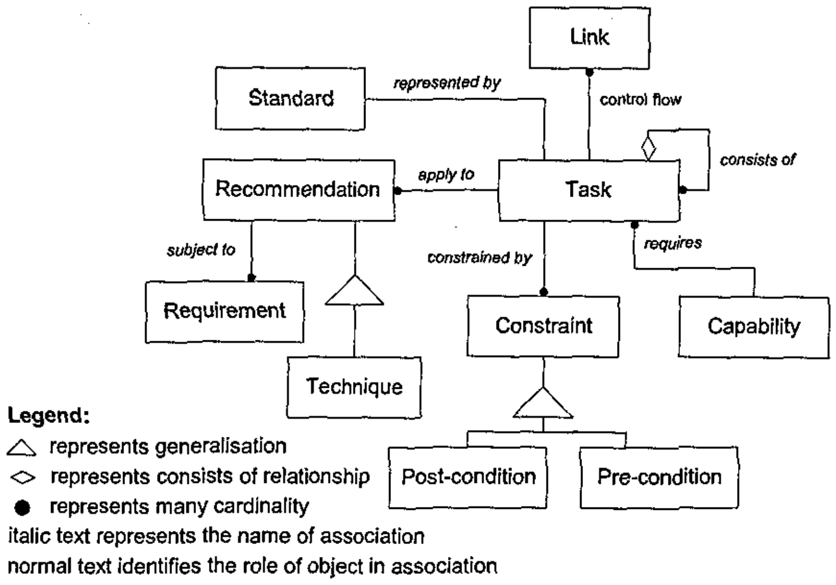

Link
Standard
represented by
control flow
consists of
Recommendation
apply to
Task
subject to
constrained by
requires
Requirement
Constraint
Capability
Technique
Post-condition
Pre-condition
Legend:
△ represents generalisation
◇ represents consists of relationship
● represents many cardinality
italic text represents the name of association
normal text identifies the role of object in association

Figure 6-2. Meta-model of standard modelling.

6.4.1 The Use of Ontology

As the information provided by the Model of Standard is used in compliance checks by matching it with the user-defined processes, the terms used in describing a concept of interest existing in both side must be the same. Compliance Flow takes the advantage of ontology to enable a matching process. All terms used to describe the concepts in the context of process management such as task, pre-conditions and post-conditions have to be selected from the ontologies. The terms of an ontology is organised into a hierarchy in which a term located in a higher level implies a higher level of abstraction, while a lower level term represents a more concrete concept of object. A set of ontologies comes with the system which, however, can be changed, removed or extended by users to adapt to the particular environment where the system is running.

6.4.2 Modelling of Task Framework

The task framework in the Safety Lifecycle proposed by IEC61508 is modelled into a hierarchical task network (HTN). A task is a basic unit of work, which can be hierarchically decomposed into sub-tasks if necessary until the required details are modelled, as long as the parent-child relationship between tasks are maintained.

Each task is associated with two sets of conditions: pre-conditions and post-conditions. The post-conditions of a task are sometimes the pre-conditions of its subsequent tasks. Performing a task requires the fulfilment of its pre-condition, and to do so, the preceding tasks that can satisfy those conditions with their post-conditions must be completed successfully in advance. The post-conditions of a task will be achieved when the task is completed successfully. Therefore the order of the execution of tasks is constrained by their dependencies.

IEC61508 views the requirements simply as the input to a distinct stage in the lifecycle, and the design specification as the output of that stage. The requirements and specifications are equivalent to the pre-conditions and the post-conditions respectively as they have to be achieved under the recommended sequence in order to comply with IEC61508. A condition is presented in the form of checklists, and it is stated as fulfilled when the checklist is checked.

6.4.3 Modelling of Task Recommendation

The recommended techniques, measures, tools or methods that have to be used for performing specific tasks to achieve the specified objectives are modelled with four parameters: (1) the task for which the technique is applied, (2) the requirements for applying the technique, (3) the technique itself, and (4) the level of recommendation. The value of second parameter can be null, implying that no requirement is necessary to apply the technique.

IEC61508 introduces sets of techniques for specific development activities with different levels of applicability according to the SIL of the product to be developed. The SIL is normally achieved after the safety requirements are addressed. Therefore, the level of SIL (from 1 to 4) becomes the requirement for applying the recommended techniques. These techniques are categorised into four levels of recommendation in IEC61508 namely Highly Recommended (HR), Recommended (R), No Recommendation (-) and Not Recommended (NR).

These recommendations can be modelled, for example, IEC61508 recommends that the technique 'Structure Methodology' (parameter 3) is Highly Recommended (parameter 4) during the achievement of objective of Clause B.30 (parameter 1) when the SIL of the product being developed is equal to 1 (parameter 2).

6.4.4 Modelling of Task Capability

A task capability refers to the required capabilities of an agent who is going to perform a specific task. The modelling of task capability adopts the same approach as the modelling of agent capability. A capability consists of two parts: the technical capability and its application area. Two sets of ontologies are used in modelling of a task capability, namely capability ontology and application ontology. While the capability ontology defines the required techniques, skills, roles and other attributes of an expected agent, the application ontology defines their application areas. The terms in an ontology are organised into a hierarchical structure which provides a hierarchy of capabilities. This hierarchical structure can ease the process of specifying capabilities since specifying a high-level capability implies that all its lower-levels are covered.

A single capability is denoted as Capability(t_c,t_a) where t_c is the term drawn from represents the capability and t_a represents the application area of t_c. The capability part and the application part may have different weight in representing a capability, and therefore two weighting points are required: w_c for the percentage of weights of t_c and w_a for the percentage of weight of t_a. The sum of w_c and w_a must equal 100. On the other hand, performing a task may require its agent possessing multiple capabilities that forms a capability set, denoted as Capability(*p1,p2...pn*). Each capability in the capability set may have different weight. A weight, denoted as w_s, is therefore required for each capability in the capability set. Similarly, the sum of w_s of all capabilities in a capability set must equal to 100. Therefore, a full description of a capability is denoted as Capability(t_c,t_a,w_c,w_a,w_s) where w_c, w_a and w_s are optional parameters. The details of capability representation and capability matching can please refer to Chapter 8, Design and Implementation.

The task capability is normally implicitly specified in most of the industry standards. A standard can be applied to wide areas where the situations are different and therefore agents with different capability will involve in performing similar tasks but in different projects or in different companies. Standards normally give the guidelines for agent selection rather than the details of required capability. For example, IEC61508 gives a number of competence factors which should be addressed when assessing and justifying the competence of persons to carry out their duties, including:

> *"(1) Engineering appropriate to the area; (2) Engineering appropriate to the technology; (3) Safety engineering appropriate to the technology; (4) Knowledge of the legal and safety regulatory framework; (5) The consequences in the event of failure of the safety-related systems. The greater the consequences the more rigorous shall the specification and assessment of competence be; ..." (IEC, 1997)*

Thus, the task capabilities of a standard have to be defined by users according to their understanding to these factors and the particular environment where the system is running. The factors such as the nature of the product, the country where product will be consumed, and organisational culture can affect the capability specification.

The required capability for the same process in different projects may vary because of different situations. For example, different techniques may require for different levels of SIL. The group of required capabilities together with the suitable weights for a particular situation can be saved as a scheme. A task is possible for assigning with multi schemes. As a result, users can choice suitable scheme of capability during the runtime.

6.5 Chapter Summary

This chapter described the standard modelling approach in Compliance Flow, using IEC61508 international standard as an example to identify the required requirements which have to be fulfilled for the application. The Model of Standard is a process model which provides information of a standard in terms of process management to be used. Such information will be used for compliance checks. To do so, such requirements have to be concerned by the standard model. The standard model is developed based on a number of process perspectives. The traditional perspectives of the process modelling are insufficient in modelling a standard. A further technical perspective is required.

The Model of Standards is capable of capturing four important aspects of a standard in terms of workflow management: (1) the proposed task framework, (2) the requirements and the deliverables of every task in the framework, (3) the required techniques, measures, tools or methods to perform a task, and (4) the required capability of task agents. As the approach of standard modelling has succeeded in modelling IEC61508, it is capable of modelling a wide range of similar standards.

Chapter 7

Managing Process Compliance Based on Process Model Reasoning

"Everything should be made as simple as possible, but not simpler."

- Albert Einstein

7.1 Introduction

The current best practice of providing reliable systems is to embody the development process in recent industry standards and guidelines, such as IEC61508 for safety and ISO9001 for quality assurance. These standards are generic, but every application is different because of the differences in project details. Once a standard has been adopted, it is important to manage project compliance according to the standard. However, current WfMS lack the ability to ensure that a process is planned and performed such that it complies with the selected standard.

This chapter presents Compliance Flow, focusing on the compliance check mechanisms. It is organised as follow: §7.2 gives a definition to process compliance. §7.3 introduces the Compliance Flow's approach, describing the compliance check mechanisms that assist users to specify a compliant process and ensure its execution is in accordance with the standards. Examples drawing on a draft version of IEC61508 standard are used to illustrate the checking mechanisms. §7.4 discusses the coupling issue between the compliance agent and the Model of standards. A chapter summary is given in §7.5.

7.2 Compliance with IEC61508

IEC61508 specifies a safety lifecycle as a guide for product development, which deals in a systematic manner all the necessary activities to achieve a required safety integrity level. As every project has its unique process details, it is therefore impracticable to impose a canonical safety lifecycle on all projects.

The following are three quotes from a draft version of IEC61508 standard:

> *"...a different overall safety lifecycle can be used... provided the objectives and requirements of each clause of standard are met."*
>
> *"The E/E/PES Safety Lifecycle that should be used...If another E/E/EPS Safety Lifecycle is used it shall be define in the Safety Plan and all the Objectives and Requirements of each Clause of this International Standard shall be met."*
>
> *"...the documentation requirements in standard are concerned, essentially, with information rather than physical documents."*

Thus, a safety system development process is IEC61508 compliant:

> *There is a clear description of the design stages and, at each stage, the inputs to that stage are fully and unambiguously defined, and also all the objectives and requirements of the standard are met.*

7.3 Compliance Checks

In Compliance Flow, a standard that is required to be complied by a development process is represented as a Model of Standards (hereinafter referred as the Model). The Model acts as a knowledge base to provide the required information to support compliance checks. It captures four important aspects of a standard in terms of workflow management:

1. The task framework of the standard development lifecycle that is used to deal with all the necessary activities to achieve the acceptable quality of product or services.
2. The requirements and the deliverables of every task in the framework.
3. The techniques, measures, tools or methods that are recommended to be used to achieve specific objectives or requirements.
4. The required capability of task agents. Capability refers to qualifications, roles, experiences or other attributes identified by a standard, for which a performer must possess in order to be qualified to carry out a specific type of task.

Compliance checks are for checking a user-defined process against the Model to identify compliance errors and assists users in specifying a process that meets the requirements of a selected standard. Compliance checks can be grouped into two categories.

Error identification and prevention:

- Correctness Check – to ensure the sequence of tasks specified in a user-defined process is in accordance with a selected standard.
- Completeness Check – to ensure all the required tasks within a standard are defined in the user-defined process.
- Capability Check – to ensure the required capabilities of an agent are specified according to a selected standard.
- Recommendation Check – to ensure the recommended techniques, measures, tools or methods for performing a particular task are fully considered.

Process management assistance:

- Planning Assistant – to remind the user of possible pre-conditions, post-conditions, recommendations, and required capability of a particular task while it is being planned.
- Information Navigation – to transfer the required information, once it is available, to the tasks where the information may be used for their execution.
- Cross-Referencing – to refer a user-defined task to the relevant part of a standard or vice

versa.

Process planning normally starts at an abstract level and becomes more concrete as planning continues. Compliance checks can be applied at any stage during process planning. The checks can also be applied to a task at any level of a HTN. If the selected task in the user-defined process is a high level task, then its child-tasks will be checked. A high level task is standard compliant if all of its child-tasks passed the compliance checks. The levels of abstraction between the user-defined process and the Model may vary. There are three possible situations:

1. A user-defined process (UDP) is at the same level of abstraction as the Model.
2. A user-defined process is at a higher level of abstraction than the Model.
3. A user-defined process is at a more detailed level than the Model.

Compliance agent will perform checks on a process or a task in response to a request from the user at build-time, but it will automatically perform them before the execution of a task at run-time. The behaviors of a compliance check during build-time and run-time may vary. In the following section, compliance checks are described based on the three situations identified above and the different behaviors during build-time and run-time are identified and discussed.

7.3.1 Definition of Terms

The basic concept of a compliance check is to perform a variety of mappings between a user-defined process and the Model of Standards to identify the compliance errors. The following terms are frequently used in the description in the rest of this thesis.

User-defined Task

A *user-defined task* is a task specified in a user-defined process. For figures in the rest of this chapter, a user-defined task is presented using a rounded rectangle with an identifier beginning with the letter 'T'. In Figure 7-1, T1 is an example of a user-defined task.

Standard Task

A *standard task* is a task contained in the Model. A standard task is presented using a rounded rectangle with an identifier beginning with the letters 'ST'. In Figure 7-1, ST1 is an example of a standard task.

Task Specification, User-defined Specification, and Standard Specification

IEC61508 views requirements simply as input to tasks in an engineering process, and specifications as output of tasks. An output specification from one task can be an input requirement of another task in the same process. The term 'specification' refers to a design specification which is modelled as a post-condition of a task. A specification is normally described in a document so it is represented using a document icon.

A *task specification* is a kind of specification that relates to a standard and is included in a user-defined process. A *user-defined specification* is another kind of specification defined by a user but its context is outside the scope of a standard. A specification that is included in the Model is called a *standard specification.*

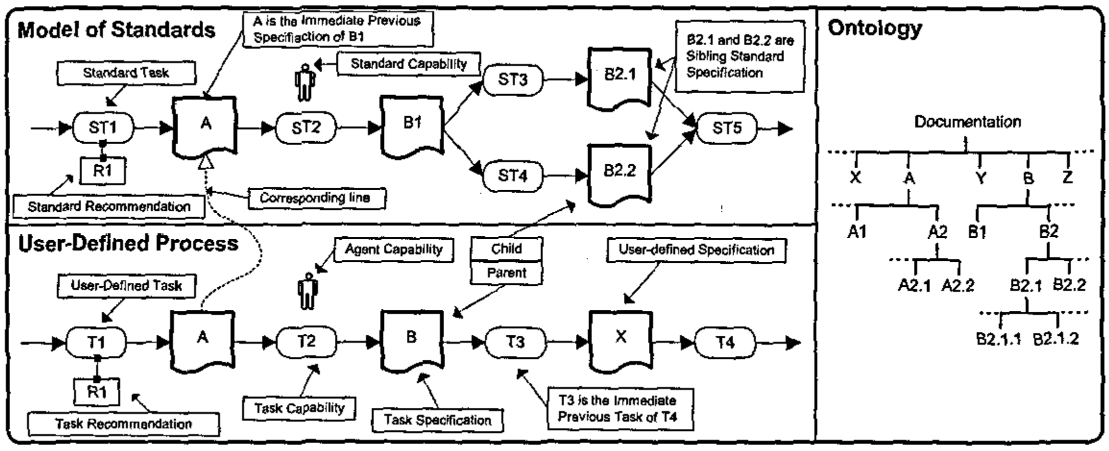


Figure 7-1. Some terms used in the description of compliance check mechanisms.

Corresponding Specification and Corresponding Task

A task specification corresponds to a standard specification if they have the same concern and the two corresponding specifications will have the same name because of the use of ontology. In Figure 7-1, the standard specification A in the Model is the *corresponding standard specification* of the task specification A in the user-defined process. A *corresponding line* is used to represent the corresponding relationship between two specifications. Its empty arrow indicates the matching direction in a particular example.

Two tasks that produce corresponding output specifications are corresponding tasks, but the tasks can have different names. For example, ST1 is the *corresponding standard task* of T1. Therefore, all the specified constraints at a standard task, including pre- and post-conditions, capability, recommendations and information must be applied to its corresponding task in the user-defined process in order to comply with the standard.

Corresponding Parent and Child Relationship

A specification may partially correspond to another specification. This relationship is captured by the parent-child relationship between two specification terms in the Documentation Ontology. For example, in Figure 7-1, B1 is a child-term of B in the ontology hierarchy, this means that the standard specification B1 corresponds to part of the task specification B. This relationship is described in the following sections as: B1 is a *corresponding child-specification* of B, or B is a *corresponding parent-specification* of B1.

There are two possibilities: (1) a standard task corresponds to multiple tasks in a user-defined process, and (2) a user-defined task corresponds to multiple tasks in the Model. For the first situation, all the constraints specified with the standard task must be applied to its corresponding tasks in a user-defined process. Similarly, for the second situation, the constraints associated with the standard tasks must be applied to the corresponding task in the user-defined process. However, some cases will override this general rule. They will be discussed in particular compliance checks.

Sibling Specification and Sibling Task

If two specifications have the same parent, then they are *sibling specifications.* In Figure 7-1, B2.1 is a sibling standard specification of B2.2, and ST3 is the *sibling standard task* of ST4, and vice versa.

Immediate Previous Specification and Immediate Previous Task

Specifications located in front of another specification in a process are called its *previous specifications.* An *immediate previous specification* is one that is immediately before another specification. In Figure 7-1, A is the *immediate previous task specification* of B. Similarly, ST2 is the *immediate previous standard task* of ST3 and ST4.

Agent Capability

An *agent capability* is a capability possessed by a task agent.

Task Capability

A *task capability* is a capability required to perform a particular task as specified by the process planner.

Standard Capability

A *standard capability* is a capability specified in the Model that a task agent has to possess in order to carry out a particular task.

Standard Recommendation

A *standard recommendation* is a recommendation defined in the Model.

Task Recommendation

A *task recommendation* is a recommendation defined in a user-defined process.

7.3.2 Identifying the Corresponding Specifications and Tasks

A common but important function in all the compliance checks is to identify the corresponding specifications and tasks in the Model in order to retrieve the required information for checking.

In the figures that follow, the rounded rectangles with identifiers beginning with the letters 'ST' represent standard tasks and identifiers that begin with letter 'T' represent user-defined tasks. The document icons represent the pre- and post-conditions of tasks. The dotted lines represent the corresponding relationships between standard specifications and task specifications. The arrows of the dotted lines represent the mapping directions of identifying the corresponding specifications and tasks in the examples.

If UDP and the Model are at the same level of abstraction, the task specification and the corresponding standard specification can be matched directly as they have the same name.

If the corresponding standard specification cannot be matched directly, it could be because UDP and the Model are at different levels of abstraction. The system will first establish whether UDP is more detailed than the Model. To do so, its parent-terms are matched with the Model, starting from the immediate parent. As an example using Figure 7-2, to identify the corresponding standard specification of B2.1.1, its parent-terms are matched with the Model in the order B2.1, B2 then B. In this example, B2.1 can be located.

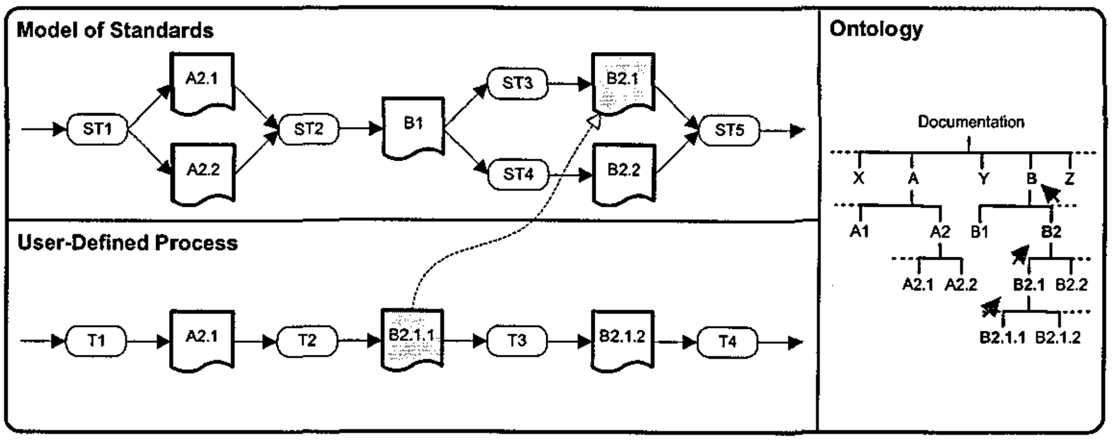


Figure 7-2. An example of UDP is at a more detailed level than the Model.

If no parent-terms can be matched, then its child-terms will be tried. As an example, in Figure 7-3 three of the child-terms of B (B1, B2.1 and B2.2) matched the Model. They are considered the corresponding child-specifications of B.

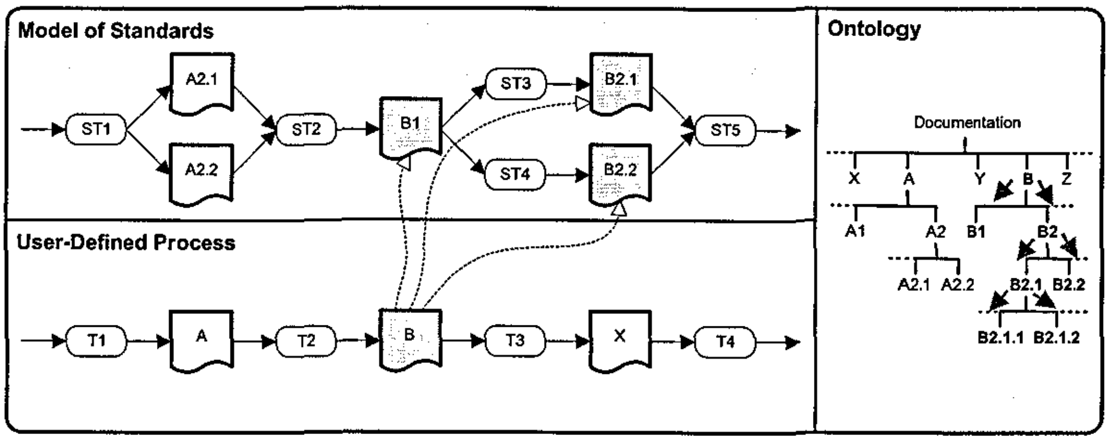


Figure 7-3. An example of UDP is at a higher level of abstraction than the Model.

If a specification or any of its parent and child-terms cannot find a match in the Model, then it is a user-defined specification, and it is not a concern of the compliance checks. An example is the specification X in Figure 7-3

7.3.3 Correctness Check

The Correctness Check is to verify that the placement of a task specification complies with the model, and therefore ensure that the execution sequence of the tasks in the user-defined process is correct.

To verify that a task specification is placed correctly, first is to identify its corresponding standard specification in the Model. Second is to ensure its immediate previous specification is also located in front of the corresponding standard specification. If this is the case then the task specification is placed correctly. If the immediate previous specification is a user-defined specification, then the next previous specification will be matched instead.

As an example, consider Figure 7-4, to check the compliance of the task specification 'Software Safety Validation Report', first is to identify its corresponding standard specification in the Model. Its previous specification is then matched against 'Software Safety Validation Plan' in the Model. The matching of X failed as it is a user-defined specification. The previous specification of X, 'Software Safety Validation Plan', is then tried. It is found in the Model and is placed in front of the corresponding standard specification. Therefore, it can be claimed that the specification 'Software Safety Validation Report' is placed in the right order.

If a user-defined process is at a higher level of abstraction than the Model, then the specifications cannot be matched directly. The identification of two corresponding specifications at different levels of abstraction is described in section 7.3.2.

When applying Correctness Check to a task, all the specifications of the task will be checked. A task is compliant if all of its specifications passed the Correctness Check. Similarly, a process is compliant when all of its tasks are compliant.

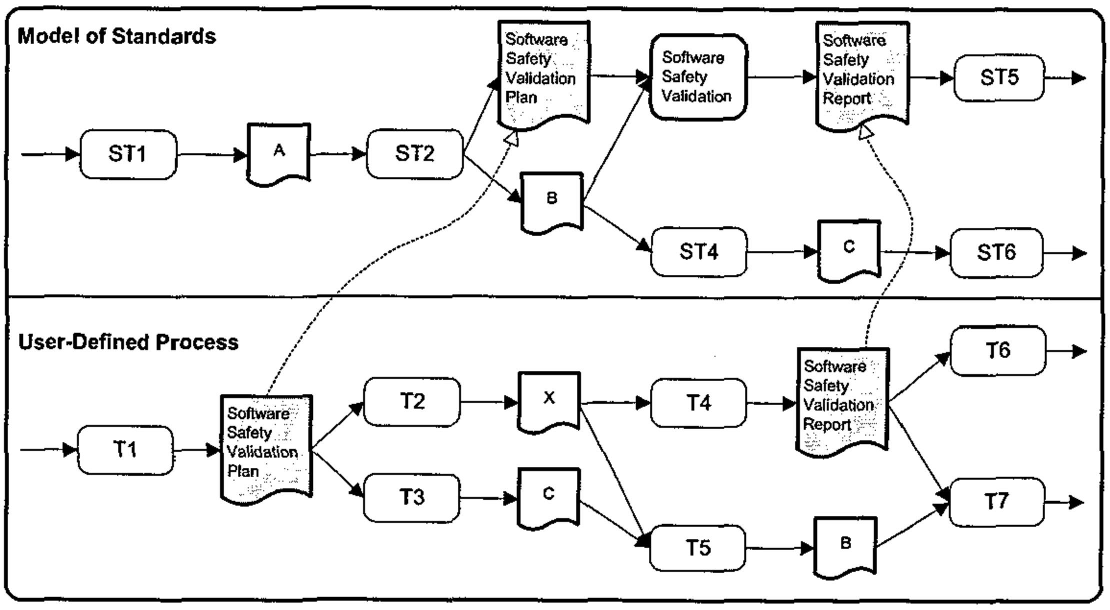

Model of Standards
ST1
A
ST2
Software Safety Validation Plan
B
Software Safety Validation
Software Safety Validation Report
ST5
ST4
C
ST6
User-Defined Process
T1
Software Safety Validation Plan
T2
X
T3
C
T4
T5
Software Safety Validation Report
B
T6
T7

Figure 7-4. Correctness Check when UDP and the Model are at the same level of abstraction.

The Correctness Check algorithm is given in Figure 7-5.

```
Function CorrectnessCheck(spec, Std) as Boolean
      spec: A specification in a user-defined process for which
            the correctness check is applied to.
      Std: The Model of the selected standard.
      ImSpec{…} = {x:x ∈ Standard Specification};
      ImSpec = GetImmediatePreviousStandardSpecification(spec,Std);
      If ImSpec ≠ 0 Then
            PvTSpec{…} = {x:x ∈ Task Specification};
            PvTSpec = GetPreviousTaskSpecifications(spec);
            If (ImSpec ⊆ PvTSpec) Then
                  Return True;
                  Exit Function;
            End If
            PImSpec{…} = {x:x ∈ Standard Specification};
            For each i ∈ ImSpec
                  PImSpec = GetParent(i);
                  If ((PImSpec ∩ PvTSpec) ≠ 0) Then
                        ImSpec = ImSpec - {i}
                  End If
            Next i
            CImSpec{…} = {x:x ∈ Standard Specification};
            For each i ∈ ImSpec
                  CImSpec = GetChildren(i);
                  If ((CImSpec ∩ PvTSpec) ≠ 0) Then
                        ImSpec = ImSpec - {i}
                  End If
            Next i
            If ImSpec ≠ 0 Then
                  Return False
            Else
                  Return True;
            Endif
      Else
            Return True;
      End If
End Function
```

Figure 7-5. Correctness Check algorithm.

7.3.4 Completeness Check

Completeness Check is to ensure that the required specifications are included in UDP and therefore ensures that the required tasks are also included. It can be applied to the whole process or a sub-process at build-time as long as the corresponding part in the Model is given. For example, to check the completeness of UDP that is related to the part of software development in IEC61508, the corresponding sub-process entitled 'Safety Software Design and Development' in the Model must be identified. Selecting an inappropriate corresponding standard sub-process will lead to a long list of missing specifications or some of the required specifications are not checked.

In Completeness Check, all the standard specifications in the given standard process are matched against the task specifications in UDP. If all the specifications are found, then the Completeness Check succeeds. In Figure 7-6, the Completeness Check fails because specification C is not in the user-defined process.

To pass Completeness Check during run-time, the required specifications to perform the user-defined task must be ready. The required specifications can be retrieved from the corresponding task in the Model. The user does not need to identify a corresponding part in the Model.

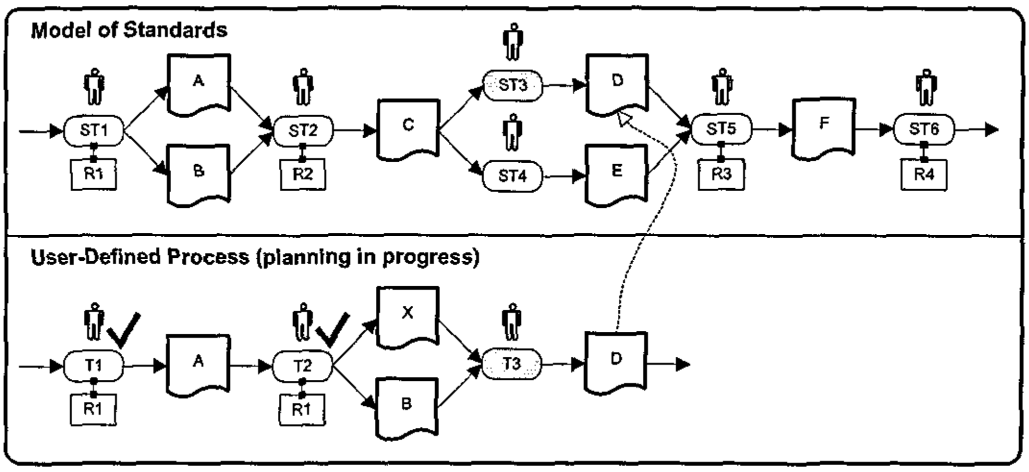


Figure 7-6. An example of applying Completeness Check to a task.

As an example using Figure 7-6, the user-defined tasks T1 and T2 are completed and the required specifications X and B for T3 are ready. ST3 is the corresponding task of T3 as they have the same post-condition. From the Model, it is known that a specification C is required for T3 to produce a specification D but it is not specified as a pre-condition of T3. Therefore, the Completeness Check applied to T3 fails.

The Completeness Check algorithm used at process build-time and run-time are given in Figure 7-7 and Figure 7-8 respectively.

```
Function CompletenessCheck(Pros, SPros) as Boolean
      Pros: A user-defined process consisted of a set of tasks
            for which the completeness check is applied to.
      SPros: A corresponding process in the Model of a
            selected standard.
      TaskSpec{…} = {x:x ∈ Task Specification};
      StdSpec{…} = {x:x ∈ Standard Specification};
      TaskSPec = GetSpecifications(Pros);
      StdSpec = GetSpecifications(SPros);
      For each i ∈ StdSpec
            If (i ∈ TaskSpec) Then
                  StdSpec = StdSpec - {i};
            Else
                  PTaskSpec{…} = {x:x ∈ Task Specification};
                  PTaskSpec = GetParent(i);
                  If ((PTaskSpec ∩ StdSpec) ≠ 0) Then
                        StdSpec = StdSpec - {i};
                  Else
                        CTaskSpec{…} = {x:x ∈ Task Specification};
                        CTaskSpec = GetChildren(i);
                        If ((CTaskSpec ∩ StdSpec) ≠ 0) Then
                              StdSpec = StdSpec - {i};
                        Else
                              Return False;
                        End If
                  End If
            End If
      Next i
      If (StdSpec ≠ 0) Then
            Return False;
      Else
            Return True;
      End If
End Function
```

Figure 7-7. Completeness Check algorithm used at process build-time.

```
Function CompletenessCheck(task, SPros) as boolean
      task: A user-defined task which user is trying to start.
      SPros: A corresponding process in the Model of a
            selected standard.
      PvTaskSpec{…} = {x:x ∈ Task Specification};
      PvTaskSPec = GetPreviousSpecifications(task);
      TaskSpec{…} = {x:x ∈ Task Specification};
      TaskSpec = GetSpecifications(task)
      CorStdSpec{…} = {x:x ∈ Standard Specification};
      For each i in TaskSpec
            CorStdSpec = CorStdSpec ∪ _
            GetCorresdpondingStdSpecifications(i,SPros);
      Next i
      StdTask = {x:x ∈ Standard Task};
      For each i in CorStdSpec
            StdTask = StdTask ∪ GetAssociatedTask(i);
      Next i
      StdSpec = {x:x ∈ Standard Specification};
      For each i in StdTask
            StdSpec = StdSpec ∪ GetPreviousSpecification(i);
      Next i
      For each i in StdSpec
            If (i ∈ PvTaskSpec) Then
                  StdSpec = StdSpec - {i};
            Else
                  ParentTaskSpec{…} = {x:x ∈ Task Specification};
                  ParentTaskSpec = GetParent(i);
                  If ((ParentTaskSpec ∩ StdSpec) ≠ 0) Then
                        StdSpec = StdSpec - {i};
                  Else
                        ChildTaskSpec{…} = {x:x ∈ Task Specification};
                        ChildTaskSpec = GetChildren(i);
                        If ((ChildTaskSpec ∩ StdSpec) ≠ 0) Then
                              StdSpec = StdSpec - {i};
                        Else
                              Return False;
                        End If
                  End If
            End If
      Next i
      If (StdSpec ≠ 0) Then
            Return False;
      Else
            Return True;
      End If
End Function
```

Figure 7-8. Completeness Check algorithm used at process run-time.

7.3.5 Capability Check

Capability Check is to ensure that a task agent possesses the required capability specified in a standard for performing a task. Capability Check returns a goodness of fit (GOF) value indicating how well an agent matches the required capabilities. The GOF is assessed from three different views: (1) the GOF of task capability against the standard capability requirement; (2) the GOF of agent capability against task capability requirement; (3) the GOF of agent capability against standard capability requirement, as shown in Figure 7-9. Each GOF is assessed through a fuzzy capability matching algorithm which requires two sets of capabilities: the required capability and the available capability. Using the first view as an example, the standard capability specified in the Model is the required capability and the task capability specified by the process planner is the available capability. Details of the matching algorithm are discussed in Chapter 8.

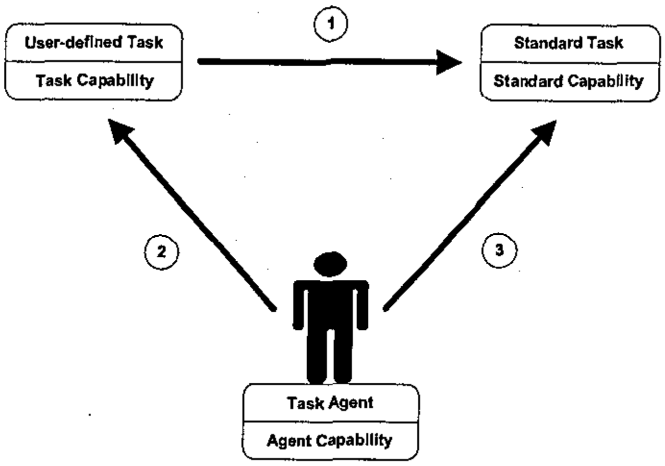


Figure 7-9. Three capability views are given in the Capability Check.

7.3.5.1 UDP and the Model at the Same Level of Abstraction

If a user-defined task is at the same level of abstraction as its corresponding standard task then the standard capability for the corresponding standard task is the required capability for the user-defined task.

As an example using Figure 7-10, Capability Check is applied to the user-defined task T4. The required capability can be retrieved from its corresponding standard task, 'Software Safety Validation', in the Model. The agent capability is retrieved from the Organisation Server. These two sets of capability are sent to the fuzzy capability matching algorithm to assess the GOF values.

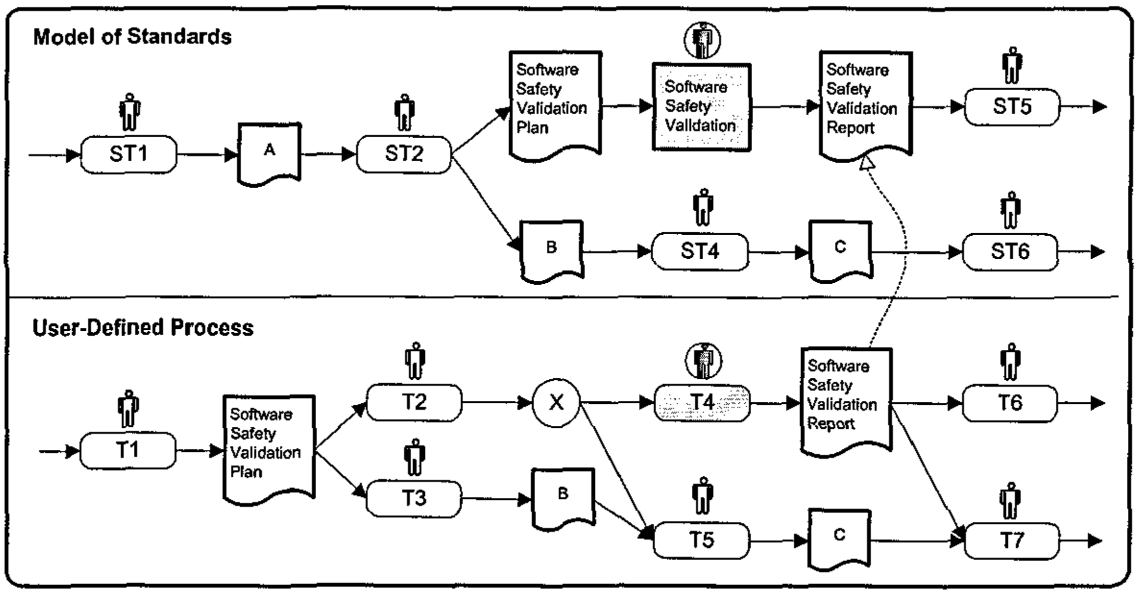

Model of Standards
ST1
A
ST2
Software Safety Validation Plan
Software Safety Validation
Software Safety Validation Report
ST5
B
ST4
C
ST6
User-Defined Process
T1
Software Safety Validation Plan
T2
T3
X
T4
Software Safety Validation Report
T6
B
T5
C
T7

Figure 7-10. Capability Check when UDP and the Model are at the Same Level of Abstraction.

7.3.5.2 UDP at a Higher Level of Abstraction than the Model

If a user-defined task is at a higher level of abstraction than the Model, then it may correspond to multiple standard tasks. The collection of the capabilities of these standard tasks will be treated as the required capabilities for the user-defined task.

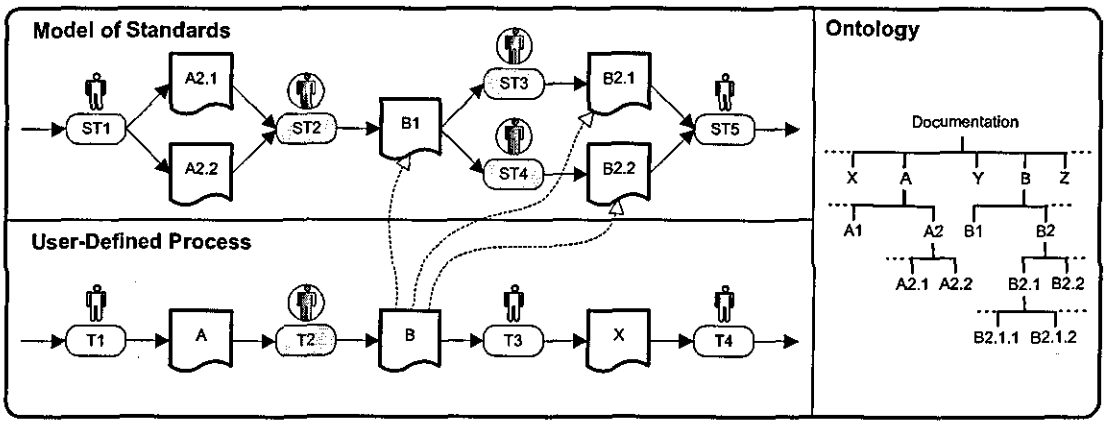


Figure 7-11. Capability Check when UDP at a higher level of abstraction than the Model.

In Figure 7-11, Capability Check is applied to the user-defined task T2 which has three corresponding standard tasks, ST2, ST3 and ST4, in the Model. The specified capabilities of these tasks are the required capabilities for T2.

If UDP is at a higher level of abstraction than the Model, then Capability Check is advisory at build-time but is enforced at run-time. At build-time, Capability Check can be applied to a task at any level of abstraction. A high level task may be assigned to a management staff who is not going to perform the sub-tasks. The sub-tasks will be specified in detail and assigned to suitable technical staffs to execute. Consequently, the manager does not need to fulfil the required technical capability specified in the Model. As process planning starts at an abstract level, compliance agent is unable to determine whether a task is to be executed or further decomposed. Therefore, the result from Capability Check is only advisory.

However, when Capability Check is applied to a task to be performed at run-time, a task should not be allowed to proceed if it fails Capability Check.

7.3.5.3 UDP at a More Detailed Level of Abstraction than the Model

If a user-defined task is at a more detailed level of abstraction than the Model, then it is only concerned with part of the corresponding standard task. Therefore, the standard capability in the Model may exceed the actual requirement for the user-defined task. Because there is no way of filtering out the exceeding part, the whole standard capability will be considered as the required capability.

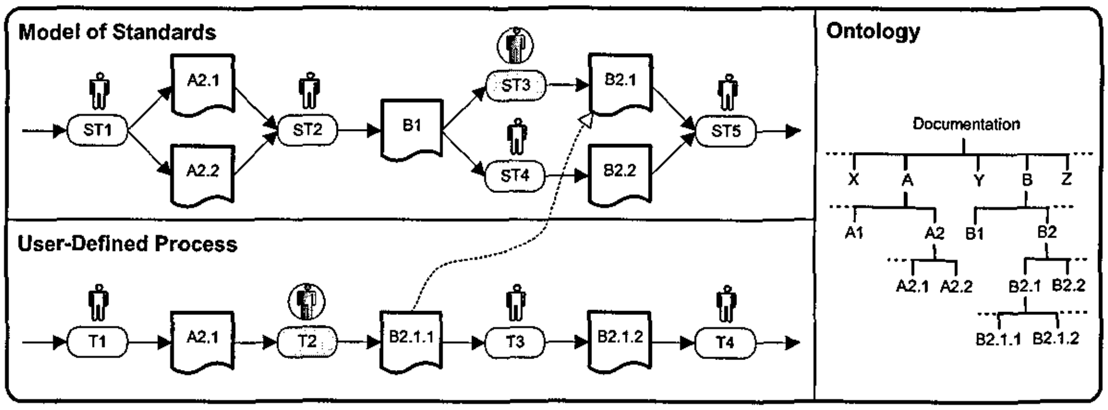


Figure 7-12. Capability Check when UDP at a more detailed level of abstraction than the Model.

As an example using Figure 7-12, Capability Check is applied to a user-defined task T2 where the corresponding parent-standard task ST3 is identified in the Model. Capability Check assumes that the standard capability specified for ST3 is required for T2 even though it may exceed the actual requirement. Therefore, the standard capability is used as the required capability for capability matching.

Capability Check can also be applied to a process at build time. To do so, compliance agent will perform Capability Check to every task in the process. The process will pass only when all its tasks passed.

The Capability Check algorithm for retrieving the task capability of a given user-defined task and its required standard capability is given in Figure 7-13. These two sets of capability are passed to the capability matching function to access GOF.

```
Function CapabilityCheck(task, Std) as Integer
        Task: A task in a user-defined process for which
                the capability check is applied to.
        Std: The Model of the selected standard.
        TaskCap{…} = {x:x ∈ Task Capability};
        TaskCap = GetCapability(task);
        TaskSpec{…} = {x:x ∈ Task Specifiation};
        TaskSpec = GetSpecifications(task);
        StdSpec{…} = (x:x ∈ Standard Specification);
        For each i ∈ TaskSpec
                StdSpec = StdSpec ∪ _
                {GetCorrespondingStdSpecification(i,Std)};
        Next i
        StdTask{…} = {x:x ∈ Standard Task}
        For each i ∈ StdSpec
                StdTask = StdTask ∪ {GetAssociatedTask(i)};
        Next i
        RequiredCap{…} = {x:x ∈ Standard Capability};
        For each i ∈ StdTask
                RequiredCap = RequiredCap ∪ {GetCapability(i)};
        Next i
        GOF = AssessGOF(TaskCap, RequiredCap);
        Return GOF
End Function
```

Figure 7-13. Capability Check Algorithm.

7.3.6 Recommendation Check

Recommendation Check is to ensure that the techniques, measures, tools or methods which are recommended by a standard for in performing particular tasks are fully considered by the user in planning a process.

IEC61508 recommends techniques and measures for the development of safety related systems for the control of failures. These recommendations are defined and related to standard tasks in the Model. On the other hand, a task recommendation is modelled as the pre-condition of a user-defined task in a user-defined process.

7.3.6.1 UDP and the Model at the Same Level of Abstraction

When carrying out the Recommendation Check (RC), the pre-condition of a user-defined task is matched against the standard recommendations retrieved from the corresponding standard task to verify whether the recommended techniques are selected. If any highly recommended (HR) technique is not used or a non-recommended technique is chosen, than an explanation is required and it will be recorded in the Tracking Server.

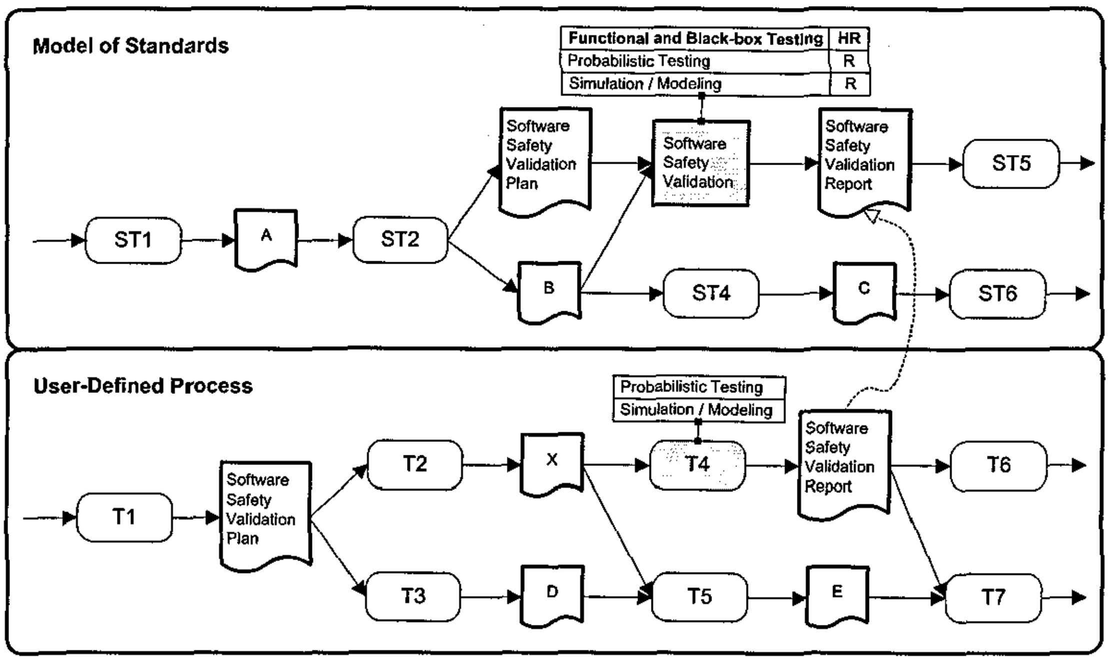

Model of Standards
Functional and Black-box Testing | HR
Probabilistic Testing | R
Simulation / Modeling | R
ST1
A
ST2
Software Safety Validation Plan
B
Software Safety Validation
ST4
Software Safety Validation Report
C
ST5
ST6
User-Defined Process
Probabilistic Testing
Simulation / Modeling
T1
Software Safety Validation Plan
T2
T3
X
D
T4
T5
Software Safety Validation Report
E
T6
T7

Figure 7-14. RC when UDP and the Model are at the same level of abstraction.

In Figure 7-14, the Recommendation Check is applied to a user-defined task T4 which is to prepare the 'Software Safety Validation Report' where two recommended (R) techniques are selected and defined as pre-conditions. Its corresponding standard task 'Software Safety Validation' has three recommended techniques one of which is a highly recommended (HR) technique 'Functional and Black-box Testing'. Because the highly recommended technique is not used in T4, the Recommendation Check fails. The check will only pass when either the reason for not using the technique is provided or the technique is added to T4 as a pre-condition.

7.3.6.2 UDP at a Higher Level of Abstraction than the Model

If a user-defined task corresponds to multiple standard tasks in the Model, then all the recommendations related to the corresponding tasks in the Model must be used or considered. If the user-defined task is to be further decomposed, then the selected recommendations will be kept with the child-tasks.

7.3.6.3 UDP at a More Detailed Level of Abstraction than the Model

When a standard task corresponds to multiple tasks in UDP, all its recommendations must be considered by the corresponding user-defined tasks.

The Recommendation Check algorithm is given in Figure 7-15.

```
Function RecommendationCheck(task, Std) as Boolean
      task: A task in a user-defined process for which the
            recommendation check is applied.
      Std: The Model of a selected standard.
      TaskSpec{...} = {x:x ∈ Task Specification};
      TaskSpec = GetSpecifications(task);
      StdSpec{...} = {x:x ∈ Standard Specification};
      For each i ∈ TaskSpec
            StdSpec = StdSpec ∪ _
            {GetCorrespondingStdSpecification(i,Std)};
      Next i
      StdTask{...} = {x:x ∈ Standard Task};
      For each i ∈ StdSpec
            StdTask = StdTask ∪ {GetAssociatedTask(i)};
      Next i
      Rcom{...} = {x:x ∈ Recommendation};
      For each i in StdTask
            Rcom = Rcom ∪ {GetRecommendations(i)};
      Next i
      PreCond(...) = {x:x ∈ Task Pre-Condition};
      PreCond = GetPreConditions(task);
      If (Rcom ⊆ PreCond) Then
            Return True;
      Else
            Return False;
      End If
End Function
```

Figure 7-15. Recommendation Check algorithm.

7.3.7 Planning Assistance

The Planning Assistance (PA) is to provide the user with the possible related information of the task being planned, including the possible pre-conditions, post-conditions, recommendations and the required capability of the task.

The concept of Planning Assistance is that if any pre- or post-condition of a user-defined task are already specified, then its corresponding standard task in the Model can be identified and the information related to the standard task may retrieved to assist task specification.

7.3.7.1 UDP and the Model at the Same Level of Abstraction

If a user-defined task is at the same level of abstraction as the Model, then the related information is retrieved from its corresponding standard task. In Figure 7-16, a user is specifying the user-defined task T3. Its corresponding standard task ST2 can be identified as specification C is given. Planning Assistance will then provide the information related to ST2 to the user for consideration. The possible information includes the pre-conditions, required capabilities and recommendations.

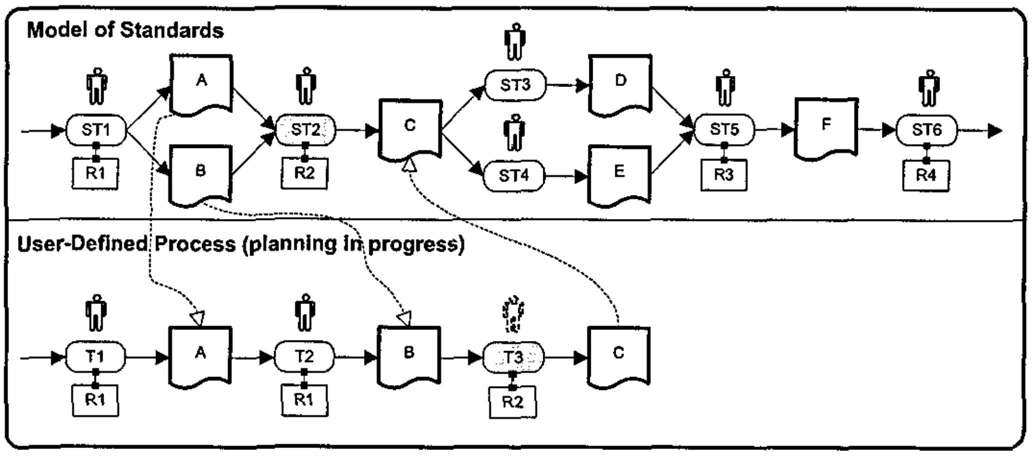


Figure 7-16. PA when UDP and the Model are at the same level of abstraction.

7.3.7.2 UDP at a Higher Level of Abstraction than the Model

If a user-defined task corresponds to more than one standard tasks in the model, all the information from the corresponding standard tasks will be treated as they are related to the user-defined task.

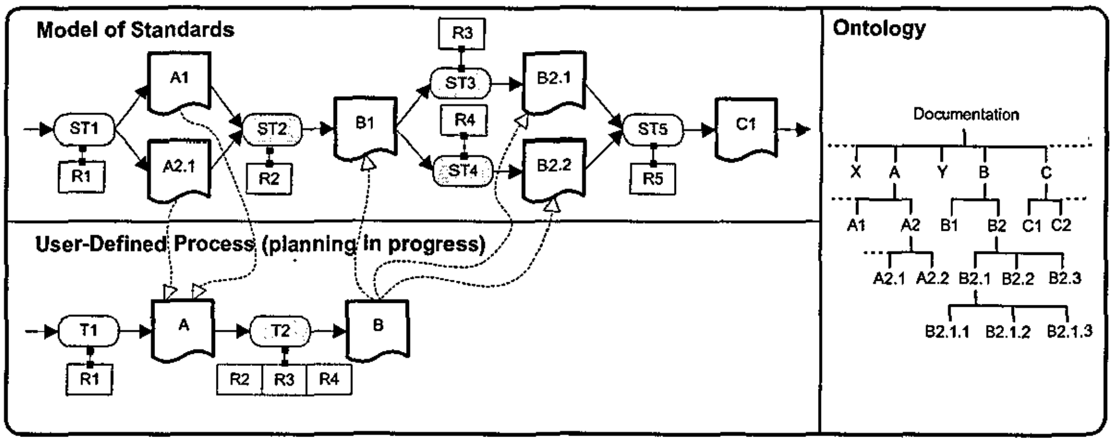


Figure 7-17. PA when UDP is at a higher level of abstraction than the Model.

As an example in Figure 7-17, a user is specifying the user-defined task T2 where a specification B is defined as the post-condition. Its corresponding standard tasks are ST2, ST3 and ST4 in the Model. Therefore, their recommendations, R2, R3 and R4 become the possible techniques for T2.

7.3.7.3 UDP at a More Detailed Level of Abstraction than the Model

If a standard task corresponds to multiple user-defined tasks in a user-defined process, then all the information from the standard task will be treated as they are required for all the corresponding user-defined tasks. An example is given in Figure 7-18.

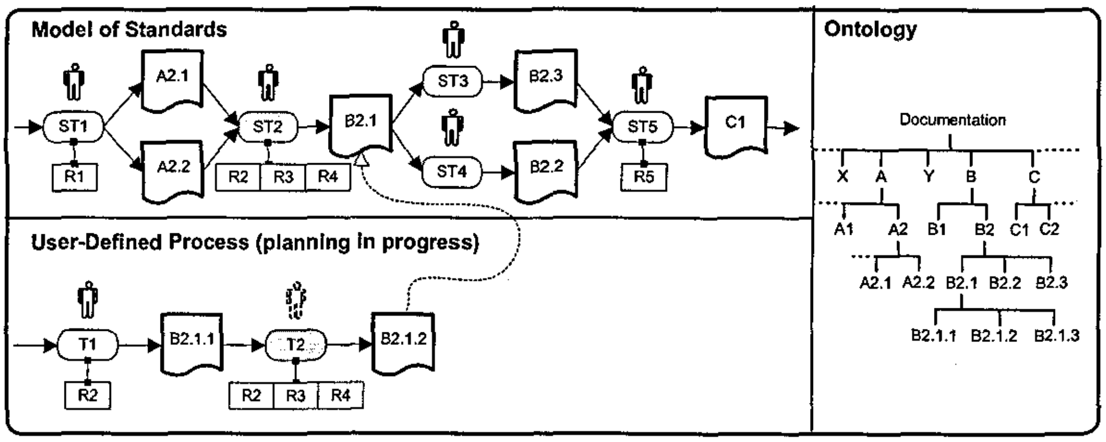


Figure 7-18. PA when UDP at a more detailed level of abstraction than the Model.

In Figure 7-18, a user is specifying the user-defined task T2 where the specification B2.1.2 is defined as a post-condition. Its corresponding standard task ST2, which is at higher level, is identified in the Model. It has three recommendations. Although R2 is already used in its sibling task T1, all of the recommendations are treated as possible techniques for T2.

The Planning Assistance algorithms are given in Figure 7-19 and Figure 7-20 respectively.

```
Function FindPossibleCapability(task, Std) as PossibleCap{…}
      Task: A task in a user-defined process for which
            the capability check is applied to.
      Std: The Model of the selected standard.
      TaskCap{…} = {x:x ∈ Task Capability};
      TaskCap = GetCapability(task);
      TaskSpec{…} = {x:x ∈ Task Specifiation};
      TaskSpec = GetSpecifications(task);
      StdSpec{…} = {x:x ∈ Standard Specification};
      For each i in TaskSpec
            StdSpec = StdSpec ∪ _
            {GetCorrespondingStdSpecification(i,Std)};
      Next i
      StdTask{…} = {x:x ∈ Standard Task}
      For each i in StdSpec
            StdTask = StdTask ∪ {AssociatedTask(i)};
      Next i
      RequiredCap{…} = {x:x ∈ Standard Capability};
      For each i in StdTask
            RequiredCap = RequiredCap ∪ {Capability(i)};
      Next i
      PossibleCap = TaskCap ⊄ RequiredCap;
End Function
```

Figure 7-19. The algorithm to find the possible capability for a user-defined task.

```
Function FindPossibleRecommendation(task, Std) as PossibleRcom{…}
      task: A task in a user-defined process for which the
            recommendation check is applied.
      Std: The Model of a selected standard.
      TaskSpec{...} = {x:x ∈ Task Specification};
      TaskSpec = SetSpecifications(task);
      StdSpec{...} = {x:x ∈ Standard Specification};
      For each i in TaskSpec
            StdSpec = StdSpec ∪ _
            {GetCorrespondingStdSpecification(i,Std)};
      Next i
      StdTask{...} = {x:x ∈ Standard Task};
      For each i in StdSpec
            StdTask = StdTask ∪ {GetAssociatedTask(i)};
      Next i
      Rcom{…} = {x:x ∈ Recommendation};
      For each i in StdTask
            Rcom = Rcom ∪ {GetRecommendations(i)};
      Next i
      PreCond(…) = {x:x ∈ Task Pre-Condition};
      PreCond = GetPreConditions(task);
      PossibleRcom = Rcom ⊄ PreCond;
      Return PossibleRcom;
End Function
```

Figure 7-20. The algorithm to find the possible recommendation for a user-defined task.

7.3.8 Information Navigation

The use of workspace enables information to be transferred from one task to another according to the task flow. However, the task flow does not guarantee that all the information has been considered.

Consider Figure 7-21, according to the Model, the specification 'Software Safety Validation Plan' is required for the task to produce a 'Software Safety Validation Report'. The user-defined process is compliant with the standard as both specifications exist and are placed in the correct sequence even though two user-defined tasks are in between. It is unknown whether T2 and T3 require the 'Software Safety Validation Plan' for their execution. However, it is certain that the plan is required for T4 as it is the actual task to produce 'Software Safety Validation Report' but it is not defined as a pre-condition because of a user mistake. Therefore the report will only exist in the workspaces of T1 and T2 when it is ready.

Information Navigation is to transfer the information, according to the Model, during process run-time to the related tasks where such information may be required for their execution. For a standard specification that is used as an input to a task to produce another standard specification (hereinafter refers to as the output specification), Information Navigation assumes that the input specification will be used for all the tasks in between these two specifications in UDP.

7.3.8.1 UDP and the Model at the Same Level of Abstraction

Once a document is uploaded to a task workspace, Information Navigation (IN) will identify the input and output standard specifications in the Model, and then copy the links of the documents to the workspaces of the user-defined tasks in between the corresponding specifications in UDP.

As an example in Figure 7-21, 'Software Safety Validation Plan' has just been uploaded to the workspace of T1. Here the 'Software Safety Validation Report' is the post-condition of the next standard task. In UDP, the tasks in between the input and output specifications are T2, T3 and T4. Therefore, links to the document are put in the workspaces of T3 and T4 also.

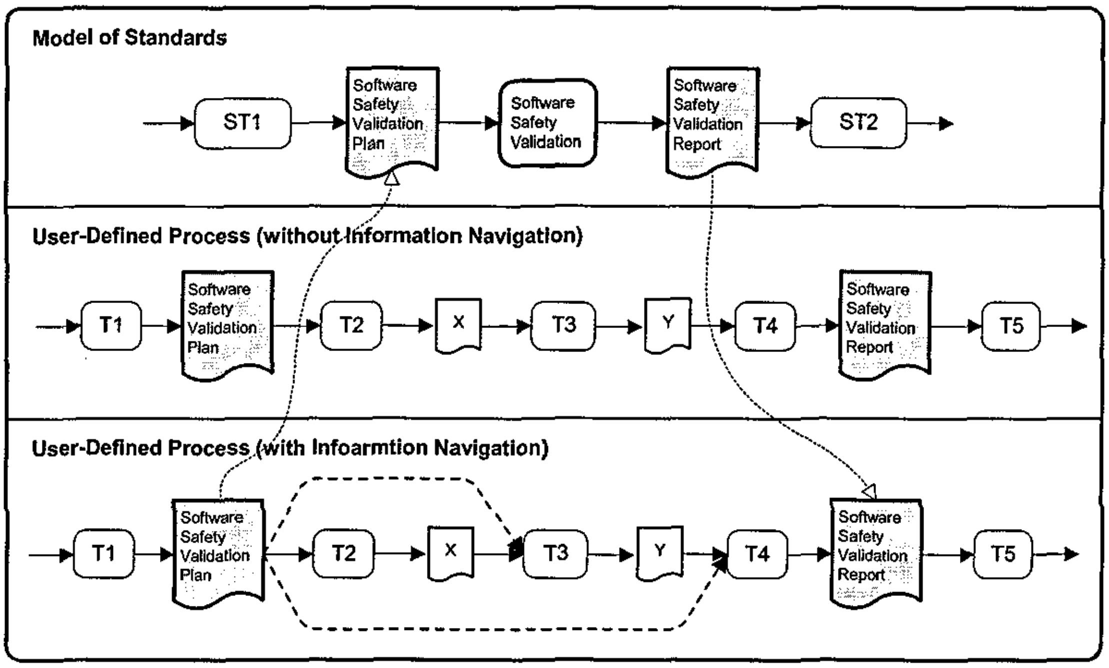

Model of Standards
ST1
Software Safety Validation Plan
Software Safety Validation
Software Safety Validation Report
ST2
User-Defined Process (without Information Navigation)
T1
Software Safety Validation Plan
T2
X
T3
Y
T4
Software Safety Validation Report
T5
User-Defined Process (with Infoarmtion Navigation)
T1
Software Safety Validation Plan
T2
X
T3
Y
T4
Software Safety Validation Report
T5

Figure 7-21. IN when UDP and the Model are at the same level of abstraction.

7.3.8.2 UDP at a Different Level of Abstraction than the Model

When UDP is at a different level of abstraction than the Model, the mechanism to identify the corresponding specifications of the input and output specifications is the same as described in section 4.1.

For some tasks in the Model, a specification is required by more than one task. This implies that all the tasks in between that specification and every output specification require that specification too. An example is given in Figure 7-22.

In Figure 7-22, specification B2.1.1 has just been uploaded to the workspace of T1. Its corresponding parent-specification B2.1 in the Model is required for standard task ST3 and ST4 where B2.2 and B2.3 are the output specifications. The corresponding specifications for B2.2 and B2.3 are B2.2.1 and B2.3.1 respectively, which forms two process paths: from B2.1.1 to B2.2.1 and B2.1.1 to B2.3.1. The tasks on the paths are assumed that B2.1.1 is required for their execution. Therefore, links to the documents are put in the workspaces of T3, T4 and T6.

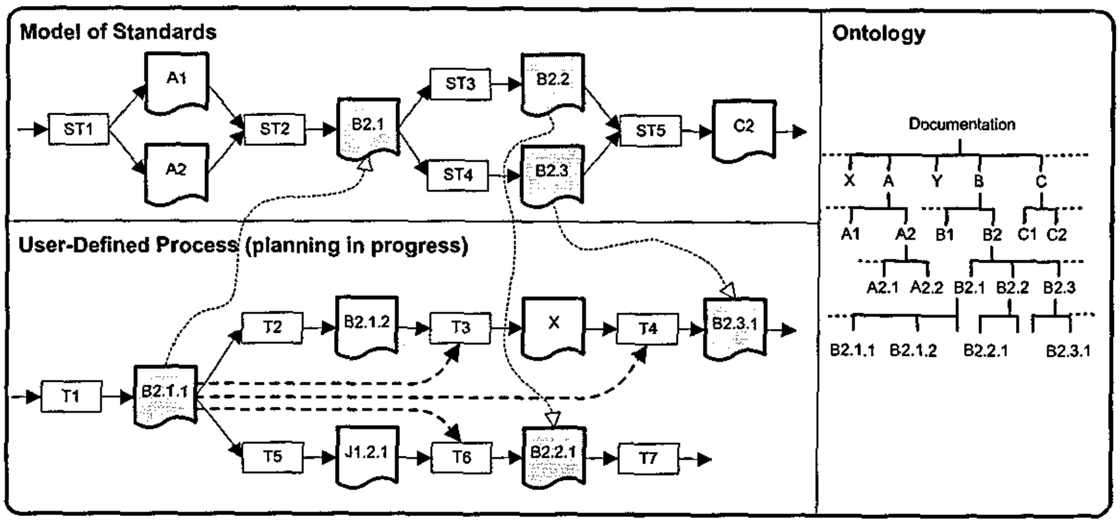


Figure 7-22. IN when UDP at the more detailed level of abstraction than the Model.

The Information Navigation algorithm is given in Figure 7-23.

```
Function NavigateDocument(doc, workspace, Std)
      doc: A document uploaded into a workspace of a
            user-defined task.
      workspace: The workspace where the document is uploaded.
      Std: The Model of a standard.
      task = GetTask(workspace);
      StdPreCond{…} = (x:x ∈ Standard Specification);
      StdPreCond = GetCorrespondingStdPreCond(doc,Std);
      StdPostCond{…} = (x:x ∈ Standard Specification);
      StdPostCond = GetStdPostCond(StdPreCond,Std);
      StdTask{...} = {x:x ∈ Standard Task};
      For each i ∈ StdPostCond
            StdTask = StdTask ∪ {GetAssociatedTask(i)};
      Next i
      TaskPreCond{…} = {x:x ∈ Task Pre-Condition};
      For each i in StdPostCond
            TaskPreCond = TaskPreCond ∪ _
            {GetCorrespondingTaskPreCond(i)};
      Next i
      UserTask{…} = {x:x ∈ User-defined Task};
      For each i in TaskPreCond
            UserTask = UserTask ∪ {GetAssociatedTask(i)};
      Next i
      PossibleTask{…} = (x:x ∈ User-defined Task);
      Workspace{…} = {x:x ∈ objects in a workspace};
      For each i in UserTask
            PossibleTask = GetTasksInBetween(task, i);
            For each j in PossbileTask
                  Workspace = GetInformationObjects(j);
                  If ({doc} ⊂ Workspace = 0 ) then
                        Workspace = {doc} ∪ Workspace;
                  End If
            Next j
      Next i
End Function
```

Figure 7-23. Algorithm of copying a document to the possible workspaces.

7.3.9 Cross-Referencing

Finally, given a task or specification in UDP, Cross-Referencing will locate the relevant part of the standard and the associated information. The inverse search is also allowed, it identifies all user-defined tasks or specifications that relate to a particular part of the standard.

7.3.10 Error Prevention

Error Prevention is performed at run-time to prevent the execution of non-compliant tasks. It includes:

- Correctness Check to ensure the required specifications are modelled properly.
- Completeness Check to ensure the required specifications are included.
- Capability Check to ensure the task agent possesses the required capability.
- Recommendation Check to ensure the required techniques are considered.

When the workflow engine starts a task, Error Prevention will perform the checks. Execution can go ahead only after passing all the checks or an explanation is provided, otherwise the task will be frozen until all the detected errors are resolved.

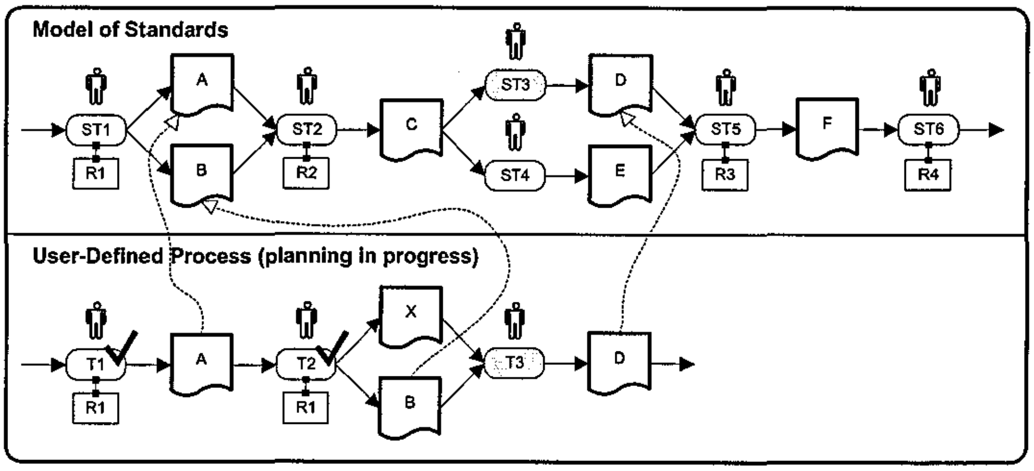


Figure 7-24. An example of Error Prevention.

Consider Figure 7-24, the user-defined tasks T1 and T2 are completed and the required specifications X and B are ready. When T3 starts, for which ST3 is the corresponding standard task, Error Prevention will be performed to ensure process compliance: Correctness Check ensures that specification D is placed in a correct sequence; Completeness Check ensures that specification C is available; Capability Check ensures that the task agent possesses the required capability; and Recommendation Check ensures that the required techniques or skills are used. Because specification C is not available, the completeness check failed and T3 is not allowed to proceed.

7.4 Coupling Issue

The development processes in different projects can be very different because of their details. Checking the user-defined process against the Model is the most flexible way to tackle the compliance problem. The degree of compliance relies greatly on the level of detail the information that the Model of Standards provides and the matching algorithm.

Sometimes, multiple standards may be used in an engineering project. To deal with this situation, there are two approaches: (1) employ a number of compliance agents, each of which is responsible for handling a particular standard, or (2) employ a single compliance agent that is capable of handling multiple standards. The difference between these two approaches is the degree of coupling between the compliance agent and the Model of Standards.

In the first approach, different standards may be modelled in different ways and the checking algorithm varies for each standard. This approach may achieve a higher degree of compliance as it can describe a standard more precisely and a dedicated algorithm is used to tackle any special features in a particular standard. However, system applicability is impaired as standard modelling cannot be performed by the user and programming work is normally required when a standard is updated or a new standard is required.

Compliance Flow advocates the second approach where a generic standard modelling language is developed with the capability of modelling a wide range of standards together with a comprehensive set of checking algorithms to ensure compliance. Users are able to model or update a standard without any programming involved. This enables users to tailor the Model to fit their organizations.

7.5 Summary

This chapter introduced the compliance checks performed by the compliance agent. Correctness Check, Completeness Check, Capability Check and Recommendation Check can identify errors at process build-time while Error Prevention is able to prohibit the execution of non-compliant tasks at run-time. Cross Referencing function allows the user to locate the relevant part of a standard by giving a user-defined task or specification, or vice versa. Planning Assistance and Information Navigator provide planning support and information management at build-time and run-time respectively.

Compliance checks could be performed at different stages during process build-time and run-time when the sub-processes may be at different levels of abstraction from the Model. There are three possible situations: (1) a process is at the same level of abstraction as the Model; (2) a process is at a higher level of abstraction than the Model; and (3) a process is at a more detailed level than the Model. Some compliance checks have different behaviours during process build-time and run-time, and which are identified and discussed.

Finally, the degree of coupling between the compliance agent and the Model was discussed. Compliance Flow advocates the use of a generic modelling language and a comprehensive set of checking algorithms to deal with multiple standards.

Part 4

System Development and Evaluation

Chapter 8

Design and Implementation

"There are two ways of constructing a software design:
one way is to make it so simple that there are obviously no deficiencies,
and the other way is to make it so complicated that there are no obvious deficiencies.
The first method is far more difficult."
- C.A.R. Hoare

8.1 Introduction

Compliance Flow prototype is developed in order to demonstrate and evaluate the proposed workflow framework for supporting the management of dynamic engineering processes, for which it has to include three features: process modelling, process management and compliance management.

This chapter presents the design and implementation of Compliance Flow. It is organised as follows: §8.2 describes the process modelling methodology which provides the basis for supporting process management; §8.3 describes standard modelling which is used to provide information for compliance management, §8.4 depicts the workflow enactment process; §8.5 discusses the system architecture. §8.6 presents the four tier object-oriented implementation architecture; §8.7 discusses the proposed capability matching approach and the fuzzy matching algorithm used. A chapter summary is given in §8.8.

8.2 Process Modelling

To facilitate process management, an enhanced activity-based modelling methodology is developed. In this methodology, a process model not only models the task structure of a project, but also captures the capabilities that are required of the tasks and possessed by the agents. In addition, it links the information objects that are required or created during task execution to the related tasks. The meta-model of process modelling is illustrated in Figure 8-1.

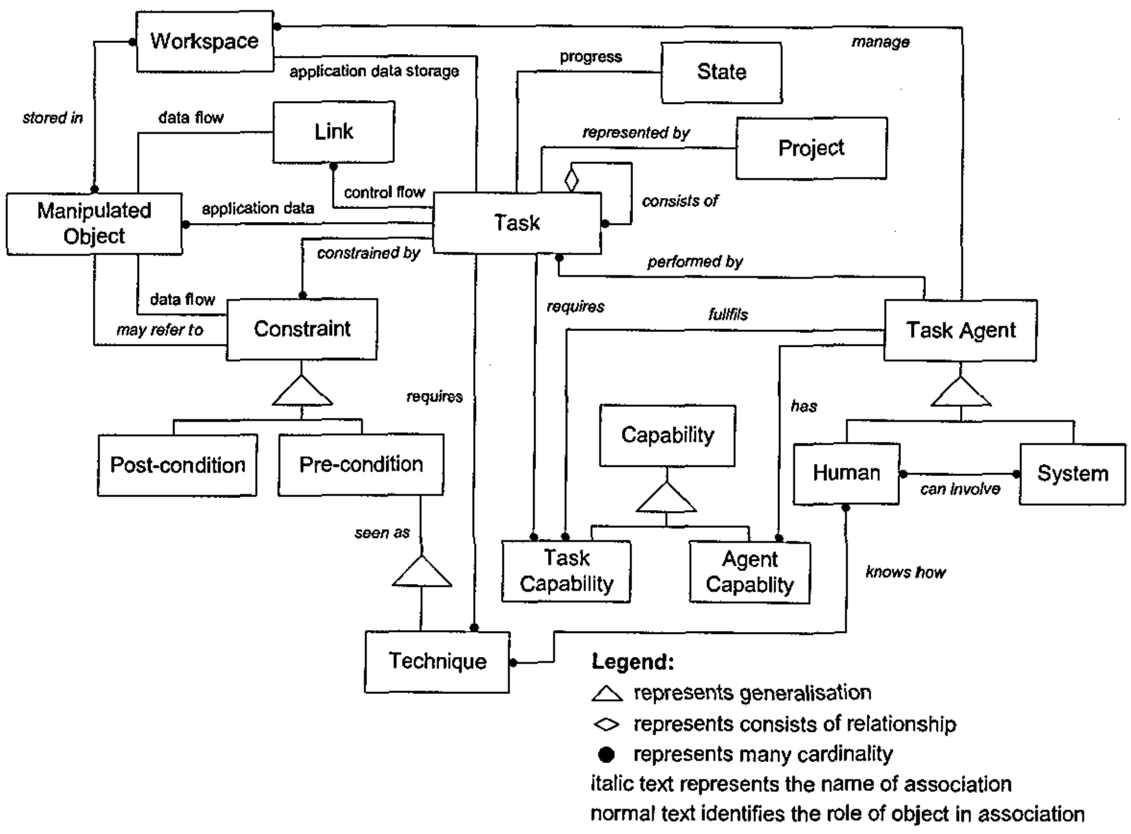


Figure 8-1. Meta-process model.

In Compliance Flow, a project is represented as a *task* which can be further divided into sub-tasks. Task decomposition can be performed to any level of detail, which forms a Hierarchical Task Network (HTN). The tasks at the lowest level are concrete work instructions, while the tasks at higher levels are more abstract. HTN helps to manage complexity by hiding less important details until they are required.

The execution sequence of tasks is modelled through links. A *link* represents the connection between two tasks and the execution order. The input and output links of a task have to be maintained in the task decomposition, as illustrated in Figure 8-2. Only the leaf tasks that have no child tasks will finally be performed.

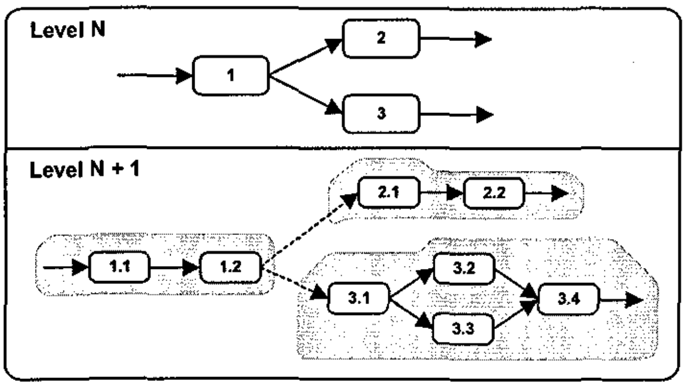


Figure 8-2. Interfaces between parents and their children.

Task execution is constrained by the pre- and post-conditions. A *pre-condition* normally represents information that is required for the execution of a task, and it has to be ready or true before the task can begin. A *post-condition* is an objective of a task that will be achieved if the task is completed successfully. The different types of task interdependence discussed in Chapter 4 that can be modelled are illustrated in Figure 8-3.

In Figure 8-3, the rounded rectangles are tasks, the white document icons are pre- and post-conditions. The pooled and sequential interdependence can be modelled in a traditional way. To model the mutual interdependence where the output of task T1 is the input of another task T2 and the output of T2 is the input back into task T1, the recursive loop is represented by a linear task flow. As in the example, document A is used in task T1 to create document B and document B is used in task T2 to create document C that is needed to send back as input to task T1. A shadow task T1' that has the same concern as task T1 is used with both the documents A and C as inputs, to capture an interaction of the recursive loop.

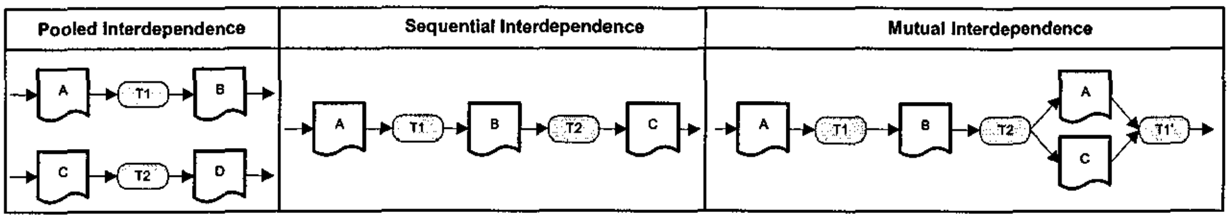


Figure 8-3. The modelling of different types of task interdependence.

Pre- and post-conditions can only be assigned to leaf tasks. The pre-condition of a parent task is the collection of all the pre-conditions of its child-tasks. Thus, the higher level a task is, the more pre-conditions it has. This is also true of post-conditions. Pre- and post-conditions are often represented as documents. Another type of pre-condition is the *technique* that has to be used to carry out a task.

The assigning of execution constraints at the lowest level of tasks fits the planning practice of engineering projects. A project plan usually starts at an abstract level where the constraints are defined at a relatively high level. While the delegation-planning continues, these constraints, like the tasks, will be refined and assigned to the appropriate sub-tasks. The constraints assigned to the higher level tasks are removed, but the task flow and structure are maintained.

A task is performed by a *task agent* and the progress of the execution of the task is represented through a number of *states*. A task agent could be a *human* or a *system*. A system may need a human to operate it. A *capability* is a skill, technique, method, knowledge or any attribute that an agent required to perform a task. While the required capability of an agent to perform a particular task is defined as the *task capability*, the *agent capability* is used to describe the capability possessed by an agent. The specifications of task capabilities and agent capabilities enable automatic identification of the most appropriate agent for a task.

Every task is associated with a *workspace* that is used to store the *information objects* related to that task, including the documents corresponding to the pre- or post-conditions of the task. A workspace associated with a high level task contains all the information objects of its child-tasks. A workspace is managed by the agent assigned to that task. Information objects can be copied or moved from one workspace to another. The information objects in a workspace are hyperlinks that link to the concrete objects stored in a secured space, such as a FTP server. Therefore, an information object may appear in multiple workspaces, but only one physical copy is stored.

8.3 Standard Modelling to Support Compliance Management

The information of a standard that is required in compliance checks is captured through a standard process model called the Model of Standard or simply the Model. The meta-model of standard modelling is illustrated in Figure 8-4. In this meta-model a *standard* is represented as a Hierarchical Task Network (HTN). A *pre-condition* of a task has to be fulfilled before the execution of that task. The objective of a task is seen as a *post-condition* that will be achieved after the completion of the task. The task flow is modelled using *links*. According to the standard, the agent of a task requires certain *capabilities*. The standard also gives *recommendations* that need to be considered for task execution. A recommendation may be a *technique* and its *applicability* is subjected to some *requirements*.

Both the User-Defined Process (UPD) model and the Model of Standards are designed based on the same activity-based modelling methodology, and thus facilitates compliance check between the two models.

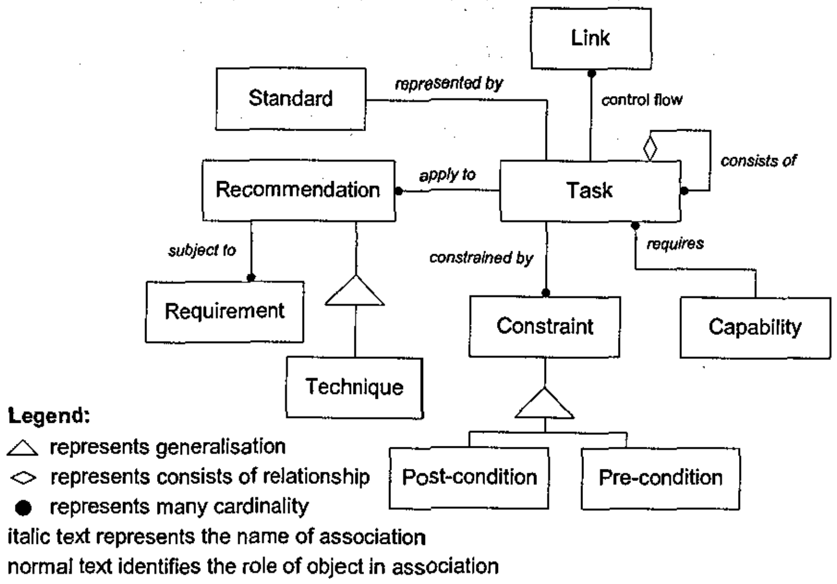

Link
Standard
represented by
control flow
consists of
Recommendation
apply to
Task
subject to
constrained by
requires
Requirement
Constraint
Capability
Technique
Legend:
△ represents generalisation
◇ represents consists of relationship
● represents many cardinality
italic text represents the name of association
normal text identifies the role of object in association
Post-condition
Pre-condition

Figure 8-4. Meta-model of the standard modelling.

The Model and compliance agent are designed to work independently of Compliance Flow's workflow engine. Therefore, the compliance checks can potentially be deployed on other WfMSs where the user-defined processes are represented using different modelling languages. The most open approach is to convert the user-defined process stored in a particular WfMS into a standard format, such as XML Process Definition Language (XPDL), and perform the compliance checks upon it. This open architecture that enables compliance management of different systems is illustrated in Figure 8-5.

To adapt to this open architecture, the Model can remain in its proprietary format but a little amendment to the compliance agent is required to access a process specified in XPDL format. A middleware can be used to convert the process specification used in a particular WfMS into XPDL format. Therefore, to deploy compliance checks to a new WfMS where a process is represented using a proprietary language, only the middleware needs to be changed.

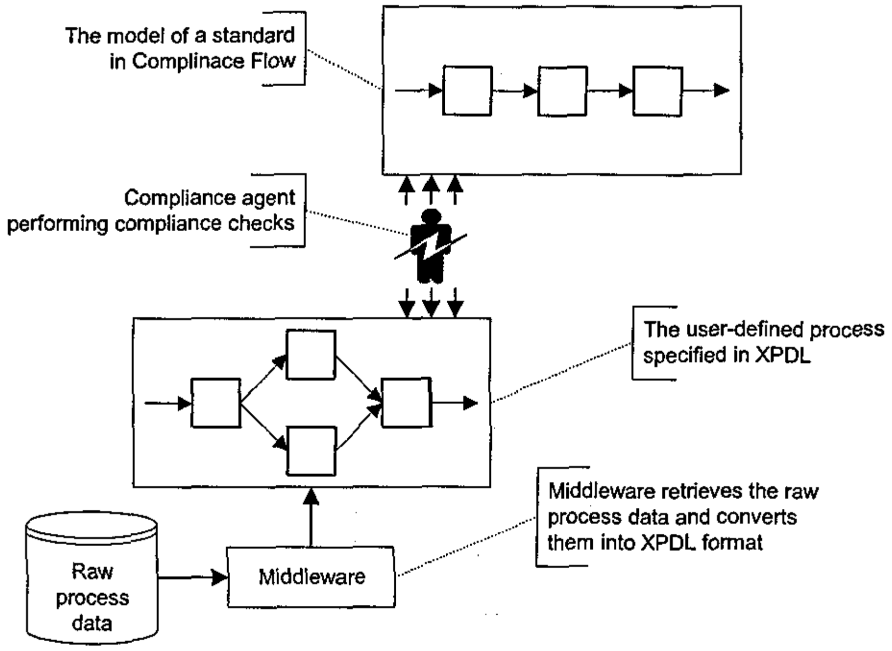

The model of a standard in Complinace Flow
Compliance agent performing compliance checks
The user-defined process specified in XPDL
Middleware retrieves the raw process data and converts them into XPDL format
Raw process data
Middleware

Figure 8-5. An open architecture to deploy compliance agent.

8.4 Task Enactment

Once a task is specified, it can be started if (1) all of its pre-tasks are completed successfully and (2) its pre-conditions are fulfilled.

An agent who has been delegated a task can execute it in two different ways:

Direct. Work is done directly on the task. Tools and documents may be used to get results. Normally, the results are recorded in a document and it is uploaded to the associated workspace after the task has been completed.

Delegation. A task can be delegated to other agent(s). This can be done by delegating the whole task or by dividing the task into sub-tasks and delegating the sub-tasks to other agents.

The progress of a task is represented by a state variable which is maintained by the workflow engine. A task can be in six possible states, as illustrated in Figure 8-6.

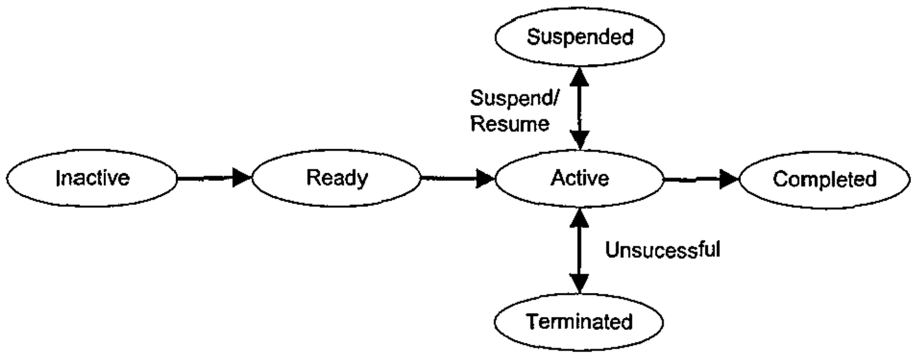


Figure 8-6. States in a task lifecycle.

Inactive. This is the initial state of a task, indicating that the pre-conditions are not fully fulfilled or the pre-tasks are not yet complete. A parent task is indicated as Inactive when none of its child tasks is being performed or at least one of its pre-conditions is not ready.

Ready. The system will automatically turn a task to this state when all of its pre-conditions are ready and the pre-tasks are completed. A task can start at any time by turning its state to 'Active'.

Active. This state indicates a leaf task is being performed. A parent task is stated as Active if one or more of its child-tasks is being performed.

Suspended. This state indicates the execution of a leaf task is temporarily stopped but it can be resumed later. A parent task is stated as Suspended when all of its child-tasks are suspended.

Terminated. This state indicates the failure of the task execution caused by some internal or external problems, which will lead to either reprocessing of some tasks or changes made to the process plan. If a parent task is stated as Terminated, then all of its child tasks will be terminated.

Completed. This state indicates a leaf task is completed successfully and its post-conditions have been achieved. A parent task is stated as Completed when all of its child tasks are completed.

8.5 Three-Tier System Architecture

Compliance Flow is designed and developed as a web-based prototype, intending to be deployed on a three-tier system architecture based upon Microsoft technologies. The system front-end is composed of a variety of ActiveX components, developed using Microsoft Visual Basic (VB), and embedded in a web page. The system back-end is supported by a Microsoft SQL server connected to an Internet Information Server (IIS), which is the middle tier of the data access architecture, as depicted in Figure 8-7.

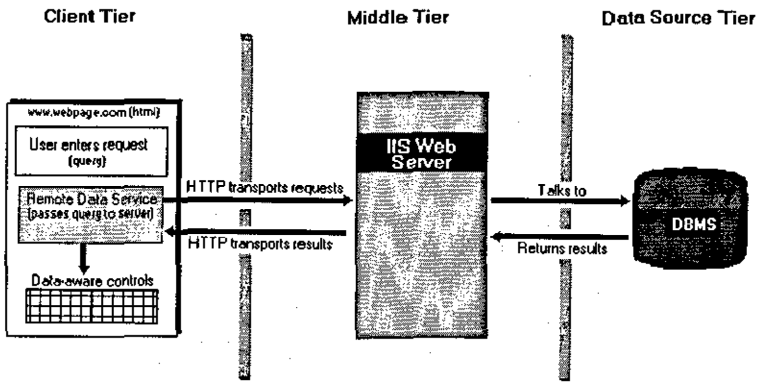


Figure 8-7. Three tier Remote Data Services (RDS) architecture.

Three-tier data access uses an intermediate agent to handle data between the client and database components, which is typically used to access data across the Internet. The ActiveX components on the client side communicate with the IIS web server via Remote Data Service (RDS) on Hypertext Transfer Protocol (HTTP). When a request is launched, the client-side RDS component sends a query to the IIS web server. The server side RDS component processes the request and then sends it to the SQL server. The SQL server responds to the request, sending back the data. The IIS Web server then transfers the data back across the Internet to the client-side RDS component where the ActiveX components can access.

It is noted that directly accessing a SQL server database through Web protocol, such as HTTP or FTP, is impractical. In most cases, these standard protocols are the only way for people outside of their company to access the web site as security firewalls and proxy servers would not allow any other kind of traffic. Thus, connecting the database server to a web server is the best way to enable an Internet database access for a web or Internet based application.

Once a user opens the web page, the ActiveX components together with the required libraries will be downloaded to the client's workstation. As such components are built upon the Microsoft's architecture, it can only run on IE browsers that are running on Microsoft Windows operating systems.

8.6 Four Tier Object-Oriented Implementation of the Client Tier

Every component inside the system is designed as an object. These objects are arranged into four tiers, as shown in Figure 8-8. An object can communicate with other objects in the tiers immediately above or below it.

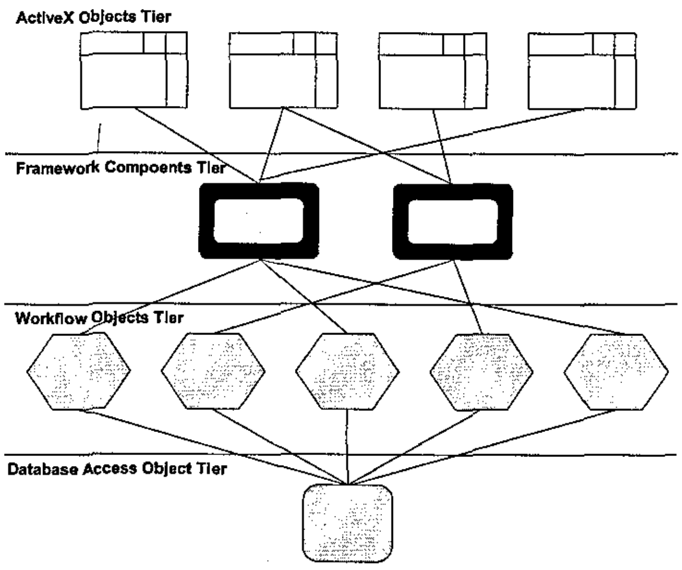


Figure 8-8. Four tier object-oriented architecture.

The first tier is the ActiveX objects which are embedded in the web pages to form the user interface. ActiveX objects are built on the functions provided by the framework components in the second tier where the system functions are abstractly presented. The more concrete workflow objects, such as the tasks, ontologies, etc are in the third tier. Such objects are saved in a SQL database, and therefore the bottom tier is a database access object that is used to provide the required database connection.

There are a number of advantages of using this four tier object oriented architecture:

Separation of concerns. The major advantage is that the different tiers are clearly separated, particularly the separation of the process objects from the data manipulation facility, which allows details of the data storage mechanisms to be abstracted away from the client processes. All the objects in the higher tier are operation requests for output from lower level objects.

Objects implementation and modification. Objects can be implemented independently which reduces the complexity of system development. New high level objects can be introduced or existing objects can be removed without, or with little needs, of modification to lower level objects. Furthermore, modification to database access is not required.

Enabling database migration. Database upgrades, migration or other changes can be performed without the necessity to alter the high level objects.

Reduced network loading. Required objects are instantiated on the client side after the information is transferred from the Web server. They together with the newly created objects will stay on the client side for further reuse. Thus, the database throughput and the network loading are reduced, which is particularly important in the stateless Internet connection. However, a mechanism to verify the update of the objects is still required to ensure the integrity in a multi-user environment.

In order to reduce the programming load, four third party's ActiveX components are used:

- DataExplorer from Infragistics Inc. (www.infragistics.com), which facilitates the creation of an outlook style User Interface (UI) where the system functions are integrated.
- AddFlow from Lassalle Technologies (www.lassalle.com), which facilitates the creation of a visual process editor by providing a drawing tool where nodes and links can be created visually.
- HFlow from Lassalle Technologies (www.lassalle.com), which is an associated object of AddFlow component. HFlow enables hierarchical layout of a process in the visual process editor.
- KFTP from Katarn Corporation (www.katarncorp.com), which simplifies the programming of uploading and downloading of files to and from a FTP server.

8.6.1 First Tier - User Interfaces

A user only requires interacting with a single web page to assess all system functions. The web page consists of an ActiveX component that is composed of a number of other reusable ActiveX objects. Figure 8-9 is the system function chart where every function is presented through a screen, which is composed of one or more small ActiveX components. All basic functions required for process management are integrated inside the Task Manager, and therefore it is the only interface required for most users. Some system functions, such as standard modelling and ontology manipulation, are organised separately.

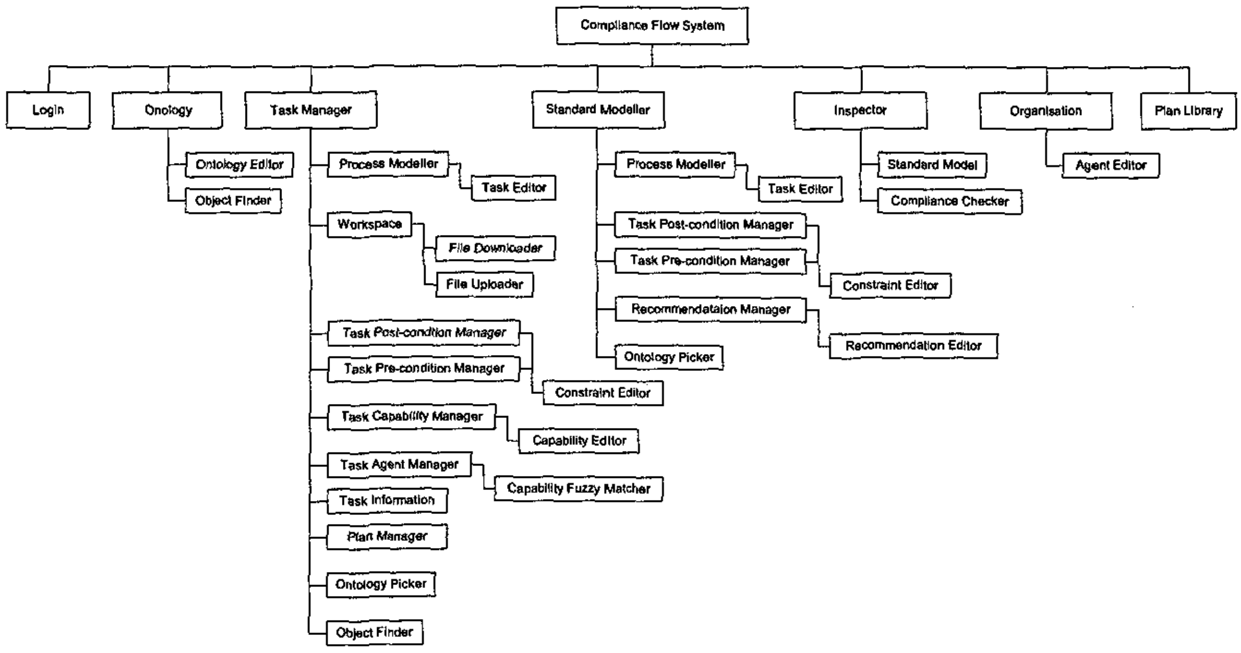
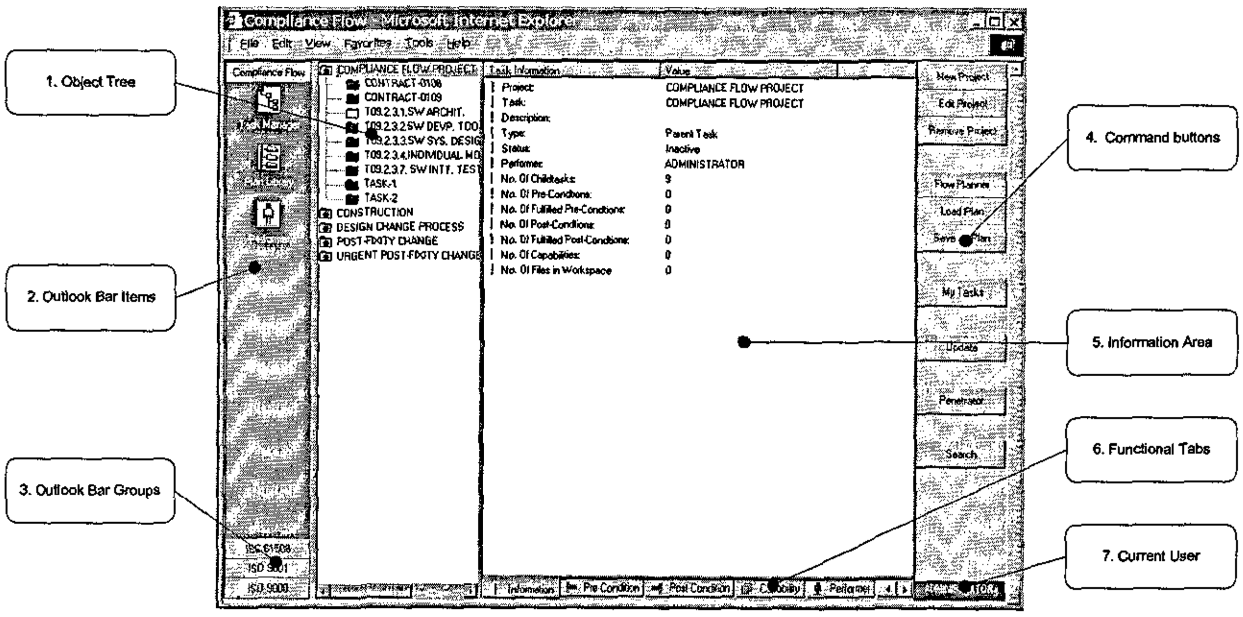


Figure 8-9. Function Structure Chart.

System functions are integrated into an Outlook style interface, as depicted in Figure 8-10. On the left hand side are the Outlook bar items (2) where each item represents a key function group, such as Task Manager, in the function structure chart. The outlook bar items are grouped into appropriate outlook bar groups (3). The system items are located in the 'Compliance Flow' group and a standard model is organised in an individual group. Object tree (1) is used to represent hierarchical structure of tasks and ontologies. The information area (5) shows the particular information of the selected item in an object tree. Information is presented in different functional perspectives based on the function tab (6) selected by current user (7). On the right hand side is a set command buttons which can be applied to the current selected item. The Inspector can be called by right clicking the mouse button to perform compliance checks.



Task Information	Value
Project:	COMPLIANCE FLOW PROJECT
Task:	COMPLIANCE FLOW PROJECT
Description:	
Type:	Parent Task
Status:	Inactive
Performer:	ADMINISTRATOR
No. Of Childtasks:	9
No. Of Pre-Conditions:	0
No. Of Fulfilled Pre-Conditions:	0
No. Of Post-Conditions:	0
No. Of Fulfilled Post-Conditions:	0
No. Of Capabilities:	0
No. Of Files in Workspace	0

Figure 8-10. Screen layout.

8.6.2 Second Tier – Framework Components

Each framework component provides a number of specific services, some of which are used to support other components. A component performs specific functions and can request services from other components. The relationship between the framework objects with their key attributes and major operations are presented in the class diagram in Figure 8-11. The operation descriptions please refer to the appendix.

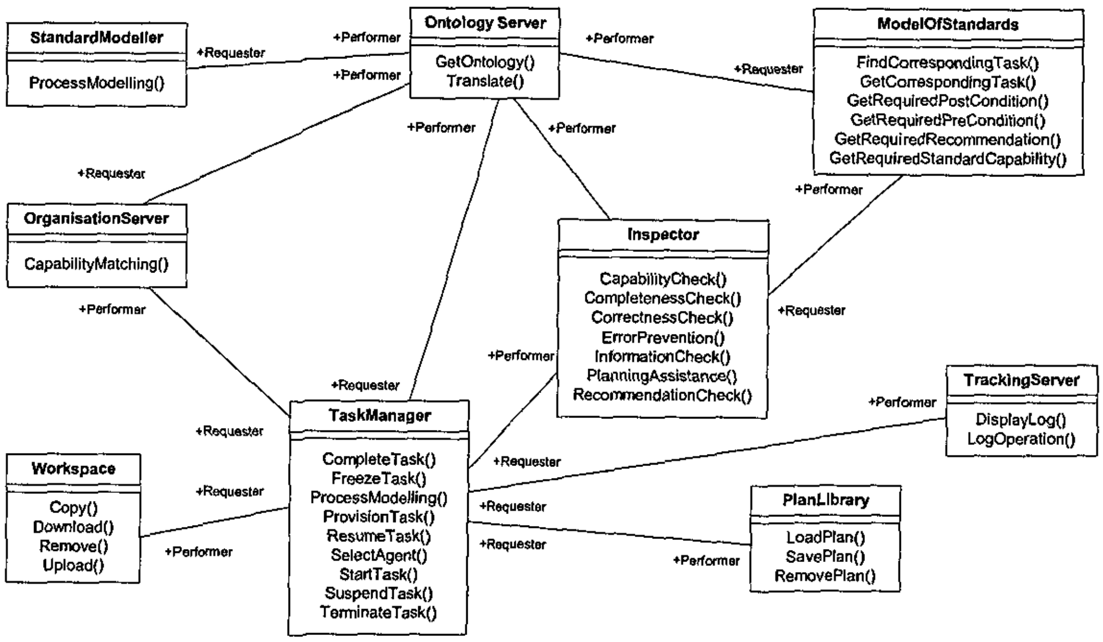


Figure 8-11. Class diagram of the framwork components.

8.6.3 Third Tier – Workflow Objects

The third tier consists of workflow objects. They can be conceptually mapped with the entities in the meta-process model given in Figure 8-1. Their operations provide services support to the second tier. Unlike the component objects in the second tier, multiple instances of an object class could exist in the system at the same time. For example, one Task Manger (framework component) can manage multiple tasks (workflow objects) at a time. The identification of an object is based on a Globally Unique Identifier (GUID) that is assigned to every object when it is first created. The workflow objects can be grouped into four categories: (1) User-defined Process, (2) Model of Standards, (3) Plan, and (4) Ontology.

All objects in the third tier have four basic operations:

- Create a new object and assign it with a GUID.
- Update the attributes of an object and its information in the database.
- Remove an object from the system, including its record in the database.
- Find an object.

8.6.3.1 Workflow Objects in User-defined Process

A user-defined process comprises of a number of objects, including tasks, links, pre- and post-conditions, task capabilities, agent capabilities, task agents, and workspaces.

The hierarchical structure of tasks is captured by adding a 'parent' attribute to the task object. It stores the GUID of its parent so that the path of a hierarchy can be identified by searching the parent-child relationship. For a root task which has no parent, the value of its 'parent' attribute will be its own GUID. Link object is used to capture the flow of tasks at the same level in a hierarchy.

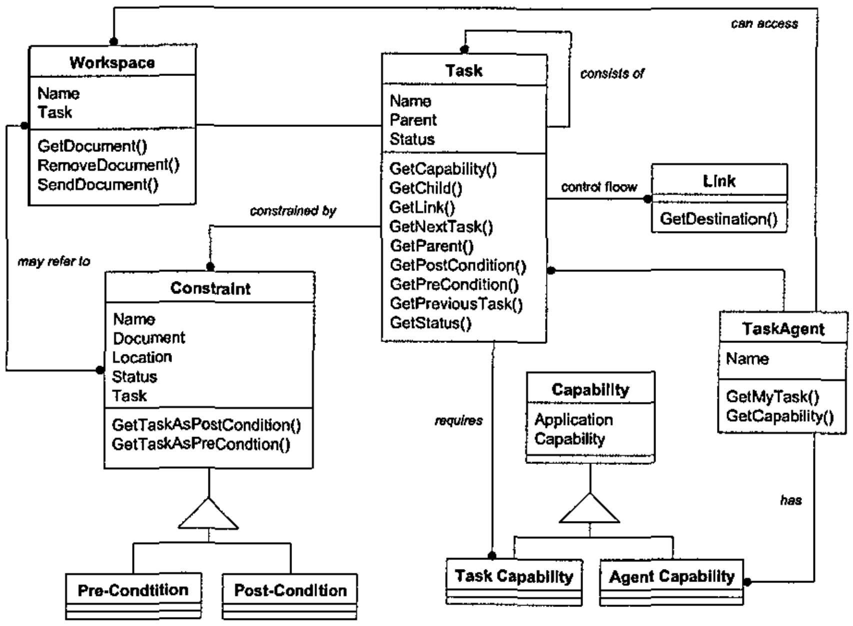


Figure 8-12. Class diagram of user-defined processes.

The relationships among the workflow objects together with their key attributes and major operations are presented in a class diagram in Figure 8-12, followed with operation descriptions. The four common operations of each object are not included to make the diagram simpler.

8.6.3.2 Workflow Objects in the Model of Standards

The Model of Standards comprises of the objects in a standard, including tasks, links, pre- and post-conditions, capabilities, and recommendations. Like the user-defined process, a standard is represented as a Hierarchical Task Network (HTN) with the details of each task included. Thus, the objects inside a standard model are similar to the workflow objects in a user-defined process. One key difference between them is that the required techniques are modelled as pre-conditions in a user-defined process, but as recommendations in the Model of Standards.

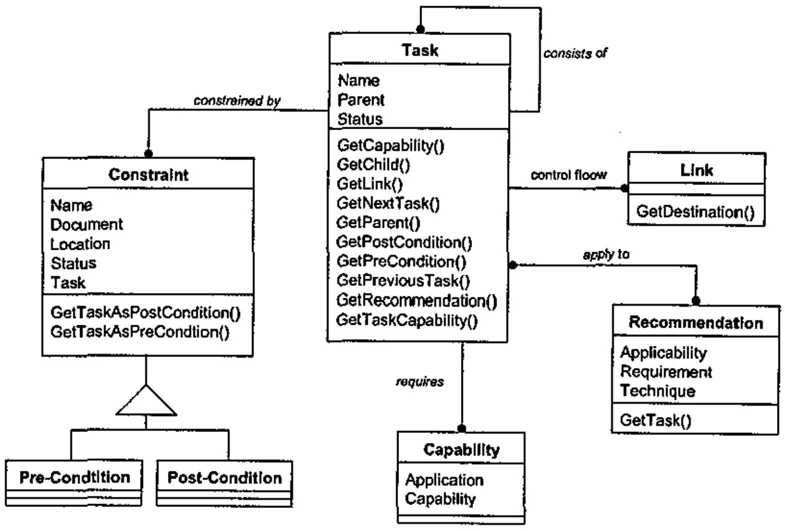


Figure 8-13. Class diagram of the Model of Standards.

The relationships among the objects with their key attributes and the major operations are presented in a class diagram in Figure 8-13, followed with the description of object operations which do not exist in the user-defined process. The four common operations of each object are not included.

8.6.3.3 Workflow Objects in the Plan

A user-defined process can be saved as a Plan for future reuse. A Plan is a simplified form of a user-defined process, which consists of tasks, links, pre- and post-conditions, and standard specified capabilities.

Plans are categorised and stored into different Plan group folders according to their natures in order to ease later retrieval. Plan groups are hierarchically organised. Therefore, all the Plans in a Plan group represent different possible solutions of a situation. The higher level of Plan group in the hierarchy, the more abstract its solutions will be.

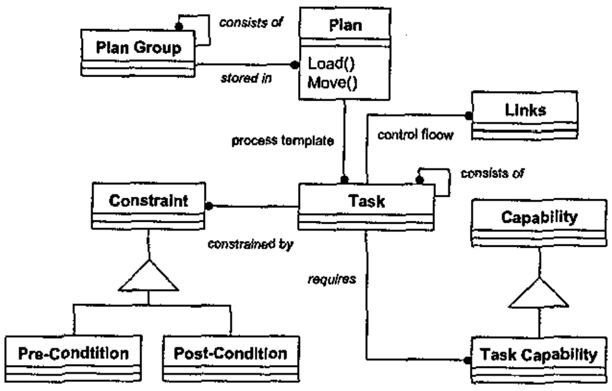


Figure 8-14. Class diagram of plans.

8.6.3.4 Ontology Object

An ontology is a set of terms which are organised into a hierarchy. Each term could have multiple synonyms. The terms for naming the different workflow objects are drawn from the appropriate ontology hierarchy. For example, capabilities are drawn from the capability ontology. The relationship between an ontology and a workflow object is presented in the class diagram in Figure 8-15.

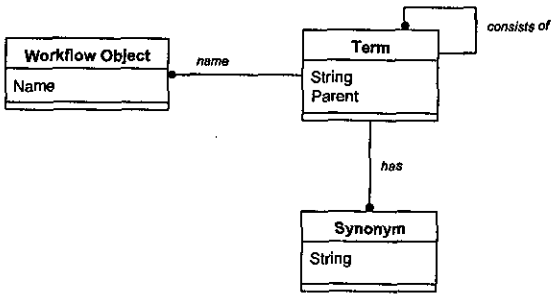


Figure 8-15. Class diagram of an ontology.

Each term is represented as an object. Like other workflow objects, the GUID is its identity and other information is represented as its attributes. Thus, a workflow object is named by giving a GUID of a term object. If the term is changed, the names of all its named objects will be changed automatically. This ability of dynamic ontology update is critical because of its dynamic nature: new terms or synonyms may be added and updated to the ontologies while processes have been started.

8.6.4 Fourth Tier – Database Access Object

All workflow objects in the third tiers are stored in a database. The database access object located in the fourth tier provides a connection to access the database, including adding, updating, removing, searching and retrieving the records. A large number of process data, such as tasks, ontologies, and the plan group folders, are organised in a hierarchical structure. An enhanced adjacent model (please refer to appendix) is used to manage such hierarchies. In the model, a node not only remembers its parent, but also the number of its children and the level it is allocated in the hierarchy. The addition or re-allocation of node may lead to changes of others which need to be automatically updated. Therefore, this enhanced adjacent model can only be managed by the database management systems (DBMS), such as MS SQL server and Oracle, which have the ability of dynamically triggering the user-defined procedures. The database structure and the hierarchical data handling technique used are provided in the appendix.

8.7 Capability Matching

In the capability matching process, the *available capability* is matched against the *required capability* and a figure in the interval (1, 100) is used to indicate their Goodness of Fit (GOF). An example of an available capability is the agent capability, and a capability specified by a standard is an example of required capability. Capability matching is used agent selection and Capability Check. To enable capability matching, the two sets of capability have to be specified in the same way.

8.7.1 Capability Specification

To facilitate capability matching, the main desired features of capability specifications are:

- Preciseness: The capability knowledge should be modelled to provide a precise capability specification. The entities in the model are quantitatively related to facilitate fuzzy matching.
- Expressiveness: The method of capability specification should be expressive enough to represent not only the technical capability, but also a wide range of knowledge such as qualification, role and authority, which are relevant for assessing suitability of an agent in performing a task.

- Comprehensiveness: A user can easily understand the capability representation and specification statement, and the system is able to process it.
- Ease of use: Every capability description should not only be easy to read and understand, but also easily defined by the user.

Capability knowledge is captured by two ontologies: Capability Ontology and Application Ontology. Capability Ontology describes the domain specific knowledge possessed by agents or required to perform particular tasks. Application Ontology describes the application areas of the specific knowledge defined in the Capability Ontology. Both ontologies are domain specified. The concepts of interest related to a particular domain are organised into a separate ontology hierarchy.

Examples of capability hierarchies are 'Programming' and 'Approach'. The Programming hierarchy describes the languages that can be used in system development, and Approach hierarchy defines the development methodologies. An example application hierarchy is 'System'. It describes the different types of software. An application hierarchy, sometimes, could be used with more than one capability hierarchies in capability specification. For example, the System hierarchy can be used with Programming and Approach hierarchies: a *programming language* is used to program a type of *system*, and a *methodology* is used to develop a type of *system.*

A capability description consists of two compulsory parts: *knowledge capabilities* denoted as t_c and their *application areas* denoted as t_a, i.e. *Capability(t_c,t_a).* The term t_c is drawn from the Capability Ontology and the term t_a is drawn from the Application Ontology. An example is *Capability(Logic,ProcessLogic),* which means that the agent knows all the *logic* programming languages and has experience in using these languages in programming the *process logic* of software system.

8.7.1.1 Traditional Ontology Hierarchy

A traditional ontology hierarchy is specified as a single generalisation hierarchy. The advantage of this hierarchy in terms of capability specification is that it provides a hierarchy of capabilities: a parent term includes all knowledge that is represented by its child-terms. For example, using the ontology given in Figure 8-16, if an agent is specified as knowing Object-Oriented Programming, it means that he/she knows all the object-oriented programming languages specified below the node "Object-Oriented" in the Programming hierarchy.

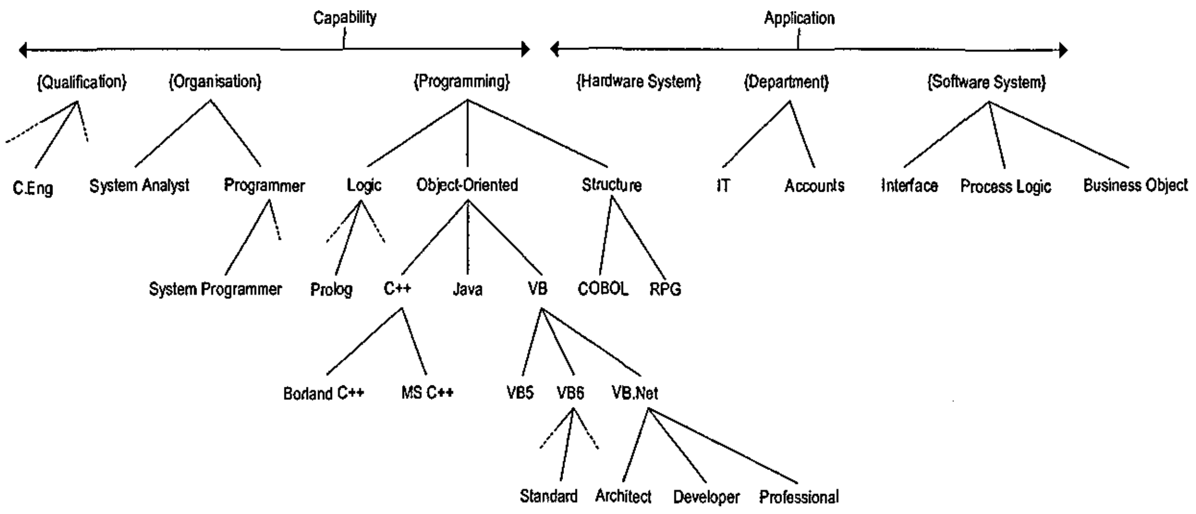


Figure 8-16. A traditional ontology hierarchies.

8.7.1.2 Traditional Approach to Assess GOF

The purpose of capability matching is to assess the Goodness of Fit (GOF) of an *available capability* set against a *required capability* set. However, finding a perfect match, i.e. all the required capability can be fulfilled perfectly by an agent, is highly unlikely.

Consider the example in Figure 8-17, all three available agents partially fulfil the requirements. To identify who is the most suitable candidate, it has to assess the closeness of each agent's capability with the required task capability.

Task	Result	Agent

Task-2:Coding
p1-Capability(C++, Process Logic);
p2-Capability(CEng, Software System);
p3-Capability(Programmer, IT);
	☑	**Agent-1:**
		Capability(C++, Business Object);
		Capability(Java, Software System);
		Capability(Programmer, IT)

	☑	**Agent-2:**
		Capability(VB, Process Logic);
		Capability(Prolog, Process Logic);
		Capability(Programmer, IT);

	☑	**Agent-3:**
		Capability(C++, Interface);
		Capability(VB6, Interface);
		Capability(System Analyst, IT);
		Capability(CEng, Software System);

Figure 8-17. An example of agent selection without a perfectly fitted agent exist.

The most common approach to assess the GOF is based on the idea of conceptual distance as discussed in (Rada et al., 1989). The conceptual distance (CD) between two terms is defined as the length of the shortest path that connects the terms in a hierarchy. This distance approach to estimate the similarity of words has been widely used, for example, in medical bibliographic retrieval system (Rada et al., 1989), spelling correction (Agirre et al., 1994), spatial query (Papadias and Delis, 1997) and word sense disambiguation (Fernández-Amorós et al., 2001) etc.

Assume o_a refers to a term used to describe the available capability and o_r refers to a term used to describe the required capability. The GOF of o_a against o_r is denoted as $GOF(o_a,o_r)$. The GOF between two terms is represented as a number in the interval (0, 100). The upper limit 100 implies a perfect match between two terms.

Table 8-1 provides some examples based on the "distance" method used in ReMind (1992), which is a case-based reasoning tool, according to the ontology hierarchies given in Figure 8-19. The equation used to assess GOF is:

$$GOF = \left(1 - \frac{IP + EP}{IR + ER}\right) \times 100$$

where

IP is the number of links between o_r and the common parent between o_r and o_a.

EP is the number of links between o_a and the common parent.

IR is the number of links between o_r and the root of the hierarchy, and

ER is the number of links between o_a and the root of the hierarchy.

From these examples, it can be seen that this approach does not always produce appropriate GOF values for the purpose of capability matching. Consider example 2 and example 3, if the required capability is Java, both available capabilities VB and C++ have the same GOF with value 50. Obviously, an agent who knows C++ language requires less effort to learn Java than another agent who knows VB.

Another problem can be found in examples 4 and 5. In example 5, the required capability is C++ and the available capability is Object-Oriented that includes C++ and thereby it should fully match the requirement. However, it has the same GOF as example 4 where the required capability is Object-Oriented and the available capability is C++. Finally, examples 6 and 7 show a serious problem when the root term is used as either o_a or o_r, the GOF value is 0 always. The results show that the symmetric nature and the lack of an ability to represent knowledge overlap make the distance approach is inappropriate for capability matching. A different way of assessing GOF is required and an ontology model that precisely describes the capability knowledge is necessary.

Example	o_a	o_r	$GOF(o_a,o_r)$
1	Java	Java	100
2	VB	Java	50
3	C++	Java	50
4	C++	Object-Oriented	66
5	Object-Oriented	C++	66
6	MS C++	Programming	0
7	Programming	MS C++	0

Table 8-1. Examples GOF calculation using traditional distance approach.

8.7.1.3 Treating Ontology as Sets

One way to improve the accurateness in assessing GOF is to enrich the hierarchy with additional information and develop an algorithm that makes use of the knowledge. Sussna (1993) proposed to assign a weight to each link in the Wordnet (Miller et al., 1990) hierarchy and calculates the closeness between two words as the total weight of the path with minimum weight. The use of weights to capture additional data, e.g. for the two pairs of terms with the same path length, the pair at lower level in the hierarchy seem to be conceptually closer. Another similar work (Agirre and Rigau, 1996) took the density of concepts in the hierarchy into consideration: concepts in a deeper part of the hierarchy should be ranked closer, and the Conceptual Density (Agirre and Rigau, 1995) formula is used to provide more accurate results. However, these approaches are still not suitable for capability specification and matching. The estimate of knowledge overlap in constructing the ontology hierarchy and capability matching is therefore proposed.

It is clear that the concepts represented by two sibling capabilities, i.e. both having the same parent, usually have certain overlap. For example, general programming knowledge can be treated as the overlap between the specific knowledge of C++, Java and VB languages. However, in general, it is easier for a C++ programmer to learn Java than a VB programmer. This is because besides the general programming knowledge, there is a further overlap between C++ and Java due to the similarity in their object-oriented approaches. The degree of knowledge overlap between capabilities can be estimated and given by a domain expert.

There are two basic overlapping relationships between two capabilities: parent-child relationship and sibling relationship as illustrated in Figure 8-18.

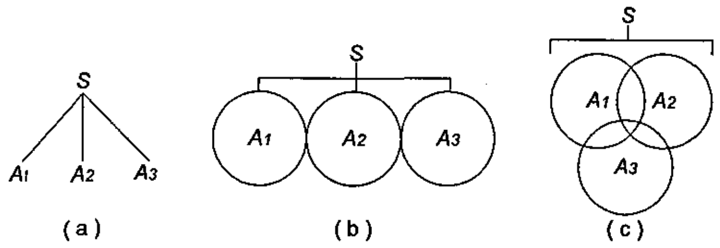


Figure 8-18. Relationship between capabilities.

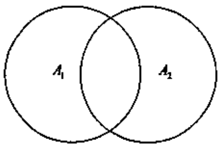
Figure 8-18 (a) shows that the term S includes three child-terms: A_1, A_2 and A_3. It is assumed that each child-term has the same importance in relation to the parent term. The default is the knowledge represented by the siblings terms do not overlap. In this case, the relationship can be illustrated using a Venn diagram as in (b). However, if the knowledge overlaps, as shown in (c), the user is allowed to give the values of overlap between these capabilities.

The conventional set theory is used as a basis to quantify the relationships between capabilities. The knowledge covered by a parent term is treated as a universal set S and each child A_i as a subset that represents a certain percentage of the knowledge $P(A_i)$ of S. An ontology hierarchy employs five basic axioms to guarantee consistent results from specifications and calculations.

1. For any capability $A_i \subseteq S$, and $0 \le P(A_i) \le 1$, i.e. a capability is included in its parent.
2. $P(S) = P(A_1 \cup A_2 \cup ... \cup A_i) = 1$, i.e. a parent capability is the union of is child capabilities and it is quantified as 1.
3. If $A_1, A_2, ... A_i$ have no overlap, then $P(A_1 \cup A_2 \cup ... \cup A_i) = P(A_1) + P(A_2) + ... + P(A_i)$, i.e. the union of the isolated sibling capabilities is equal to their sum.
4. $P(A_1) = P(A_2) = ... = P(A_i)$, i.e. every child capability is assumed to have the same coverage within the parent set.

5. If $P\left(A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_k}\right)$ is not given, then $P\left(A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_k}\right) = P\left(\bigcap_{i=1}^{k} A_i\right)$. Using the example in Figure 8-18, if the overlap between A_1 and A_2 ($P(A_1 \cap A_2)$) is not given, then the general overlap between A_1, A2 and A_3 ($P(A_1 \cap A_2 \cap A_3)$) will be used as the default overlap value of $P(A_1 \cap A_2)$.

Therefore, by applying the set theory, the relationship between two sibling capabilities A_1 and A_2 can be described as:

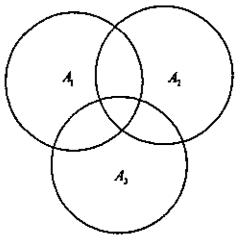


$$P\left(A_1 \cup A_2\right) = P\left(A_1\right) + P\left(A_2\right) - P\left(A_1 \cap A_2\right)$$

Equation 8-1. Relationship between two sibling capabilities.

If and A_1 and A_2 have no knowledge overlap, i.e. they are disjoint sets, then $P(A_1 \cap A_2) = 0$ and thus $P(A_1 \cup A_2) = P(A_1) + P(A_2)$ that matches Axiom 3.

Similarly, the relationships of the three sibling capabilities A_1, A_2 and A_3 can be described as:



$$P\left(A_1 \cup A_2 \cup A_3\right) = P\left(A_1\right) + P\left(A_2\right) + P\left(A_3\right) - P\left(A_1 \cap A_2\right) - P\left(A_1 \cap A_3\right)$$
$$-P\left(A_2 \cap A_3\right) + P\left(A_1 \cap A_2 \cap A_3\right)$$

Equation 8-2. Relationship between three sibling capabilities.

The relationship of sibling capabilities A_1, A_2...A_n can be described as:

$$P\left(\bigcup_{i=1}^{N} A_i\right) = \sum_{i=1}^{N} P(A_i) - \sum_{i_1 < i_2} P\left(A_{i_1} \cap A_{i_2}\right) + \sum_{i_1 < i_2 < i_3} P\left(A_{i_1} \cap A_{i_2} \cap A_{i_3}\right) + ...$$
$$+(-1)^{k+1} \sum_{i_1 < i_2 < ... < i_k} P\left(A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_k}\right) + ...$$
$$+(-1)^{N+1} P\left(A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_N}\right)$$

Equation 8-3. Relationship between sibling capabilities A_1, A_2...A_n.

Based on the equations, the user is only required to give the overlap values between the sibling capabilities. The relationship between a child capability and its parent $P(A_i)$ can be calculated. Consider the example in Figure 8-16, let *C++* be A_1, *Java* be A_2, *VB* be A_3 and *Object-Oriented* be the parent *S*, by using Equation 8-2, their relationship can be described as:

$$\begin{aligned}P(S) = P(C{+}{+} \cup Java \cup VB) = P(C{+}{+}) + P(Java) + P(VB) - P(C{+}{+} \cap Java) \\ -P(Java \cap VB) + P(C{+}{+} \cap Java \cap VB)\end{aligned}$$

Let $x = P(C{+}{+}) = P(VB) = P(Java)$ (by axiom 4) and the overlap between *C++* and *Java* is given as 40%, i.e. $P(C{+}{+} \cap Java) = 0.4x$, the overlap among *C++*, *VB* and *Java* is given as 15%, i.e. $P(C{+}{+} \cap Java \cap VB) = 0.15x$. By Axiom 5, $P(C{+}{+} \cap VB) = P(Java \cap VB) = 0.15x$.

Therefore,

$$P(S) = P(C{+}{+} \cup VB \cup Java) = x + x + x - 0.4x - 0.15x - 0.15x + 0.15x$$

It is also known that $P(S) = P(C{+}{+} \cup VB \cup Java) = 1$ (axiom 2), then

$$\begin{aligned}1 &= 2.45x \\ x &= \frac{1}{2.45} \\ x &= 0.41\end{aligned}$$

The result means that, for example, if an agent knows *C++*, then he knows about 41% of all *Object-Oriented* programming languages.

Both Capability Ontology and Application Ontology are created based on the 'overlap' approach. $P(A_i)$ are calculated immediately after a term is added to the ontology hierarchy by the user. The given knowledge overlaps and $P(A_i)$ will be stored in the database. Using some sample overlap figures and the above formula, the traditional ontology hierarchy given in Figure 8-16 can be enriched as in Figure 8-19.

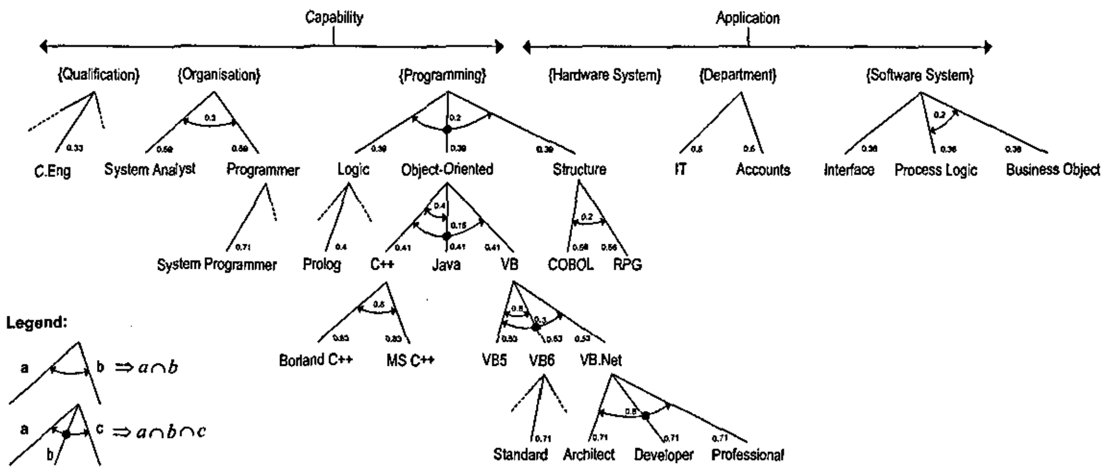


Figure 8-19. Example of ontology hierarchies based on the 'overlap' approach.

8.7.1.4 Optional Capability Parameters

The two compulsory parts may have different importance in a capability specification. For example, a task may require an agent who is familiar with C++ is prefer and is experienced in the development of system interface. An additional weighting parameter can be assigned to each compulsory part to indicate its importance. The weighting parameter for t_c is denoted as w_c and another for t_a is denoted as w_a. To indicate that knowledge of C++ is less important than the experience in system interface, the user can, for example, assign $w_c = 70$ and $w_a = 30$. The total value of w_c and w_a must equal to 100. The default value is 50, implying each has an equal importance to another. If the default value is used, then w_c and w_a do not need to be specified.

To perform a task, multiple capabilities may be required. Each capability in such capability set can have different importance from the others. Therefore another weighting parameter denoted as w_s is assigned. The total value of all the w_s in a capability set must equal to 100. The default value of w_s is $\frac{100}{n}$ where n is the number of capabilities in the capability set.

Thus, a full description of a required capability is denoted as *Capability(t_c,t_a,w_c,w_a,w_s)*. For example, *Capability(C++,Interface,70,30,100)* means that the expected agent knows C++ with the experience in system interface implementation is preferred. The value 100 of w_s implies that there is only one required capability.

8.7.1.5 Beyond Technical Capability

Based on this approach, a capability is not limited to skills or techniques, but can specify a wide range of knowledge, such as authority, as long as such knowledge is captured and represented through the use of hierarchical ontology. An example of agent selection with and without organisation knowledge is given in Figure 8-20.

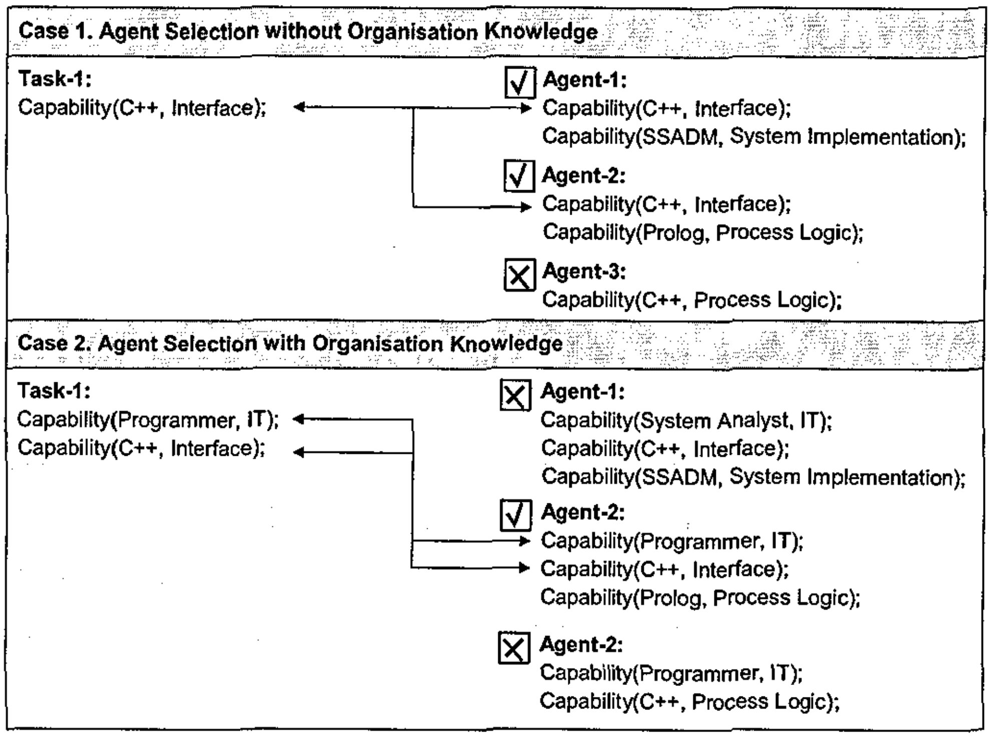

Case 1. Agent Selection without Organisation Knowledge
Task-1:
Capability(C++, Interface);
☑ Agent-1:
Capability(C++, Interface);
Capability(SSADM, System Implementation);
☑ Agent-2:
Capability(C++, Interface);
Capability(Prolog, Process Logic);
☒ Agent-3:
Capability(C++, Process Logic);
Case 2. Agent Selection with Organisation Knowledge
Task-1:
Capability(Programmer, IT);
Capability(C++, Interface);
☒ Agent-1:
Capability(System Analyst, IT);
Capability(C++, Interface);
Capability(SSADM, System Implementation);
☑ Agent-2:
Capability(Programmer, IT);
Capability(C++, Interface);
Capability(Prolog, Process Logic);
☒ Agent-2:
Capability(Programmer, IT);
Capability(C++, Process Logic);

Figure 8-20. Examples of an agent selection with and without organisation knowledge.

In case 1, the agent selection process considers only technical capability issues. Both Agent-1 and Agent-2 are suitable candidates as they both fulfil the capability required of Task-1.

In case 2, agent selection is extended to take organisational knowledge into consideration where *Capability(Programmer, IT)* means that the expected agent is a programmer who works in the IT department. Based on this information, a more suitable agent is identified, resulting in a more appropriate selection.

8.7.1.6 Capability Scheme

The same task in different projects may have different required capabilities. Therefore, the required capabilities in the Model for different situations are specified in different capability schemes. Users can select a suitable scheme for a given task, as different situations can be dealt with appropriately. For example, assumes that the required assessor for a SIL1 system is an *independent person* and SIL2 system is an *independent organisation.* These two different requirements can be saved into two capability schemes. Once the required SIL for current project is identified and given, the suitable required capability can be retrieved from the Model for the compliance checks.

8.7.2 Assessing Goodness of Fit Using Fuzzy Matching

The inappropriate outcomes from traditional approach in assessing GOF show that the different types of relationship between o_a and o_r have to be identified and dealt with appropriately. Chung and Jefferson (1998) identified four different categories in matching conceptual terms. In applying the categories to capability matching, we have:

1. o_a is the same as o_r;
2. o_a is an ancestor of o_r in the hierarchy;
3. o_a is a descendant of o_r in the hierarchy;
4. o_a and o_r are on different branches in the hierarchy;

Additional category in the Compliance Flow framework would be:

5. o_a and o_r are on different hierarchies;

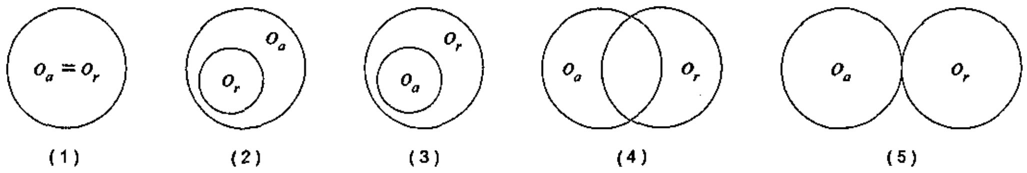


Figure 8-21. Five categories in matching two capability terms.

The five categories are illustrated using Venn diagrams in Figure 8-21.

In category 1, as o_a is the same as o_r, i.e. $o_a = o_r$, it is obvious that $GOF(o_a, o_r) = 1$.

In category 2, o_a includs o_r, i.e. $o_a \supseteq o_r$, implying that the available capability includes the required one, and thus $GOF(o_a, o_r) = 1$.

In category 3, o_a is part of o_r, i.e. $o_a \subseteq o_r$, implying that the agent only partially fulfils the required capability. In this case, the area of o_a partitioning o_r should be the GOF, and thus $GOF(o_a, o_r) = P(o_a)$.

In category 4, o_a and o_r are located on different branches in a hierarchy, i.e. $x \in o_a \cap o_r \Leftrightarrow x \in o_a$ and $x \in o_r$ where x is the overlap element. In this case, the overlap between o_a and o_r should be the GOF, and thus $GOF(o_a, o_r) = P(o_a \cap o_r)$ where $\{x : 0 \text{ p } x \text{ p } 1\} \subseteq P(o_a \cap o_r)$.

In category 5, o_a and o_r are located on different hierarchies, i.e. $x \in o_a \Leftrightarrow x \notin o_r$ or $x \in o_r \Leftrightarrow x \notin o_a$. As every hierarchy represents a particular domain of interest, there should has no knowledge overlap between ontology hierarchies, and thus $GOF(o_a, o_r) = 0$.

For the categories 1, 2 and 5, once the locations of o_a and o_r in the hierarchy is identified, the default GOF values can be given. For the categories 3 and 4, further calculations are required.

For category 3, for example, let *MS C++* is o_a and *Programming* is o_r, then

$$GOF(MSC{+}{+}, Programming) =$$
$$P\Big(Programming \cap P\big(Object\text{-}Oriented\big(C{+}{+}(MSC{+}{+})\big)\big)\Big)$$

as

$$P(MSC{+}{+}) = 0.83$$
$$P\big(C{+}{+}(MSC{+}{+})\big) = 0.83 \times 0.41$$
$$P\Big(Object - Oriented\big(C{+}{+}(MSC{+}{+})\big)\Big) = 0.83 \times 0.41$$
$$P\Big(Programming \cap P\Big(Object - Oriented\big(C{+}{+}(MSC{+}{+})\big)\Big)\Big) = 0.39 \times 0.83 \times 0.41 = 0.13$$

Therefore,

$$GOF(MSC{+}{+}, Programming) = 0.13$$

For category 4, for example, let *Prolog* is o_a and *MSC++* is o_r, then

$$GOF(Prolog, MSC{+}{+}) =$$
$$P\Big(Logic(Prolog) \cap Object\text{-}Oriented\big(C{+}{+}(MSC{+}{+})\big)\Big)$$

where

$$P\big(Logic(Prolog)\big) = 0.4$$
$$P\Big(\Big(Object - Oriented\big(C{+}{+}(MSC{+}{+})\big)\Big)\Big) = 0.83 \times 0.41 = 0.34$$

and

$$P\left(Logic \cap Object-Oriented\right) = 0.2$$

Therefore,

$$GOF\left(Prolog, MSC{+}{+}\right) = 0.4 \times 0.34 \times 0.2 = 0.03$$

Upon applying the overlap approach to the examples in Table 8-1, using the overlap value shown in Figure 8-19 the results are much more reasonable as shown in Table 8-2,

Example	$\mathbf{o_a}$	$\mathbf{o_r}$	$\mathbf{GOF(o_a,o_r)}_{Distance}$	$\mathbf{GOF(o_a,o_r)}_{Overlap}$
1	Java	Java	100	100
2	VB	Java	50	15
3	C++	Java	50	40
4	C++	Object-Oriented	66	41
5	Object-Oriented	C++	66	100
6	MS C++	Programming	0	13
7	Programming	MS C++	0	100

Table 8-2. Examples GOF calculation using overlap approach.

8.7.2.1 Assessing Goodness of Fit between Two Capabilities

As a capability is specified with two compulsory parts, the assessment of GOF between two capabilities involves two matches: the matching of capability and the matching of its application area. The GOF of an available capability *cap₁* against a required capability *cap₂* can be formulated as follow:

$$GOF\left[cap_1\left(t_{c1},t_{a1}\right),cap_2\left(t_{c2},t_{a2},w_{c2},w_{a2,}w_{s2}\right)\right] = \frac{w_{s2}}{100} \times \left[\frac{w_{c2}}{100} \times GOF\left(t_{c1},t_{c2}\right) + \frac{w_{a2}}{100} \times GOF\left(t_{c1},t_{a2}\right)\right]$$

Equation 8-4. Assess GOF of an available capability against a required capability.

Example 6. If a task requires *Capability(C++,ProcessLogic,70,30,100)*, and an agent has *Capability(C++,BusinessObject)*, then the GOF of agent's capability against the required capability can be calculated as:

$$GOF\left[Cap\left(C++,BusinessObject\right),Cap\left(C++,ProcessLogic,70,30,100\right)\right]$$
$$=\frac{100}{100}\times\left[\frac{70}{100}\times GOF\left(C++,C++\right)+\frac{30}{100}\times GOF\left(BusinessObject,ProcessLogic\right)\right]$$

Where

$$GOF\left(C++,C++\right)=1 \text{ by category 1;}$$
$$GOF\left(BusinessObject,ProcessLogic\right)=P\left(BusinessObject\cap ProcessLogic\right)=0.2$$
$$=\frac{100}{100}\times\left(\frac{70}{100}\times 100+\frac{30}{100}\times 0.2\right)$$
$$=76\%$$

Example 7. If a task requires *Capability(C++,ProcessLogic,70,30,100)*, and an agent has *Capability(Prolog,ProcessObject)*, then the GOF of agent's capability against the required capability can be calculated as:

$$GOF\left[Cap\left(C++,ProcessLogic\right),Cap\left(Prolog,ProcessLogic,70,30,100\right)\right]$$
$$=\frac{100}{100}\times\left[\frac{70}{100}\times GOF\left(C++,Prolog\right)+\frac{30}{100}\times GOF\left(ProcessLogic,ProcessLogic\right)\right]$$

Where

$$GOF\left(C++,Prolog\right)=P\left(C++\cap Prolog\right) \text{ (category 4.)}$$
$$P\left(C++\cap Prolog\right)=P\left(Object-Oriented\left(C++\right)\cap Logic\left(Prolog\right)\right)$$

It is known that

$$P\left(Object-Oriented\left(C++\right)\right)=0.41$$
$$P\left(Logic\left(Prolog\right)\right)=0.4$$
$$P\left(Object-Oriented\cap Logic\right)=0.2 \text{ (given by the user)}$$

Therefore,

$$P\left(Object-Oriented\left(C++\right)\cap Logic\left(Prolog\right)\right)=0.41\times 0.4\times 0.2=0.03$$
$$GOF\left(ProcessLogic,ProcessLogic\right)=1 \text{ (Axiom 1)}$$

Therefore,

$$=\frac{100}{100}\times\left[\frac{70}{100}\times 0.033+\frac{30}{100}\times 1\right]$$
$$=32.31\%$$

It is noted that if t_{c1} and t_{c2} are not from the same hierarchy, then the GOF = 0. The reason is that if two capabilities are from two different domains, then they should have no relationship even though they are applied to the same area.

Example 8. If a task requires *Capability(C++,ProcessLogic,70,30,100)*, and an agent has *Capability(CEng,SoftwareSystem)*, then the GOF of agent's capability against the required capability is 0.

8.7.2.2 Assessing Goodness of Fit between Two Capability Sets.

To perform a task sometimes requires multiple capabilities which form a capability set. Similarly, an agent also possesses a set of capabilities. When assessing the GOF of two capability sets, the total GOF of agent capability set against every task capability is assessed. The sum of GOF of every task capability is the finial GOF between the two capability sets, as illustrated in Figure 8-22.

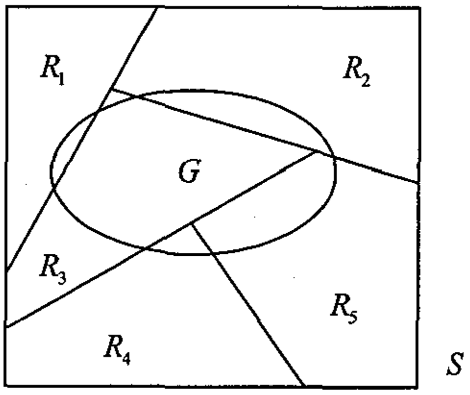


Figure 8-22. GOF of agent capability set against task capability set.

Suppose that S is the universal set that represents a required capability set. The required capabilities in the set are disjoint and each partitions w_s percent (the third option parameter of capability specification) space S. Let $R_1, R_2 ... R_i$ be the required capabilities and G is the knowledge overlap between the two capability sets. Then

$$G = G \cap S = G \cap \left(\bigcup_i R_i \right) = \bigcup_i \left(G \cap R_i \right)$$

Since the R_i are disjoint, the $(G \cap R_i)$ are disjoint too. So,

$$P(G) = \sum_i P(G \cap R_i) \text{ (by axiom 4)}$$

To assess a $P(G \cap R)$, i.e. all the available capabilities against to a single required capability, the sum of GOF of every available capability against the required capability is inappropriate as there may have overlap among the available capabilities.

Consider example 8, suppose the required capability is *Java* and the available capabilities are *C++* and *VB*. It is known that the overlap between *C++* and *Java* is 40% and the overlap among the three languages is 15%. However, the knowledge inside the area $P(C++ \cap Java)$ should include the 15% common knowledge overlap. Therefore the sum, i.e. $P(C++ \cap Java)$ $+P(Java \cap VB) = 0.4 + 0.15 = 55\%$, is not corrected.

The Venn diagram in Figure 8-23 illustrates the scenario given in example 8. The GOF should be the shadow area. Therefore,

$$\begin{aligned} P(G) &= P(C++ \cap Java) + P(Java \cap VB) - P(C++ \cap Java \cap VB) \\ &= 0.4 + 0.15 - 0.15 \\ &= 0.4 \text{ or } 40\% \end{aligned}$$

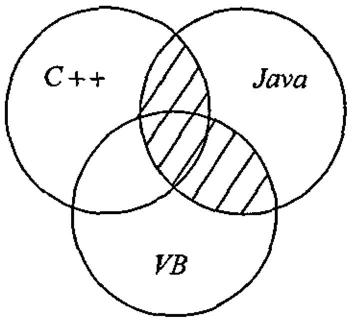


Figure 8-23. Venn diagram for example 8.

Let $A_1, A_2 ... A_k$ are the available capabilities and R is the required capability. P(G) can be formulated as:

$$\begin{aligned} P(G) &= P\left(\bigcup_{i=1}^{N} A_i \cap R\right) \\ &= \sum_{i=1}^{k} (A_i \cap R) - \sum_{i_1 < i_2} (R \cap A_1 \cap A_2) - ... \\ &-(-1)^{k+1} \sum_{i_1 < i_2 < ... < i_k} P(R \cap A_1 \cap A_2 \cap ... \cap A_k) - ... \\ &-(-1)^{N+1} P(R \cap A_1 \cap A_2 \cap ... \cap A_k) \end{aligned}$$

Equation 8-5. The GOF of an available capability set against a required capability.

Let $A_1, A_2 \ldots A_k$ are the available capabilities and $R_1, R_2 \ldots R_i$ are the required capabilities. *P(G)* can be formulised as:

$$P(G) = \sum_i P\left(\bigcup_{i=1}^{N} A_i \cap R_i \right)$$

Equation 8-6. The GOF of an available capability set against a required capability set.

8.7.2.3 Matching Example

In Figure 8-24, Task-2 requires three capabilities: p1, p2 and p3 with the three optional parameters of each capability are given. Agent-1 also has three capabilities: a1, a2 and a3.

Task	*Wc*	*Wa*	*Ws*	GOF(%)	Result	Agent

Task-2:Coding
p1-*Capability(C++,Process Logic);*
p2-Capability(CEng,Software System);
p3-Capability(Programmer,IT);
	
	70
	50
	50
		
		30
		50
		50
			
			30
			50
			20
				Final = 50
				p1 = 30
				p2 = 0
				p3 = 20
					☒	**Agent-1:**
						a1-*Capability*(C++,Business Object);
						a2-Capability(Java,Software System);
						a3-Capability(Programmer,IT)

				Final = 62
				p1 = 42
				p2 = 0
				p3 = 20
					☒	**Agent-2:**
						a1-Capability(VB,Process Logic);
						a2-Capability(Prolog,Process Logic);
						a3-Capability(Programmer,IT);

				Final = 84
				p1 = 21
				p2 = 50
				p3 = 13
					☑	**Agent-3:**
						a1-*Capability(C++,Interface);*
						a2-Capability(VB6,Interface);
						a3-Capability(System Analyst,IT);
						a4-Capability(CEng,Software System);

Figure 8-24. An example of agent selection using fuzzy matching.

Step1: assess $GOF(a1 \ldots a3, p1)$

Capabilities of $a1$ and $a2$ come from the same hierarchy of $p1$, therefore

$$GOF(a1 \ldots a3, p1) = P\big((a1 \cup a2) \cap p1\big)$$
$$= 100 \times \left(0.3 \times \left(\begin{array}{l} 0.7 \times P\big((C{+}{+} \cup Java) \cap C{+}{+}\big) \\ +0.3 \times P\big((\text{BusinessObject} \cup \text{SoftwareSystem}) \cap \text{ProcessLogic}\big) \end{array} \right) \right)$$

Where

$$P\big((C{+}{+} \cup Java) \cap C{+}{+}\big) = 1 \text{ (by category 1)}$$
$$P\big((\text{BusinessO.} \cup \text{SoftwareS.}) \cap \text{ProcessL.}\big) = 1 \text{ (by category 2)}$$

Therefore,

$$GOF(a1...a3, p1) = 100 \times (0.3 \times (0.7 \times 1 + 0.3 \times 1)) = 30$$

As $a3$ come from a different hierarchy from $p1$, it has no overlap between them (by category 5) and therefore can be ignored.

Step2: assess $GOF(a1...a3, p2)$

As a1, a2 and a3 come from different hierarchies from $p2$, $GOF(a1...a3, p2) = 0$, i.e. $P(G) = 0$ (by category 5).

Step3: assess $GOF(a1...a3, p3)$

As $a3$ come from the same hierarchy from $p3$ and $a3 = p3$, $GOF(a1...a3, p3) = 1$, i.e. $P(G) = 20$ (by category 1).

Step4: sum the $GOFs$.

$$GOF(Agent-1, Task-2) = 30 + 0 + 20 = 50$$

The GOFs of Agent-2 and Agent-3 against the task are calculated in the same way. Agent-3 is selected as he/she has the highest GOF value.

8.8 Chapter Summary

This chapter introduces the design and implementation of the Compliance Flow system. A specific hierarchical activity-based process model is developed. The integration of workspace and the ability to capture the knowledge of capability are two major differences from other process models.

A standard is modelled using the same approach. This does not mean that compliance checks can only be applied to workflow systems that adopt a similar process model. The architecture of deploying the compliance agent on different workflow systems is discussed.

Compliance Flow is a web-based system and is developed based on Microsoft's technology. The three-tier system architecture is described. The prototype is developed based upon a four-tier object-oriented implementation architecture, and its benefits are outlined.

A fuzzy agent selection approach is developed, which advocates the use of set theory to assess the GOF. The use of ontology to modelling capability knowledge and an algorithm for the purpose is proposed and discussed. A comparison between the proposed approach and the traditional approach is performed. It is found that the traditional "distance" approach is not suitable for the purpose of capability matching. The proposed approach gives more reasonable results.

Chapter 9

Evaluation and Comparison

"All our science,
measured against reality,
is primitive and childlike –
and yet it is the most precious thing we have."

- Albert Einstein

9.1 Introduction

This chapter provides an evaluation of the Compliance Flow system. Two real-life case studies are performed to evaluate the system and a factitious case is used to evaluation the capability specification and matching approach. The first is the lightguard development in the Assuring Programmable Electronic System (APES) project from ERA Technology Limited. The second is the modification of a safety shutdown system from ABB Limited. The two studies focus on the evaluation of (1) the ability of managing process compliance through compliance checks and (2) the ability of task management based on the proposed framework. The factitious case focuses on the evaluation of the proposed capability specification in terms of the main desired features identified in Chapter 8 and the degree of precision is assessed. Finally, two comparisons between Compliance Flow and other existing systems are given. The first one compares the proposed treatment of compliance management against traditional document based approaches. The second compares Compliance Flow against the flexible workflow systems discussed in Chapter 3 in the context of supporting engineering processes.

This chapter is organised as follow: §9.2 and §9.3 describes the lightguard development and safety shutdown system modification case studies respectively; §9.4 describes the factitious case for the evaluation of capability specification and matching. §9.5 compares the idea of compliance management in Compliance Flow with traditional approaches; §9.6 compares Compliance Flow with other flexible workflow systems; a chapter summary is given in §9.7.

9.2 Case Study 1 – Lightguard Development

The lightguard development project is one of three trail applications in the Assuring Programmable Electronic Systems (APES) project from ERA Technology Limited. The lightguard is originally developed by MTE Limited to comply with BS EN61496, Safety of Machinery – Electro-Sensitive Protective Equipment, and BS EN954, Safety of Machinery – Safety Related Parts of Control Systems. BS EN61496 is a product family standard specifically with requirements for lightguard using active opto-electronic protective devices. BS EN954 is an application standard which provides guidance on the design and assessment of machinery control systems.

The lightguard development process is, however, incompliant with IEC61508. It has been analysed in APES project and a number of correction issues towards IEC61508 compliance are given. A safety plan is proposed by ERA, however, neither the original process nor the corrected one are explicitly given in the publication. For the case study, a simulation of using Compliance Flow to manage the re-organised development tasks was performed. Many of the errors of the development process were identified and Compliance Flow is an effective process management tool.

9.2.1 Project Overview

A lightguard performs a single generic safety function. These sets of infrared beams are used to scan a protected area. If a light beam is blocked, the machine it is guarding will be switch off. The main components of the lightguard and their interconnections are illustrated in Figure 9-1.

In a lightguard, a set of transmitter (Tx) and receiver (Rx) processors transmit and receive data via a set of infrared light beams. Two independent controlling and monitoring channels (Control1 and Control2) are incorporated which control the state of the two outputs signals that connect to the final switching devices (FSD1 and FSD2) in the machinery under control. Once any beams are blocked, Control1 and Control2 will be de-energised and therefore de-energised FSD1 and FSD2. In order to limit common mode failures of the equipment, the two channels make use of two different microcontrollers, an Atmel microcontroller and a PIC microcontroller. A cross check is performed within the controlling and monitoring channels for fault detection purposes. The data from Control2 are shown on a local LED display that is controlled by a diagnostics processor. The programs for the transmitter, receiver, control and diagnostics processors are written in an assembly language.

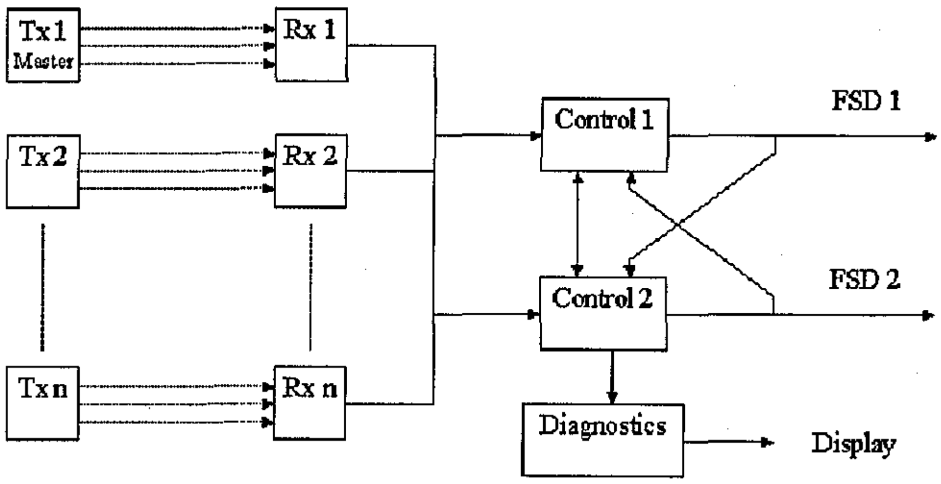


Figure 9-1. Lightguard components and interconnections.

9.2.2 Lightguard Development Process

Table 9-1 lists the high level stages of the lightguard design process and the specification that is produced for each stage. The dependencies between the different stages are identified by their input and output requirements.

Stage ID	**Specifications Produced**	**Dependencies (Required Input Specifications)**
1	Requirements Specification for the Lightguard System	None.
2	Hardware Specification for the Lightguard System	Stage 1 (Requirements Specification for the Lightguard System)
3	Functional Descriptions of individual modules	Stage 1 (Requirements Specification for the Lightguard System) Stage 2 (Hardware Specification for the Lightguard System)
4	Circuit Diagrams, Component Lists etc.	Stage 3 (Functional Descriptions of individual Modules)
5	PCB Layout Diagrams etc.	Stage 4 (Circuit Diagrams, Component Lists etc.)
6	Software Requirements Specification for the Light Guard	Stage 1 (Requirements Specification for the Lightguard System) Stage 2 (Hardware Specification for the Lightguard System)
7	Software Design Specification for the Light Guard	Stage 6 (Software Requirements Specification for the Lightguard)
8	Software Flowcharts	Stage 6 (Software Requirements Specification for the Lightguard) Stage 7 (Software Design Specification for the Light Guard)
9	Software Source Code Listings	Stage 8 (Software Flowcharts)

Table 9-1. Design stages and corresponding input and output specifications.

The lightguard development process was designed to comply with BS EN61496 and BS EN 954, one of the purposes of the case study was to check whether it complies with the IEC61508 application requirements discussed in Chapter 6. The summary of the result is listed in Table 9-2.

Requirement 1: The development shall be carried out in accordance with a defined quality and safety plan.	Not compliant.
Requirement 2: Suitable techniques and measures shall be selected.	Partially compliant.
Requirement 3: The project organisation and allocated responsibilities shall be defined.	Not compliant.
Requirement 4: Configuration management and change control procedures shall be defined.	Partially compliant.
Requirement 5: Design reviews shall be planned and carried out.	Partially compliant.
Requirement 6: Documentation shall be produced.	Partially compliant.
Requirement 7: Hazard analysis and risk assessment shall be carried out.	Partially compliant.
Requirement 8: A safety requirements specification shall be documented.	Unsure
Requirement 9: There shall be clear traceability from requirements through design.	Unsure
Requirement 10: Appropriate programmable electronics architecture shall be selected.	Compliant.
Requirement 11: Appropriate design techniques shall be employed depending on the required safety integrity level.	Partially compliant.
Requirement 12: Test specifications shall be prepared prior to testing.	Not compliant.
Requirement 13: The results of test and analysis activities shall be recorded.	Not compliant.
Requirement 14: The development shall be subjected to independent safety validation and assessment.	Unsure.

Table 9-2. Compliance summary of lightguard development project.

9.2.3 Compliance Management with IEC61508 Requirements

For the case study, the lightguard design process and its relevant information, such as the input (design requirements), the output (design specifications) and the techniques, involved in each stage were input to Compliance Flow. A screen shot of modelling the high level process structure using Compliance Flow is given in Figure 9-2. The simulation demonstrates that Compliance Flow is able to provide intelligent assistance in detecting and managing many of the compliance errors. The findings of using Compliance Flow to deal with such errors are discussed below.

Requirement 1: The development shall be performed in accordance with a defined quality and safety plan.

Error: Identified some procedures are in place, which cover programmable electronics and software development activities carried out generally within the Design and Development stage. However, a document that defines the lightguard development process in full is missing.

Compliance Flow Solution: The use of a workflow system to mange the development process is a solution. As the development process is modelled as a process plan in a workflow system where all the details can be captured during task execution, and relevant reports can be generated as required.

Requirement 2: Suitable techniques and measures shall be selected.

Error: Identified test specifications are not available, and therefore what testing techniques are employed is not clear.

Compliance Flow Solution: Completeness Check can ensure that test specifications are prepared before testing, and Recommendation Check ensure that the recommended techniques are considered for particular tasks. The recommended techniques for each test task can be listed by the system.

Requirement 3: The project organisation and allocated responsibilities shall be defined.

Error: Identified the lightguard project organisation and allocation of responsibilities are not documented explicitly.

Finding: The staff information, such as role and capability, can be maintained in the Organisation

Server. As a task has to be assigned an agent before execution, the project organisation and allocation of responsibility must be defined and will be captured in the process model. In addition, the Capability Check can ensure that the assigned agents have the required authorities and skills to perform their tasks.

Requirement 4: Configuration management and change control procedures shall be defined.

Error: The Change Request Notes (CRN) and Change Notes (CN) are used. A spreadsheet is maintained of all firmware releases in which references are made to the corresponding CN. However, it is not clear that the impact of changes are systematically reviewed and documented.

Compliance Flow Solution: The procedures to deal with a change can be defined as a plan and maintained in Plan Library. Once an update is released, an appropriate plan is called and the required reviews will be performed and documented during the execution.

Requirement 5: Design reviews shall be planned and carried out.

Error: There is an only ongoing process of review during the lightguard development process rather than organised design review meetings. In addition, the formal minutes of these review meetings are not produced.

Compliance Flow Solution: The design tasks, including review meetings, can be modelled in the process plan. The post-condition is a set of minutes that will be produced after a meeting task is completed.

Requirement 6: Documentation shall be produced.

Error: Identified there are no test specifications relating to testing of the system.

Compliance Flow Solution: Compliance Check can ensure the required test specifications are prepared before the testing.

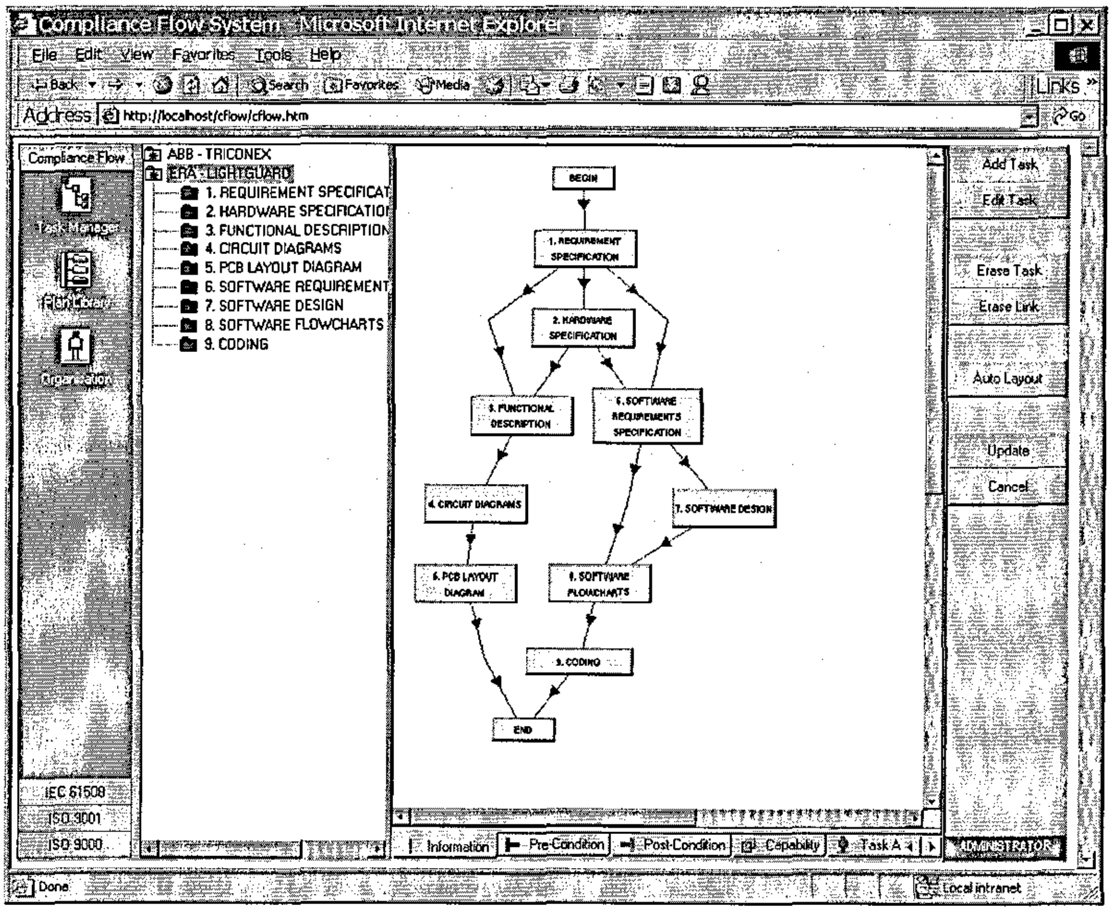


Figure 9-2. A screen shot of the high level process model of a lightguard development process.

Requirement 7: Hazard analysis and risk assessment shall be carried out.

Error: Potential failure modes of the lightguard equipment are identified and recorded in a database. However, the objectives and procedures for these activities are not clearly defined.

Compliance Flow solution: The required objectives will be dealt with by Completeness Check while Correctness Check will ensure that the related activities are defined in a proper sequence.

Requirement 8: A safety requirements specification shall be documented.

Error: No error can be identified even though the software design specification does not provide a complete and unambiguous specification of the safety requirements for the lightguard development. For example, in the software design specification, it is simply stated that "the serial numbers shall be contained in the transmitter and by each independent microprocessor in the receiver".

Compliance Flow Solution: Compliance Flow can provide a little assistance in managing errors that concern with the document content. It is only able to ensure that the required information (sections) is involved in a document by defining the structures of such information in the hierarchical documentation ontology, and which will be defined as the post-conditions of relevant tasks. A document with such contents will be produced after the tasks are completed. However, the document context is unable to be assessed.

Requirement 9: There shall be clear traceability from requirements through design.

Error: Because Compliance Flow is unable to assess the document context, no error can be detected even through the structure of the requirements and design documentation for the lightguard does not exhibit explicit traceability between the safety requirements and the associated implementation.

Compliance Flow Solution: As the development process is captured by the process model and the details of the task executions are recorded in Tracking Server, a limited support can be provided by replaying the implementation process and the decision paths. However, tracking process is far away from what the standard required.

Requirement 11: Appropriate design techniques shall be employed depending on the required safety integrity level.

Error: Identified only timing diagrams and software flow charts are used in the software design. Some highly recommended techniques are not used and required explanations are not provided.

Compliance Flow Solution: The recommended techniques for different SILs are provided, ordered by their importance, by Recommendation Check. Suitable techniques are selected and defined as pre-conditions of particular tasks. Explanations of not using the recommended techniques have to be provided by users.

Requirement 12: Test specifications shall be prepared prior to testing.

Error: Identified a test plan and test specifications are not documented. Functional testing of the integrated software and target hardware are informal.

Compliance Flow Solution: Completeness Check and Correctness Check can ensure the test plan and test specifications are developed in advance of testing. Recommendation Check can ensure every test task is performed with appropriate techniques while Capability Check ensures the task is performed by qualified people.

Requirement 13: The results of test and analysis activities shall be recorded.

Error: Identified software testing is performed informally and only recorded in project notebooks.

Compliance Flow Solution: The test procedures and activities are specified in a process plan with their executions under the control of the workflow engine, the required test results and analysis activities are recorded.

Requirement 14: The development shall be subjected to independent safety validation and assessment.

Error: It is unclear how the requirement for independent validation of the lightguard design is to be handled.

Compliance Flow Solution: As procedures and activities can be specified in a process plan, the validation and assessment activities should be included. The independence of an assessor can be defined as a capability so that suitable assessors can be identified in the agent selection process supported by capability matching. The concept of workspace facilitates the assessors to retrieve the required information effectively.

9.3 Case Study 2 – Safety Shutdown System Modification

A safety shutdown system modification was conducted by ABB limited for one of their clients. The project detail cannot be published because of its disclosure restriction. Unlike the lightguard development, the project is fully complied with IEC61508. The project size is relative small; the physical documentation is within 1500 pages. For the case study, a simulation that uses the reorganised development process is performed. It shows that Compliance Flow not only provides support for process compliance, its framework also facilitates the management of development tasks. A number of suggestions for extension were given by ABB Limited.

9.3.1 Project Overview

Modifications were made to a safety shutdown system during a plant shutdown. The software was upgraded from version 1 to version 2, and minor changes were made to a pair of existing trips. Although the changes are minor, the high integrity nature of the trip system meant that the entire logic within the shutdown system, together with its links to the distributed controlled system (DCS), be re-tested.

During the modification, a fundamental assumption is that the existing system has been comprehensively tested and that the test procedure used are complete and accurate, and that no changes have been made to the DCS that could impact on the safety shutdown system. Therefore, testing of existing trips is reduced to re-validating the existing logic. Only the mappings of tags to I/O points and I/O revalidation tests that include confirming associated DCS functionality are required.

9.3.2 Study Findings

The tasks of the Modification together with its relevant information were used as input to Compliance Flow system for a simulation. The case study was performed together with three ABB staffs, a safety project manager, a safety software manager and a programmer. The findings against the requirements identified in Chapter 4 are outlined and discussed as follow:

Compliance Management

Like the lightguard development project, the process structure differs from the framework proposed by IEC61508. As expected, Compliance Flow dealt with it without any problem. Screen shots of the high level tasks of the modification and the software lifecycle are given in Figure 9-3 and Figure 9-4 respectively. For the purpose of the simulation, a number of required specifications are deliberately left out in the process plan and some other specifications are specified in improper sequence. For example, 'Test Specification' is placed behind the 'Test Report'. All mistakes were successfully detected by Completeness Check and Correctness Check.

The IEC61508 recommended techniques for performing a particular task are successfully retrieved and listed out by Recommendation Check. The highly recommended techniques that have not been specified in the user-defined process were highlighted for attention.

ABB has a competent library which records the capabilities of every technical staff in the context of developing safety related equipments. The competent list is defined based on a proprietary ontology used in ABB and can be used in capability matching. Capability Check succeeded in the assessment of whether an agent can fulfil the required capability. However, as expected, the value of goodness of fits (GOF) retrieved by a capability matching is only meaningful when it is equal 100 which implies a perfect compliance or it is used to compare with another, for example, the agent with GOF(75) should be more suitable than another with GOF(60).

Traceability

Although IEC61508 emphasises process compliance, current best practice can only be assessed based on project documentation. The traceability proposed in Compliance Flow facilitates the identification of project compliance at process level, which is the concern of most projects. Compliance Flow can also illustrate how a document is produced.

Selection of Agent

As the number of staff involved in this project is relatively small, the proposed agent selection by capability matching cannot manifest its full power. However, ABB agrees that the approach is valuable for selecting an agent from a large pool of resource, particularly for companies that adopt matrix organisation.

Flexibility

The original Gantt Chart for the Modification was not available. The process structure used as the input to the system was recreated by the software safety manager based on his memory. A number of mistakes, such as improper task structure, were made as he cannot remember every project details. Such errors were discovered during the evaluation and corrected in the system immediately. Some errors were detected and corrected when simulation of the process had started. Users found it is easy to update the process plan. The interleaving between the process build-time and run-time was successfully demonstrated.

Process Reuse

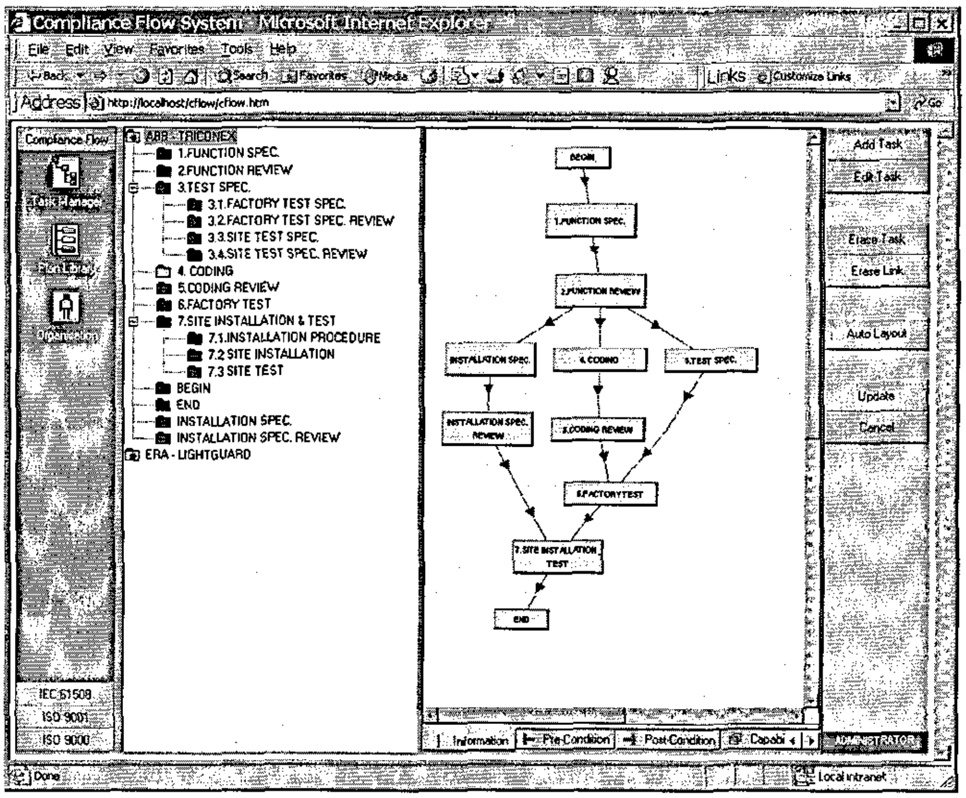
A number of Plans were created during the simulation of the Modification project. Such sub-Plans were used to construct a new process plan for a fictitious project with certain similarity. It shows that the use of Plans can effectively shorten and ease the planning process.

Management at Different Level

The hierarchical presentation of project structure enables that users with different responsibilities in a project have an overview of the progress at a glance while concentrating on their own tasks. The use of workspace facilitates the senior staffs to monitor and access the information relevant to tasks performing by their teams.

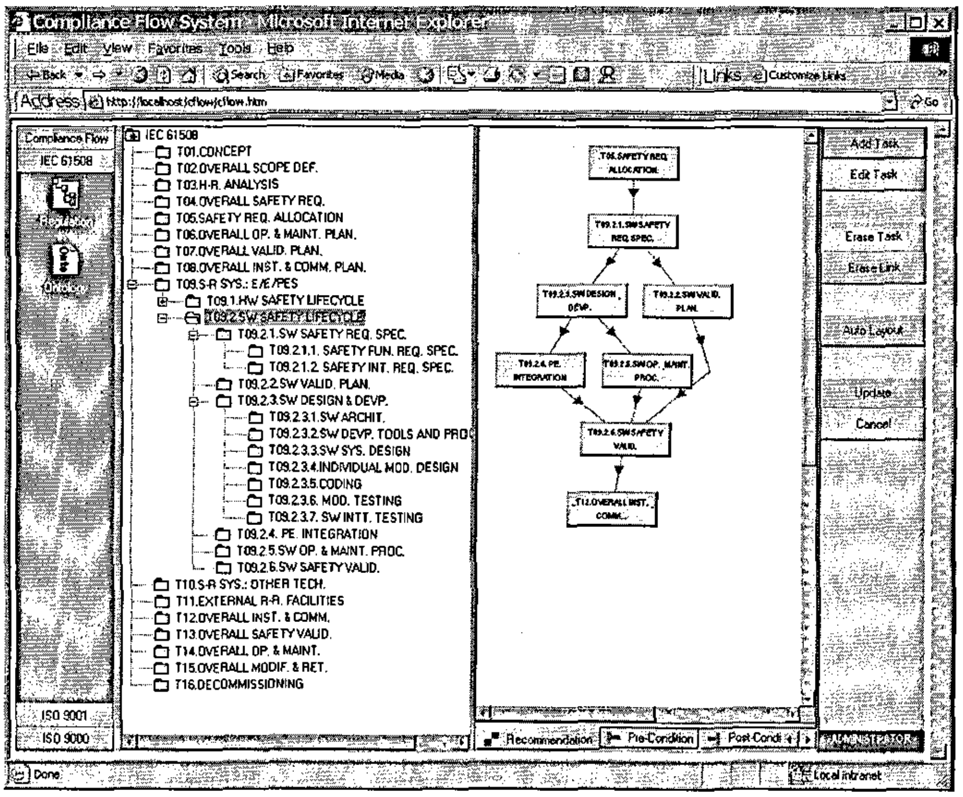
Process and Information Management

The use of workspace demonstrated the ability to automatically transfer information from a task to another according to the user-defined process. Information Check has further transferred the information to the subsequence tasks for which the flow of information is not specified in the process plan but may be needed. Although not all the information transferred by Information Check will be used during the task execution, it demonstrates that Information Check can effectively reduce the impact from such mistakes where the task interdependence is inefficiently captured.



- ABB - TRICONEX
  - 1.FUNCTION SPEC.
  - 2.FUNCTION REVIEW
  - 3.TEST SPEC.
    - 3.1.FACTORY TEST SPEC.
    - 3.2.FACTORY TEST SPEC. REVIEW
    - 3.3.SITE TEST SPEC.
    - 3.4.SITE TEST SPEC. REVIEW
  - 4. CODING
  - 5.CODING REVIEW
  - 6.FACTORY TEST
  - 7.SITE INSTALLATION & TEST
    - 7.1.INSTALLATION PROCEDURE
    - 7.2 SITE INSTALLATION
    - 7.3 SITE TEST
  - BEGIN
  - END
  - INSTALLATION SPEC.
  - INSTALLATION SPEC. REVIEW
- ERA - LIGHTGUARD

Figure 9-3. High level process model of the modification project.



- IEC 61508
  - T01.CONCEPT
  - T02.OVERALL SCOPE DEF.
  - T03.H-R. ANALYSIS
  - T04.OVERALL SAFETY REQ.
  - T05.SAFETY REQ. ALLOCATION
  - T06.OVERALL OP. & MAINT. PLAN.
  - T07.OVERALL VALID. PLAN.
  - T08.OVERALL INST. & COMM. PLAN.
  - T09.S-R SYS.: E/E/PES
    - T09.1.HW SAFETY LIFECYCLE
    - T09.2.SW SAFETY LIFECYCLE
      - T09.2.1.SW SAFETY REQ. SPEC.
        - T09.2.1.1. SAFETY FUN. REQ. SPEC.
        - T09.2.1.2. SAFETY INT. REQ. SPEC.
      - T09.2.2.SW VALID. PLAN.
      - T09.2.3.SW DESIGN & DEVP.
        - T09.2.3.1.SW ARCHIT.
        - T09.2.3.2.SW DEVP. TOOLS AND PROC
        - T09.2.3.3.SW SYS. DESIGN
        - T09.2.3.4.INDIVIDUAL MOD. DESIGN
        - T09.2.3.5.CODING
        - T09.2.3.6. MOD. TESTING
        - T09.2.3.7. SW INTT. TESTING
      - T09.2.4. PE. INTEGRATION
      - T09.2.5.SW OP. & MAINT. PROC.
      - T09.2.6.SW SAFETY VALID.
  - T10.S-R SYS.: OTHER TECH.
  - T11.EXTERNAL R-R. FACILITIES
  - T12.OVERALL INST. & COMM.
  - T13.OVERALL SAFETY VALID.
  - T14.OVERALL OP. & MAINT.
  - T15.OVERALL MODIF. & RET.
  - T16.DECOMMISSIONING

Figure 9-4. High level process model of software lifecycle of IEC61509.

9.3.3 Recommendations from ABB Limited

ABB is impressed by Compliance Flow. They are encouraged by the demonstration and give following summaries:

> *"Compliance Flow has potential! We will definitively buy it!"-Mr. John Walkington*

> *"It is intelligent! Many of the errors can be detected by the system...Capability matching suits our competent library...facilitates the selection of suitable staffs for tasks." - Mr. Paul Lucas*

In addition, a number of recommendations are given to further improve Compliance Flow for practical use.

Enriched Standard Model

Standards, such as IEC61508, are generally sizeable. It is very difficult for a person to remember every detail, particularly for the projects for which multiple standards are applied. Thus, a rich standard process model that provides sufficient information to assist users to understand particular situation in the context of standard compliance is necessary. For example, when a specification is identified as being wrongly placed, compliance agent should give further details of the specification, such as its objectives and issues of concern, rather than just points out its possible locations in the user-defined process plan.

Using Wizard to Deal with Complex Operations

Pointing out the identified errors and providing relevant information are not enough as most of people may be still vague about the detail steps of correction procedures. For these situations, wizards can be used to guide the user step by step to perform some complex operations. Wizards can be widely used in the Compliance Flow framework, which includes (1) fixing errors identified by compliance agent, (2) selecting appropriate Plans for a task, (3) assisting process specification, (4) retrieving track records of task execution, (5) specifying capability and (6) selecting task agent.

Security and Authority Control

Security and authority are important issues in safety product development. Although capability matching is able to ensure the tasks are assigned to appropriate staff, to identify that the tasks are actually performed by the assigned staff is necessary. Thus, some advanced identification methods, such as fingerprint identification, could be used in Compliance Flow system.

Integration with Other Applications

Besides the task management system, implementing an engineering project also requires other types of supporting tools, such as risk assessment systems. The ability to work together with other types of systems is also critical. For example, workspaces can be integrated with a document management system to provide versioning control.

9.4 Case Study 3 – Capability Specification and Matching

A factitious case was specifically developed and performed to evaluate capability specification and matching in five aspects.

In terms of the capability specification:

1. Preciseness: The capability knowledge should be modelled to provide a precise capability specification. The entities in the model are quantitatively related to facilitate fuzzy matching.
2. Expressiveness: The method of capability specification is expressive enough to represent not only the technical capability, but also a wide range of knowledge such as qualification, role and authority, which are related for assessing suitability of an agent in performing a task.
3. Comprehensiveness: A user can understand the capability representation and specification statement, and the system is able to process it.
4. Ease of use: Every capability description should not only be easy to read and understand, but also easily defined by the user.

In terms of the capability matching:

5. Accuracy: The proposed fuzzy matching algorithm should be able to deliver an acceptable result, particularly when a perfect match between the required capabilities and available capabilities is not possible.

9.4.1 Study Overview

Four research and MSc students were invited to participate in the case study. They all have programming experience and some system development knowledge. The general concept of capability specification and matching and their applications in terms of workflow management were described to them before the case study. The case study includes four phases. The evaluation forms are attached as an appendix of this thesis.

Phase 1

The first phase evaluates the difficulty of creating ontology hierarchies and assessing conceptual overlaps between different terms. This phase has two steps. First, a description of a software company with the focus on the development department is given to the participants. They are required to construct the capability and application ontologies, based on the given description. Second, the participants are required to assign the knowledge overlap values between the concepts of interest in these ontologies.

Phase 2

The second phase evaluates the difficulty of interpreting the formal capability representations. Two sets of capability and application ontologies, one with knowledge overlap and another without, which are created based on the given scenario, are given as the standard ontologies to participants for phases 2, 3 and 4. The reason for not using the ontologies created by the participants is that they are different, both in their structures and the overlap figures, which may lead to inconsistent results in the latter phases. In the second phase, a number of formally represented capability specifications are given. The participants are required to describe the meaning of these formal specifications, using the given ontologies, in words.

Phase 3

The third phase evaluates the difficulty of specifying capabilities. The general descriptions of a number of capabilities, including required and available capabilities, are given, and the participants are required to give the formal capability specifications accordingly.

Phase 4

The final phase evaluates the accuracy of the proposed fuzzy matching algorithm. The participants are given capability specifications of the three tasks and eight agents where a perfect match does not exist between them. The participants are required to select the most suitable agent for each task. The results produced are used to compare with the system's choice.

9.4.2 Study Findings

The results show that the capability scenario is effectively expressed by the ontologies. The proposed formal capability representation is easy to learn, understand and use. Most importantly, the fuzzy matching algorithm is able to deliver an accurate result.

Phase 1

Suitable ontology hierarchies can be constructed by every participant in minutes, though the ontology structures are different. A sample is given in Figure 9-5 where (a) integrates Role and Department into a single hierarchy while (b) treats them separately. However, in general, (b) is preferred as each hierarchy is more specific, and more accurate result will be generated by the proposed algorithm.

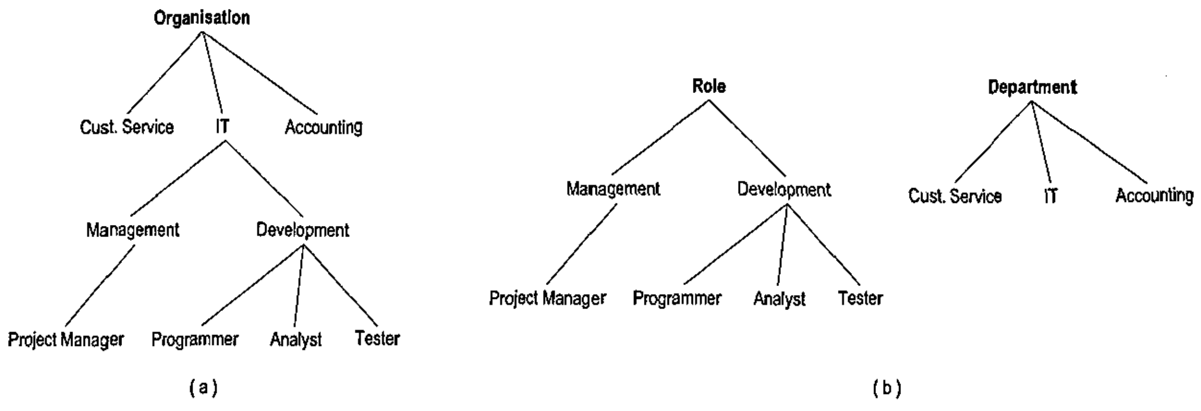


Figure 9-5. Participants construct different ontology hierarchies.

To assess the knowledge overlap between two concepts of interest is the most difficult in the whole evaluation. Some figures given by the participants vary significantly. After a discussion with each participant, the reasons can be identified as either due to their different perspectives on knowledge overlap or the lack of a complete consideration. Using Java and C++ as an example, the figures given by the four participants are shown in Figure 9-6. The knowledge overlaps between the two languages should be twofold: the common programming statements, like If...Then...Else, and the use of object-oriented (O-O) concept. Participant-1 and Participant-2 consider all parts while Participant-3 and Participant-4 only take one part into consideration, and therefore lead to the significant different overlap figures.

Participant-3 and participant-4 have the same limited perspective and gave figures that are quite similar. Participant-3 only considers the statement similarity between two languages and gives an overlapping value 10. He agrees that the O-O concept should be considered in the assessment but was missed out. Participant-4 insists that the similarity of the programming statements is enough to represent the overlapping between two languages. He gave a value of 20.

Though Participant-1 and Participant-2 were thinking similarly, they gave very different subjective figures. However, through a group discussion, a capability and an application ontology with figures accepted by the participants were constructed in minutes.

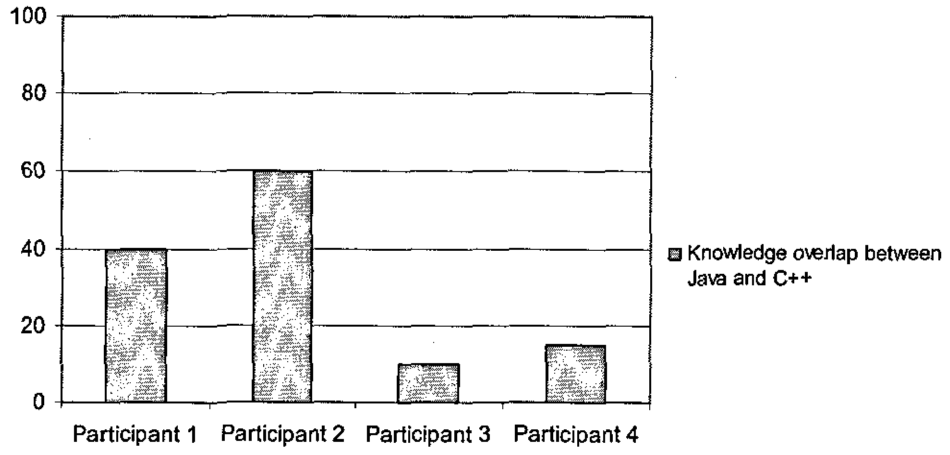


Figure 9-6. A sample knowledge overlap given by the paticipants.

Therefore, it is recommended that a guideline of assessing knowledge overlap for particular domain is necessary and need to be understood and followed by users otherwise they may not be able to specify capability properly.

The expressiveness of ontology hierarchy is also demonstrated. It is agreed by all participants that the scenario can be modelled by the ontologies in the proposed hierarchy structure and the relationship between the terms in the ontology hierarchy can be quantified based on their understandings.

Phase 2

In the second phase, all the participants gave the correct descriptions of the given capability specifications instantly. This demonstrates the the formal capability specification is easy to understand.

Phase 3

In the third phase, all the participants were able to correctly specify the given capability description, though different weights given. Using question 5 as an example, in which 'chartered' qualification is the most important requirement.

Question 5: A task requires a Borland C++ programmer. The programmer must be a chartered software engineer and is experienced in developing safety system. The programming will be performed under Windows environment.

1. Capability(Programmer, IT)
2. Capability(C.Eng, Safety)
3. Capability(C++, Windows)

All the participants can construct the above capability set based on the description. Although different weights were assigned to the capabilities by the participants as shown in Figure 9-7, the importance of qualification is emphasised. The variations only slightly affect the GOF values and the most suitable agent is ranked on the top of the list in all the cases.

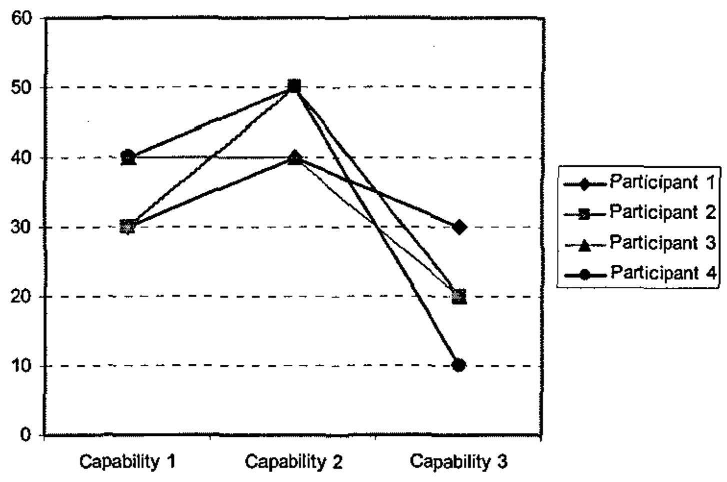


Figure 9-7. Different weighting points given by the participants.

The results indicate that the proposed capability specification and representation approach is comprehensive and easy to use as the participants can put it into practice with a quick learn.

Phase 4

In the fourth phase, the results generated using the proposed fuzzy matching algorithm and the selections made by the participants are exactly the same, indicating its ability to deliver a human-alike decision. All participants realise that endowing ontology hierarchies with knowledge overlaps facilitates understanding of a capability specification and matching, particularly when perfect matches do not exist.

9.5 Process-based Vs Document-based

Much of current research regarding compliance management, such as the work by Emmerich et al. (1998), adopt a document-based approach in which the development processes are implicitly represented in the product. The compliance is treated as a problem that is closely related to inconsistency management in specification (Easterbrook et al., 1994; Finkelstein et al., 1994). Such an approach uses a document schema specification to elaborate and formalise the definitions of document structure suggested in the standard so that properties can be checked against them. Appropriate checks will be triggered when events occur on documentation during the development process. This approach can ascertain that the expected qualified document is produced, which matches current quality control practices where the compliance checks are performed at the end of development stages by individual assessors. However, it lacks process management ability that proactively prevents unqualified products resulting from a wrongly planned process, which is an essential requirement for standards like IEC61508. As Moore indicated:

> *"Some companies in our industry claim to have IEC61508 compliant products. In fact they have only had an assessment done on a single product, not on their company's processes to design and produce that product. This is a severe shortcut, and certainly not in keeping with the intention of the standard." - (Moore, 2002)*

The following are two key benefits in terms of compliance management that Compliance Flow can provide over other document-based support systems.

Process Level Compliance Management

Compliance Flow is able to provide support for compliance management at process level. Like IEC61508, many standards emphasise that compliance should be managed at the process level as a proper process is more likely to produce a quality product. However, for systems that provide support at documentation level, a key challenge is that it is difficult to prove that a document is produced through a correct process. This is tackled in Compliance Flow as a result of the management of the development process.

Pro-active Error Identification

Compliance Flow is able to identify errors before the execution. For many current solutions, errors are identified when the documents are produced and tasks are performed. There significant resources are devoted to managing standard compliance. However, the earlier the errors can be identified, the cheaper will be the development cost and the proper process will result a higher quality product.

9.6 System Comparison

Compliance Flow is designed to provide intelligent support for engineering processes, for which an enriched process model and flexibility of task management are essential. The following is a comparison of Compliance Flow with the adaptive workflow systems discussed in Chapter 3, with the focus on their flexibility features.

9.6.1 Compliance Flow Vs InConcert

The two major features provided by InConcert are: (1) the interleaving between process build-time and run-time, and (2) the use of workspace to enable the sharing of documents. The interleaving between process build-time and run-time is fully supported in Compliance Flow.

The use of workspace in the two systems, however, is different. In InConcert, the documents inside a workspace are linked to the relevant tasks with no relationship to the task dependences. In Compliance Flow, a document can be linked to a task as a pre- or post-condition. A document can be an independent object inside a workspace. A document will be transferred to another task automatically once it is available if it has been defined as a pre-condition of another that task. The conceptual difference in the use of workspace is that InConcert sees a document as a resource required for a job and needs to be shared; Compliance Flow emphasises information management where the required documents will be automatically collected into the workspace of the related tasks, which is more powerful than the workspace concept in InConcert.

9.6.2 Compliance Flow Vs TeamWare

The strength of TeamWare is its support for collaborative planning which allows individuals to "plan their own parts". This important feature for supporting an engineering process is supported in Compliance Flow where the superior raises only the requirements of tasks for which the technical staff have to detail, plan and perform themselves. Each member of staff has a collection of his own library of organisational sub-process. A process plan can be delivered on-the-fly by individuals working on their parts.

Collaborative planning is also supported in Compliance Flow. High level tasks are specified by senior staff and dispatched to their teams. Team members can then further specify the assigned tasks, dispatch them to other staff, or directly perform them. This planning-dispatching process can be performed until the required details of tasks are reached.

Past experience can be captured and maintained in the Plan Library. A Plan represents a possible method of achieving a task at a particular level of abstraction. Unlike TeamWare where the process templates are represented as lists, in Compliance Flow they are represented as task hierarchies.

9.6.3 Compliance Flow Vs TBPM

TBPM's process model is designed to support scale-up processes where parts of a process specification can only be specified and performed when the results of particular tasks are available. Process plans are selected to enable the system to adapt to specific needs. In addition, capability matching is proposed for agent selection.

A task in Compliance Flow can be started when its pre-conditions are fulfilled and pre-tasks are completed. With the ability of interleaving process built-time and process run-time, a scale-up processes can be handled without any problem. The hierarchically organised process plans are also used to increase system adaptability where a suitable Plan to correct a process at different level of abstraction can be easily identified.

Capability matching is extended in Compliance Flow by adopting a flexible scheme approach. It is limited in TBPM as both task and agent can only assign with one set of technical capability where each capability has the same importance. In Compliance Flow, a scheme can be applied to (1) allow each capability in a capability set has different importance and (2) enable a task has multiple capability sets where each is used to deal with a particular situation. In addition, capability in Compliance Flow is extended to refer to not only technical capability but also the organisational knowledge and authority.

9.6.4 Compliance Flow Vs Agent Enhanced Workflow

The use of agents to handle internal and external process exceptions in a traditional workflow system is demonstrated in Agent Enhanced Workflow. Such exceptions cannot be completely perceived and therefore difficult to model. The beauty of using agent is that no or minimum amendment is needed to the workflow system.

A similar concept is used in Compliance Flow to manage process compliance where a compliance agent called Inspector is responsible for checking the compliance of the user-defined process by matching it with a standard process model. The Inspector and the standard process model can be seen as separate components from a workflow system. This architecture enables the deployment of compliance management on different workflow systems with minimum amendment.

9.6.5 Comparison Summary

Compliance Flow has all the flexible features provided in the four adaptive workflow systems with significant improvements. More importantly, the ability of ensuring the compliance of a user-defined process with a standard is a novel feature that current workflow systems do not support. The key features provided in Compliance Flow and the four adaptive workflow systems discussed above is summarised in Table 9-3.

Features	**Compliance Flow**	**InConcert**	**TeamWare**	**TBPM**	**AEW**
Process Modelling					
Capability Information	X			X	
Information Object	X	X		X	
Task Decomposition	X	X	X	X	
Visual Modelling Language	X	X	X		X
Task Management					
Agent Selection by Capability Matching	X			X	
Collaborative Planning	X	X			
Information Management	X			X	
Interleaving Between Process build-time and run-time	X	X	X	X	
Meta-process (Compliance) Management	X				
Partial Definition	X	X	X	X	
Process Template (Plan)	X	X	X	X	
Shared Workspace	X	X			
Use of Software Agent	X				X

Table 9-3. Summary of feature comparison.

9.7 Chapter Summary

This chapter evaluated the Compliance Flow system. Three case studies were performed. The original lightguard development project does not comply with IEC61508 standard as it does not fulfil all the requirements. Compliance Flow demonstrated its ability in detecting errors and managing a standard complied project. The safety shut down system modification project was design and implemented using the guidance of IEC61508. Its process is inputted into Compliance Flow with a number of fictitious errors. The case study shows such errors are successfully detected and managed by the system. Insights gained from the case study are discussed. Finally, the case study of capability specification and matching shows that the proposed capability specification approach is easy to learn, understand and use, and the fuzzy matching algorithm can perform precise matching and deliver a human-alike decision.

From the compliance management perspective, Compliance Flow provides process level compliance management, pro-active error identification and document production control. These three features are critical in compliance management, but are not supported by current systems. From the workflow management perspective, Compliance Flow is able to support all the flexible features provided in the four adaptive WfMS discussed before. The ability to check process compliance against a standard is a innovation in workflow management.

Chapter 10

Conclusions and Future Work

"Starting a PhD is one thing,
finishing it is another."

- Paul Chung

10.1 Introduction

This chapter gives a brief review of this thesis, summarises the achievements and points out some directions for future work. It is organised as §10.2 gives a review of this thesis; §10.3 summarises the major contributions; §10.4 outlines the limitations of this thesis and some future directions.

10.2 Thesis Review

The research question this thesis chose to explore was:

How is it possible to provide intelligent support for the management of dynamic engineering processes, with the focus of ensuring that their specification and performance are compliant with particular industry standards?

Currently, a large amount of time of engineers, managers and quality assurance teams is occupied with tracking and managing the compliance of projects to ensure product acceptability or safety. In this context, process compliance means that there is a clear description of the design stages and, at each stage, the inputs to that stage are fully and unambiguously defined, and all the objectives and requirements of the standard are met. If a workflow system has compliance management ability, then it not only shortens the development time and reduces cost, but also improves the quality of the process and product. Thus, a solution to this problem is important to industry.

To answer the question, current workflow technologies were studied first, with the focus on the technologies that enable workflow systems to adapt to a dynamic environment. Han's conceptual framework, which categories the adaptations of workflow systems to changes into five levels, was discussed. In addition, a survey of five adaptive workflow systems was carried out.

Product development processes were studied in order to understand engineering processes, and their characteristics were identified. Engineering processes differ from general business processes, because they are highly technical, dynamic, ad-hoc, collaborative and involve a vast amount of information interchange of which current workflow systems lack the ability to support. A number of requirements of a system which is to succeed in supporting engineering processes were identified and discussed, including (1) compliance management, (2) traceability, (3), selection of agent, (4) flexibility, (5) common process, (6) management at different levels, and (7) process and information management, in which flexibility is the most important requirement while the ability of compliance management becomes the major challenge of this project.

A novel framework was proposed which provides a platform to enable the integration of a number of enabling technologies to deal with more complex and flexible engineering processes.

The requirements identified are dealt with in the following ways:

- Compliance with Standards: A Compliance Agent performs a number of compliance checks between the Model of Standards and the user-defined processes during process build and run time. As a result, compliance errors are identified during process planning, and tasks with compliance problems are prohibited from execution during run time.
- Traceability: A Tracking Server records decision rationale so that the implementation and decision paths can be traced.
- Selection of Agent: An Organisation Server provides a fuzzy capability matching mechanism for this purpose.
- Flexibility: The interleaving between process build and run time allows parts of a process to be executed while other parts are being refined.
- Common Process: A Plan Library provides support in the form of pre-specified process structure that can be assembled and adapted to suit the specific needs of the current situation.
- Process and Information Management: The process management facilities together with the provision of workspaces provide a collaboration environment for engineering processes. The information will be transferred to the relevant tasks once they are ready. The personal 'Bag' can effectively maintain the necessary privacy for each agent.

To support this framework, a novel workflow model that captures capability knowledge and integrates the concept of CSCW is proposed. A new capability matching approach which utilises a fuzzy matching algorithm was developed to enhance the agent selection procedure. The use of workspace not only allows users to collaborate when a process is running but also provides information management support where the required documents will be automatically collected into the workspace of the related tasks. This is more powerful than the workspace concepts used in other workflow systems.

The proposed solution to the compliance management problem is to use a compliance agent to perform a number of compliance checks between a standard model that captures the required knowledge of a standard, and the user-defined process during both process specification and execution to identify and resolve errors. To do so, the first step is to investigate standard modelling which is used to capture the knowledge of a standard into a standard model for further retrieval in performing compliance checks.

A standard modelling approach was proposed, which is capable of capturing four important aspects of a standard in terms of workflow management: (1) the proposed task framework, (2) the requirements and the deliverables of every task in the framework, (3) the recommended techniques, measures, tools or methods to perform a task, and (4) the required capability of a task agent. The approach has been successfully used in modelling IEC61508.

Compliance management is broken down into six separate issues and tackled appropriately: (1) the required tasks are included in the user-defined process; (2) they are performed in the correct sequence (3) with sufficient information (4) using suitable techniques (5) by qualified persons; and finally (6) the required documents are delivered. Accordingly, a number of compliance checks were developed.

During the process build-time, while the Completeness Check ensures the required activities are included in a user-defined process, the Correctness Check ensures these activities are performed in the recommended sequence. On the other hand, the Capability Check makes certain that the task agents possess the required capability for performing their tasks and the Recommendation Check reminds task agents of the techniques or skills which are recommended to be used to perform their tasks. In addition, the Planning Assistance assists process specification by providing possible required information, such as required capability for tasks. Furthermore, when a task starts, the compliance agent will check its compliance. If the task is not fully compliant with the selected standard, it will be frozen until the identified errors are resolved or an explanation is given. During process run-time, the documents created during the process will be transferred by the Information Navigation to these tasks which may require them for their execution. Finally, the Cross Referencing function allows a user to refer a user-defined task to a standard or vice versa.

A system prototype was developed to evaluate the proposed ideas. Three case studies were performed. The first is the lightguard development from ERA technology. The second is the modification of a safety shutdown system from ABB limited and the third is capability matching evaluation. The original lightguard development project does not comply with the IEC61508 standard as it does not fulfil all the requirements. Compliance Flow demonstrated its ability in detecting errors and managing a standard compliant project. The safety shut down system modification project was designed and implemented using the guidance of IEC61508. Its process was input into the prototype with a number of fictitious errors. The case study shows such errors are successfully detected and managed by the system. The evaluation of capability matching shows that the proposed capability representation is easy to understand and use, and the matching algorithm is able to deliver a human-alike decision in selecting agents for tasks.

Two comparisons between Compliance Flow and other existing systems are also given. The first compares the proposed treatment for compliance management against traditional document based approaches. The second compares Compliance Flow against the flexible workflow systems discussed in Chapter 3 in the context of supporting engineering processes. From the compliance management perspective, Compliance Flow provides process level compliance management, pro-active error identification and document production control. These three features are critical in compliance management, but they are not supported by current systems. From the workflow management perspective, Compliance Flow is able to support all the flexible features provided in the four adaptive workflow systems discussed before. The ability to check process compliance against a standard is an innovation in workflow management.

10.3 Summary of Contributions

The contributions of this thesis are:

- Through the application of workflow management to engineering processes, it has contributed to an enhanced understanding of "adaptive workflow technology".
- It contributes to the compliance domain by empowering the compliance management at process level where errors are detected and prevented in advance of process execution.
- The novel feature of process compliance management creates a new research direction in the workflow community, and its application in managing project compliance indicates the importance of this meta-process control capability.
- A new approach of using set theory to tackle the problem of traditional "distance" approach to assess goodness of fit (GOF) between two concepts of interest.
- A new approach of integrating the concept of CSCW with workflow management is proposed, allowing users to collaborate while the process is running.
- A new approach of agent selection based on fuzzy capability matching.

- A novel process model that further captures capability knowledge and enables that information objects are linked to their related tasks.
- The proposed framework gives a platform where a variety of technologies can be used to increase the required adaptation of a workflow management system in supporting dynamic processes.

10.4 Limitations and Future Directions

The following issues are currently being addressed, or should be addressed in future work:

Handling multiple standards

In this thesis, IEC61508 is used as example to evaluate (1) the proposed approach to modelling a standard and (2) the compliance checks for managing process compliance. The use of one standard is insufficient. Instead, evaluation of the system can be extended to include different standards as there are differences between them. For example, the European Space Agency (ESA) PSS-05 (Mazza et al, 1994) standard for software development does not prescribe a particular lifecycle model. It lists almost 200 practices which are comparatively ambiguous and need to be interpreted and integrated to form a standard model. A typical practice is:

UR04 – For incremental delivery, each user requirement shall include a measure of priority so that the developer can decide the production schedule.

This practice can be interpreted and modelled as: (1) the concerned task: incremental delivery, (2) its post-condition: a measure of priority, and (3) a consequential task: decide the production schedule.

The created model may not accurately represent the original standard due to misinterpretation or ambiguity in the standard. On other hand, the modelling framework may not be appropriate for particular standard. In this case, the process model and compliance checks have to be extended, which may lead to another research project.

Extensions to manage compliance at document level

The proposed approach is designed to ensure that a product is developed through a standard complied process. However, a qualified document is not a guaranteed outcome from a proper development process. Therefore, it is a need to provide support to assess the document context as other compliance management tools do. There are two possible approaches. The first is to integrate Compliance Flow with a current document based compliance management system. However, significant redundancy will exist as a result of the overlapping between these two approaches. The second approach is to extend current process model to capture the required information at documentation level. Further checking of the documents against the model can then be carried out. Simply combining the proposed process model with the document management model used in current systems is likely to be impractical. A more sophisticated modelling and reasoning approach may be required.

Extensions to comply with workflow standards

Compliance Flow is a research prototype and will to be further developed to conform to workflow standards introduced in Chapter 2. Compliance with workflow standards is important and benefits Compliance Flow in two ways: (1) it facilitates the integration and the communication with other WfMS, and (2) it enables the deployment of compliance agents in other WfMS.

A workflow application interface (WAPI) that is complied with WfMC's workflow reference model should be developed in order to allowed third party's tools to integrate into Compliance Flow framework. However, as the proposed process model concerns extra aspects of workflow management, to access such information through the WAPI can be a big problem.

The deployment of a compliance agent is discussed in Chapter 8. The best solution is to use middleware to convert the process information into a file in XPDL format and then perform the compliance checks upon that file. Similarly, the use of XPDL to represent the required information for compliance checks needs to be investigated.

Change Management

For task management in particular engineering domains, such as construction, the changes cannot be directly made to the original plan. Suitable change procedures, such as change identification and evaluation, must be performed to study the current situation. The use of the Tracking Server to record the process planning and execution activities is insufficient. The causes of changes and the decision rationale must be recorded.

One possible solution is to define the change procedures as process plans. When a change has occurred, an appropriate process plan will be instantiated and performed so that the change will be under control. However, identifying the related process plans and generating a solution based on past experiences are ongoing research topics.

Knowledge Representation

The proposed knowledge modelling in which the knowledge hierarchies are endowed with overlapping information between the concepts captured is preliminary evaluated. The result shows that the approach contributes to assessing the GOF between two terms, in the area of capability matching, is better than the traditional approaches. This modelling approach has the potential to be applied in other areas, such as retrieval of images using descriptors (Smeaton and Quigley, 1996). To do so, larger scale tests should be carried out in terms of the type of hierarchy.

Agent selection

The proposed capability specification and matching using fuzzy matching algorithm for agent selection is based on an assumption that one task is performed by one person or a team. It is unable to deal with the situations where, for example, multiple agents are dynamically selected from a resource pool to collaborate to perform a task.

One possible solution is to merge the capabilities of multiple agents into a capability set and match it with the required capabilities. However, the large number of merged capabilities may leads to an unacceptable matching time in a selection process and the degree of accuracy is still unknown. Therefore, further research and evaluation in this context are required.

10.5 Overall Conclusion

An advanced solution to the engineering process management is proposed in this project. The requirements of a workflow system in supporting engineering processes are tackled successfully by the Compliance Flow's framework. In particular the novel feature of process compliance management creates a new research direction in the workflow community, and its application in managing project compliance indicates the importance of a meta-process control capability in workflow management.

References

Abbott K. R. and Sarin S. K., 1994, *Experiences with Workflow Management: Issues for the Next Generation*, in Furuta and Neuwirth, pp. 113-120.

Ader M., 1997, *Seven Workflow Engines Reviewed*, Document World, vol. 2, no 3.

Agirre E. and Rigau G., 1995, A *Proposal for Word Sense Disambiguation using Conceptual Distance*, International Conference on Recent Advances in Natural Language Processing. Tzigov Chark, Bulgaria, September 1995.

Agirre E. and Rigau G., 1996, *Word sense disambiguation using conceptual density*, In proceeding of COLING-96.

Agirre E., Arregi X., Artola X., Daz de Ilarazza A., and Sarasola K., 1994, *Conceptual Distance and Automatic Spelling Correction*, In Proceedings of the Workshop on Computational Linguistics for Speech and Handwriting Recognition, Leeds.

Agosta J. M. and Wilkins D. E., 1996, *Using sipe-2 to plan emergency response to marine oil spills*. IEEE Expert 11(6), pp. 6-8.

Attie P. C., Singh M. P., Emerson E., Sheth A., and Rusinkiewicz M.,1996, *Scheduling Workflows by Enforcing Intertask Dependencies*, Distributed Systems Engineering, 3(4), pp. 222-238.

Azzone, G. and Bertele, U., 1994, *Techniques for comparing the economic effectiveness of Concurrent and Traditional Engineering*, in: Leondes, C.T. (Ed), Concurrent Engineering, techniques and applications, Academic Press, San Diego, pp. 25-58.

BAe/WIT/ML/GEN/SWE/1227, British Aerospace, PLC, Warton Aerodrome, Preston, UK. desJardins, M. 1996, *Knowledge acquisition tools for planning systems*, in Tate, pp. 53-61.

Bannon L. and Schmidt K., 1991, *CSCW: Four Characters in Search on context*, in J.M. Bowers and S.D. Benford (eds): Studies in Computer Supported Cooperative Work. Theory, Practice and Design, North-Holland, Amsterdam.

Barros A.P. and Hofstede A.H.M. ter, 1998, *Towards the Construction of Workflow-Suitable Conceptual Modeling Techniques*, Information Systems Journal, 8(4), pp313-337, October 1998.

Barros A.P., Hofstede A.H.M. ter, Proper H.A., and Creasy P.N., 1996, *Business Suitability Principles for Workflow Modelling*, Technical Report 380, Department of Computer Science, University of Queensland, Brisbane, Australia, August 1996.

Berry P.M. and Drabble B., 1999, *SWIM: An AI-based System for Workflow Enabled Reactive Control*, In proceedings of the IJCAI Workshop on Workflow and Process Management held as part of IJCAI-99, August 1999.

Burns T. and Stalker G.M., 1961, 1994 Revised Edition, *The management of innovation, Oxford University Press*, Oxford.

Campbell A. E. and Shapiron S. C., 1995, *Ontological Mediation: An Overview*, Proceedings of the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, Menlo Park CA: AAAI Press.

Carlsen S., 1997, *Conceptual Modelling and Composition of Flexible Workflow Models*, PhD Thesis, Department of Computer Science and Information Science, Norwegian University of Science and Technology, Norway.

Chien S. A., Govindjee A., Estlin T., Wang X., and Hill R., 1997, *Automated Generation of Tracking Plans for a Network of Communications Antennas*, Proceedings of the 1997 IEEE Aerospace Conference, Aspen, CO, February 1997, vol. 1, pp. 343-359.

Christie A., 1995, *Software Process Automation*, Springer-Verlag.

Chung P. and Jefferson M., 1998, *A Fuzzy Approach to Accessing Accident Databases*, Applied Intelligence 9 (2): 129-137.

Cichocki, Helal A. S., Rusinkiewicz M., and Woelk D., 1998, *Workflow and Process Automation: Concepts and Technology*, Kluwer.

Clark K. B. and Fujimoto T., 1991, *Product Development Performance*, Harvard Business School Press, Boston.

Cognitive Systems Inc., *ReMind Reference Manual*, Boston.

Computer Sciences Corp., Integrated Systems Division, 1998, *JCALS PC Client Simple Workflow Access Protocol (SWAP) Interface Design Document*, December 1998.

Cooper R. G., 1983, *A Process Model for Industrial New Product Development*, IEEE Transactions on Engineering Management, 30(1).

Curtis B., Kellner M., and Over J., 1992, *Process Modelling*, in Communications of the ACM, 35(9).

Daft, R. L., 1994, *Management*, The Dryden Press, Fort Worth.

Davis D. and Smith R.G., 1983, *Negotiation as a Metaphor for Distributed Problem Solving*, Artificial Intelligence, Vol. 20, pp. 63-109.

Dean J. W. and Susman G. I., 1989, *Organising for Manufacturable Design*, Harvard Business Rev., Vol. 67, No. 1, pp. 49-57.

Dellen B., Maurer F., and Pews G., 1997, *Knowledge-based Techniques to Increase the Flexibility of Workflow Management*, Data and Knowledge Engineering, North-Holland.

Dourish P., Holmes J., MacLean A., Marqvarsdsen P., and Zbyslaw A., 1996, *Freeflow: mediating between representation and action in workflow systems*, In Proceedings, CSCW'96, Boston, ACM, 190-208.

Easterbrook S., Finkelstein A., Kramer J, and Nuseibeh B., 1994, *Coordinating Distributed ViewPoints: the Anatomy of a Consistency Check.* International Journal on Concurrent Engineering: Research and Applications, 2, 3, pp. 209-222.

Eckerson W., *1994, Case Study: The Role of IS in Reengineering*, Open Information Systems, Patricia Seybold Group, Vol. 9, No. 2, February 1994.

Ehrlenspiel K., 1985, *Kostenguenstig Konstruieren (Design for Cost)*, Springer-Verlag, Berlin.

Emmerich W., Finkelstein A., Montangero C., Antonelli S., Armitage S., and Stevens R., 1999, *Managing Standards Compliance*, IEEE Trans. Software Engineering, 25 (6), November/December 1999.

ERA Technology, 2000, *Assuring Programmable Electronic Systems (APES) Project.*

Fernández-Amorós D., Gonzalo J., and Verdejo F., 2001, *The role of conceptual relations in Word Sense Disambiguation*, In Proceedings of the 6th International Workshop on Applications of Natural Language for Information Systems (NLDB-01).

Finkelstein A., Gabbay D., Hunter A., Kramer J., and Nuseibeh B., 1994, *Inconsistency Handling In Multi-Perspective Specifications*, IEEE Transactions on Software Engineering, 20, 8, pp. 569-578.

Galbraith J.R., 1973, *Designing Complex Organisations*, Addision-Wesley, Reading, Masschusets.

Genesereth M. R. and Ketchpel S. P., 1994, *Software Agents*, Communications of the ACM 37 (7), pp. 48-53.

Genesereth M.R., 1999, *Knowledge Interchange Format – draft proposed American National Standard (dpANS)*, NCITS.T2/98-004. Online Paper as of 1999-09-04.

Georgakopoulos D., Hornick M., and Shet A., 1995, *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*, Distributed and Parallel Databases, 3(2), April 1995, pp 119-153.

Glance, Natalie S., Daniele S. Pagani, and Remo Pareschi, 1996, *Generalized Process Structure Grammars (GPSG) for flexible representations of work*, in Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work, Boston, Massachusetts, pp. 180-189.

Godefroid P. and Kabanza F., 1991, *An efficient reactive planner for synthesizing reactive plans*, In Proceedings of AAAI-91, pp. 640-645.

Gruber T. R., 1992, *Ontolingua: A mechanism to support portable ontologies*, Knowledge Acquisition, vol. 5, no 2.

Hammond K. J., 1989, *Case-Based Planning: Viewing Planning as a Memory Task*, Perspectives in Artificial Intelligence, Academic Press, Boston, USA.

Han Y., Sheith A., and Bussler C., 1998, *A Taxonomy of Adaptive Workflow Management*, Proceedings of the CSCW-98 Workshop Towards Adaptive Workflow System, held during the 1998 Conference on Computer-Supported Cooperative Work in Seattle, USA.

Han Y., Sheith A., and Bussler C., 1998, *A Taxonomy of Adaptive Workflow Management*, Proc. CSCW-98 Workshop Towards Adaptive Workflow System, held during the 1998 Conference on Computer-Supported Cooperative Work in Seattle, USA.

Hayes R. H., Wheelwright S.C., and Clark, K.B., 1988, *Dynamic Manufacturing*, Free Press, New York.

Hebbar K., Smith S. J. J., Minis I., and Nau D. S., 1996, *Plan-based evaluation of designs for microwave modules*, ASME 1996 Design Engineering Technical Conference and Computers in Engineering Conference, Irving, California.

Howe E., 1995, *Improving the reliability of artificial intelligence planning systems by analysing their failure recovery,* IEEE Transactions on Knowledge and Data Engineering, vol. 7, pp. 14-25.

Hruby P., 1998, *Specification of Workflow Management Systems with UML,* Proceedings of the 1998 OOPSLA Workshop on Implementation and Application of Object-oriented Workflow Management Systems, Vancouver.

Hundal M.S., 1998, *Time-Driven Product Development,* in Integrated Product and Process Development: Methods, Tools, and Techniques, edited by Usher J., Roy U., and Parsaei H., John Wiley and Sons Inc., pp. 59-83.

IEC, 1997, *Draft Standard IEC61508 Functional safety of electrical/ electronic/ programmable electronic (E/E/PES) safety-related systems*, Parts 1 to 7, December 1997.

ISO, *Introduction to ISO*, 1997, http://www.iso.ch/infoe/intro.html.

Jarvis P., 1998, *Workflow literature Survey: the state-of-the-art in workflow systems*, Artificial Intelligence Applications Institute, The University of Edinburgh.

Jarvis P., Stader J., Macintosh A., Moore J.P., and Chung P.W.H, 1999a, *A Framework for Equipping Workflow Systems with Knowledge about Organisational Structure and Authority,* Proceedings of the Workshop on Systems Modelling for Business Process Improvement (SMBPI-99), University of Ulster, Co Antrim, Nthn Ireland, pp. 205-219.

Jarvis P., Stader J., Macintosh Ann., Moore J., and Chung P., 1999b, *Exploiting AI Technologies to Realise Adaptive Workflow Systems,* 15th European Conference on Artificial Intelligence, ECAI '02, Lyon, France, July 1999.

Jarvis P., Moore J.P., Stader J., Macintosh A., and Chung P.W.H, 2000, *Harnessing AI Technologies to Meet the Requirements of Adaptive Workflow Systems* , in Enterprise

Information Systems , Filipe, J. (ed), Kluwer Academic Publishers , pp. 163-170, ISBN 0-7923-6239 .

Joosten S., 1996, *Workflow Management Research Area Overview,* Proceedings of Second Americas Conference on Information Systems, Phoenix, Arizona.

Judge D.W., Odgers B.R., Shepherdson J.W., and Cui Z., 1998, *Agent Enhanced Workflow,* BT Technology Journal, 16, No 3, pp. 79-85.

Kambhampati S. and Hendler J., 1992, *Validation-structure -based theory of plan modification and reuse,* Artificial Intelligence, 55(2), pp. 192-258.

Kappel G., Lang P., Rausch-Schott S., and Retschitzegger R., 1995, *Workflow Management Based on Objects, Rules, and Roles,* IEEE Bulletin of the Technical Committee on Data Engineering, 18(1), pp. 11-17.

Kappel G., Rausch-Scott S., and Retschitzegger W., 2000, *A framework for workflow management systems based on objects, rules and roles,* ACM Computing Surveys, 32(1).

Kappel G., Rausch-Scott S., and Retschitzegger W., 2000, *A framework for workflow management systems based on objects, rules and roles,* ACM Computing Surveys, 32(1).

Kung D.C., 1993, *The Behaviour Network Model for Conceptual Information Modelling,* Information Systems, 18(1); pp. 1-21.

Lampson B.W., 1974, *Proctection,* ACM Operation Systems Review, Vol. 8, pp 18-24, 1974

Lee J., Grunninger M., Jin Y, Malone T., Tate A., Yost G., and other members of the PIF Working Group, 1998, *The PIF Process Interchange Format and Framework Version 1.2,* The Knowledge Engineering Review, Vol. 13, No. 1, Cambridge University Press, March 1998, pp. 91-120.

Lee Thomas J. and David E. Wilkins, 1996, *Using SIPE-2 to integrate planning for military air campaigns,* in Trends and Controversies, IEEE Expert, December 1996.

Manuela M. Veloso, Martha E. Pollack, and Michael T. Cox., 1998, *A rationale-based monitoring for planning in dynamic environments,* In Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems (AIPS--98).

March J.G. and Simon H. A., 1958, *Organisations,* Wiley, New York, USA.

Marshak R.T., 1994, *Workflow white paper: An overview of workflow software*, In WORKFLOW '94 Conference Proceedings, San Jose, August 1994.

Marshak R.T., 1997, *InConcert Workflow: Independent from XSOFT, InConcert Inc. Provides Flexible Workflow Underlying Engineering Team Support*, Workgroup Computing Report, Patricia Seybold Group, March 1997.

Mazza C., Fairclough J., Melton B., De Pablo D., Scheffer A., and Stevens R., 1994, *Software Engineering Standards*, Prentice Hall.

McCarthy D.R. and Sarin S.K., 1993, *Workflow and Transaction in InConcert*, Bulletin of the Technical Committee on Data Engineering, Vol. 16 N2 - IEEE, June. Special Issue on Workflow Extended Transaction Systems.

Medina-Mora R., Wong H., and Flores P., 1992, *The ActionWorkflow Approach to Workflow Management*, Proceedings of the Fourth Conference on Computer-Supported Cooperative Work, June 1992.

Michael zur Muehlen and Joerg Becker, 1999, *Workflow Process Definition Language - Development and Directions of a Meta-Language for Workflow Processes*, Proceedings of the 1st KnowTech Forum, Potsdam, September, 17th-19th 1999.

Miers D., 1996, *The Workware Evaluation Framework*, ENIX Ltd.

Miller G.A., Beckwith R., Fellbaum C., Gross D., and Miller K.J., 1990, *Introduction to wordnet: An on-line lexical database*, Journal of Lexicography, 3(4):234-244.

Moore J., Inder R., Chung P., Macintosh A., and Stader J., 2000a, *Combining and Adapting Process Patterns for Flexible Workflow*, In 11th International Conference on Database and Expert Systems Applications (DEXA'00), London, Greenwich, September 2000.

Moore J., Inder, Chung P., Macintosh A., and Stader J., 2000, *Who Does What? Matching Agents to Tasks in Adaptive Workflow*, In Proceedings of the 2nd International Conference on Enterprise Information Systems (ICEIS 2000), B. Sharp, J. Cordeiro, and J. Filipe (eds), Stafford, July 2000, pp 181-185, ISBN 972-98050-1-6.

Moore J.P., Inder R., Chung P.W., Macintosh A., and Stader J., 2000b, *Who Does What? Matching Agents to Tasks in Adaptive Workflow*, Proceedings of the Second International

Conference on Enterprise Information Systems, Y, Sharp, B., Cordeiro, J. and Filipe, J. (eds) , Stafford, July 2000, pp. 181-185, ISBN 972-98050-1-6 .

Moore L., 2002, *Epigram Profit from Safe Systems,* Edited by Nunns S., Spring 2002.

Musliner D.J., Durfee E.H., and Shin K.G., 1993, *CIRCA: a cooperative intelligent real-time control architecture*, IEEE Trans. Systems, Man, and Cybernetics, 23(6), pp. 1561-1574.

Myers K. and Berry P., 1999, *Workflow Management Systems: An AI Perspective*, Technical Report, AIC, SRI International, USA.

Nwana H.S., 1996, *Software agents: an overview*, In The Knowledge Engineering Review, 11, no. 3, pp. 205-244.

Object Management Group, 1995, *Object Management Architecture Guide*, Third Edition, R.M. Soley (ed.), John Wiley & Sons, Inc., New York, June 1995,

Object Management Group, 1997, *Workflow Management Facility RFP*, Issued on May 1997, Document Numbers cf/97-05-06.

Object Management Group, 1998, *Workflow Management Facility*, Joint Final Submission for the Business Object Component Architecture (BOCA) by DataAccess, EDS, NIIIP, Sematech, Genesis Development Corporation, Prism Technologies and IONA, Document Number bom/98-06-07, revised submission, 4 July 1998.

O'Brien P.D. and Wiegand M.E., 1996, *Agents of change in business process management*, BT Technology, 14, no. 4, pp. 133-140.

Opdahl A.L., Sindre G., 1993, *A taxonomy for real-world modelling concepts*, Information Systems, 19(3), pp. 229-241.

Paashuis V., 1998, *The Organisation of Integrated Product Development, Springer-Verlag Berlin Heidelberg*, New York, USA.

Pahl G. and Beitz W., 1996, *Engineering design - a systematic approach*, 2nd edn, Springer-Verlag, London, UK.

Papadias D. and Delis V., 1997, *Relation-Based Similarity*, ACM-GIS 1997: 1-4.

Pelled L.H. and Adler P.S., 1994, *Antecedents of Integroup Conflict in Multifunctional Product Development Teams: A Conceptual Model*, IEEE Transactions on Engineering Management, vol. 41, no. 1, pp.21-8.

Perrow C., 1967, *A Framework for the Comparative Analysis of Organisations*, American Sociological Reivew, vol. 32, pp. 194-208.

Perrow C., 1970, *Organisational Analysis: a sociological review*, Wadsworth.

Peter J., Jonathan M., Jussi S., Ann M., Adrew C.M., and Paul C., 1999, *Ontologies to Support the Management of New Product Development in the Chemical Process Industries*; In Proceedings of the International Conference on Engineering Design (ICED 99), Munich, Germany; ISBN 3-922979-53-X, Vol.1, pp. 159 - 164.

Peterson J.L., 1977, *Petri nets*, ACM Computing Surveys, 9(3):233 – 251, September 1997.

Petri C.A., 1962, *Kommunikation mit Automaten*, PhD thesis, University of Bonn, Bonn, Germany.

Pohl K., Weidenhaupt K., Domges R., Haumer R., Haumer P., Jarke M., and Klamma R., 1999, *PRIME-Toward Process-Integrated modelling Environments:1*, ACM Transactions on Software Engineering and Methodology, Volume 8, Issue 4, October, 1999.

Rada R., Mili H., Bicknell E., and Blettner M., 1989, *Development an Application of a Metric on Semantic Nets,* IEEE Transactions on Systems, Man and Cybernetics, 19(1), pp. 17-30.

Rupietta W., 1997, *Organisation and Role Models for Workflow Processes*, in Workflow Handbook, P. Lawrence (ed.), Wiley.

Sarin S.K., Abbott K.R., and McCarthy D R., 1991, *A Process Model and System for Supporting Collaborative Work*, in P. de Jong #ed.#, Proc. of the Conf. on Organizational Computing Systems COOCS'91, ACM, ACM Press, pp. 213-224.

Smeaton A. and Quigley I., 1996, *Experiment on Using Semantic Distance Between Words in Image Caption Retrieval*, in 19th International Conference on Research and Development in Information Retrieval SIGIR'96, Zurich, Switzerland.

Smith S.J.J., Nau D.S., and Throop T.A., 1996, *A Planning Approach to Declarer Play in Contract Bridge,* Computational Intelligence, 12(1).

Smith P. G. and Reinertsen D. G., 1991, *Developing Products in Half the Time,* Van Nostrand Reinhold, Network.

Stader J., Moore J., Chung P., McBriar I., Ravinranathan M., and Macintosh A., 2000, *Applying Intelligent Workflow Management in the Chemicals Industries,* In The Workflow

Handbook 2001, L. Fisher (ed), Published in association with the Workflow Management Coalition (WfMC), October 2000, pp 161-181, ISBN 0-9703509-0-2.

Stader J., Moore J., Chung P., McBriar I., Ravinranathan M., and Macintosh A., 2001, *Applying Intelligent Workflow Management in the Chemicals Industries*, In The Workflow Handbook 2001, L. Fisher (ed), Published in association with the Workflow Management Coalition (WfMC), October 2000, pp. 161-181, ISBN 0-9703509-0-2.

Stader J., Moore J., Chung P., McBriar I., Ravinranathan M., and Macintosh A., 2000, *Applying Intelligent Workflow Management in the Chemicals Industries*, In The Workflow Handbook 2001, Fisher L (ed), Published in association with the Workflow Management Coalition (WfMC), Oct 2000, pp. 161-181, ISBN 0-9703509-0-2.

Sussna M., 1993, *Word Sense Disambiguation for Free-text Indexing Using a Massive Semantic Network*, in Proceedings of the Second International Conference on Information and Knowledge Management, Airlington, Virginia USA.

Swenson K.D., 1993b, *Visual Support for Reengineering Work Processes*, in S. Kaplan #ed.#, Proc. of the Conf. on Organizational Computing Systems COOCS'93,ACM, ACM Press, pp. 130-141.

Swenson K.D., 1994, *The Future Workflow Technology: Collaborative Planning*, Groupware 1994, San Jose, California.

Swenson K.D., Irwin I., Matsumoto T., Maxwel R. J., and Saghari B., 1994a, *Collaborative Planning: Empowering the user in a Process Support Environment*, 15th Interdisciplinary Workshop on "Informatics and Psychology", Schaerding, Austria.

Swenson K.D., Maxwell R. J., Matsumoto T., Saghari B., and Irwin I., 1994b, *A Business Process Environment Supporting Collaborative Planning*, Journal of Collaborative Computing, vol. 1, no.1, pp. 15-34.

Swenson K.D., 1998, *Simple Workflow Access Protocol*, IETF internet draft, August 1998.

Swenson K.D., 1993a, *A Visual Language to Describe Collaborative Work*, in Proceedings of the 1993 IEEE Symposium on Visual Languages, IEEE CS Press, pp. 298-303.

Sycara K. and Miyashita K., 1992, *Incremental Schedule Modification*, Working Notes: Symposium on Computational Considerations in Supporting Incremental Modification and Reuse, AAAI Spring Symposium Series, Stanford University.

Tabbara B., Tabbara A., and Alberto Sangiovanni-Vincentelli A., 2000, *Task Response Time Optimization Using Cost-based Operation Motion*, In Proceedings of the Eighth International Workshop on Hardware/Software Codesign, San Diego, California, USA.

Tate A., 1993, *Authority Management - Coordination between Planning, Scheduling and Control*. Workshop on Knowledge-based Production Planning, Scheduling and Control at the International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France.

Thompson J.D., 1967, *Organisations in Action, McGraw-Hill*, New York, USA.

Uschold M. and Gruninger M., 1996, *Ontologies: Principles, Methods and Applications*, The Knowledge Engineering Review, Vol. 11, No. 2, pp. 93-136.

Uschold M., King M., Moralee S., and Zorgios Y., 1998, *The Enterprise Ontology*, The Knowledge Engineering Review , vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate).

Van de Ven A. H., Delbecq A.L., and Koenig Jr.R., 1976, *Determinants of Coordination Modes within Organisations*, American Sociological Review, vol. 41, pp. 322-38.

Veloso M.M., 1992, *Learning by Analogical Reasoning in General Problem Solving*, Technical Report CMU-CS-92-174, Department of Computer Science, Carnegie Mellon University.

W.M.P. van der Aalst, 1998, *The Application of Petri Nets to Workflow Management*, The Journal of Circuits, Systems and Computers, 8(1), pp. 21 - 66.

W.M.P. van der Aalst., 1996, *Petri-net-based Workflow Management Software*, In A. Sheth, editor, Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems, Athens, Georgia, May 1996, pp. 114 - 118.

WfMC, 1996, *Workflow Interface 4 – Interoperability Abstract Specification WFMC-TC-1012 V 1.0*, Workflow Management Coalition.

WfMC, 1998, *Workflow Interface 2 – Workflow Client Application Application Programming Interface (Interface 2 & 3) Specification WFMC-TC-1009 V 2.0 Naming Conventions WFMC-TC-1013 V 1.4*, Workflow Management Coalition.

WfMC, 1999, *Workflow Interface 1 – Process Definition Interchange V 1.1 Final WfMC-TC-1016-P*, Workflow Management Coalition.

WfMC, 1999, *XML based Process Management Standard*, launched by Workflow Management Coalition – "Wf-XML", WfMC Press, July 1999.

WfMC, 2000a, *Workflow Handbook 2001*, Future Strategies Inc., 2000, ISBN 0-9703509-0-2.

WfMC, 2000b, *Workflow Interface 4 – Interoperability Internet e-mail MIME Binding WFMC-TC-1018 V 1.2*, Workflow Management Coalition.

WfMC, 2001, *Workflow Standard – Interoperability Wf-XML Binding 1.1*, WfMC-TC-1023, Version 1.1, November 2001.

WfMC, 2002, *Workflow Process Definition Interface—XML Process Definition Language (XPDL) WFMC-TC-1025 Final*, Workflow Management Coalition, October 2002.

Wiegert O., 1998, *Business Process Modelling and Workflow Definition with UML: Deficiencies and Actions to Improve.*

Wilkins D.E. and Desimone R.V., 1994, *Applying an AI Planner to Military Operations Planning,* Intelligent Scheduling, M. Zweben and M. Fox, eds., Morgan Kaufmann.

Wind Y. and Robertson T.S. (1983), *Marketing Strategy: New Directions for Theory and Research*, Journal of Marketing, vol. 47, no. 2, pp. 12-25.

Winograd T. and Fernando F., 1986, *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corporation.

Zweben M., Daun B., Davis E., and Deale M., 1994, *Scheduling and Rescheduling with Iterative Repair.* In Zweben, M., and Fox, M. S. (eds.), Intelligent Scheduling, Morgan Kaufmann, pp. 241-255.

Appendix

List of Publications

Larry Y. C. Cheung, Paul W. H. Chung, Ray J. Dawson, *Supporting Engineering Design Process with an Intelligent Compliance Agent: A Way to Ensure a Standard Complied Process*. Fourth International Conference on Enterprise Information Systems, Spain, pp 341-349, April 2002.

Larry Y.C. Cheung, Paul W. H. Chung, Ray J. Dawson, *Managing Process Compliance with Standards*, Issues & Trends of Information Technology Management in Contemporary Organizations, Information Resources Management Association, USA, Edited by Mehdi Khosrowpour, May 2002.

Larry Y. C. Cheung, Paul W. H. Chung, *Supporting Engineering Design Process with Compliance Flow – An Intelligent Workflow Management System*, Engineering Design Conference, London, July 2002

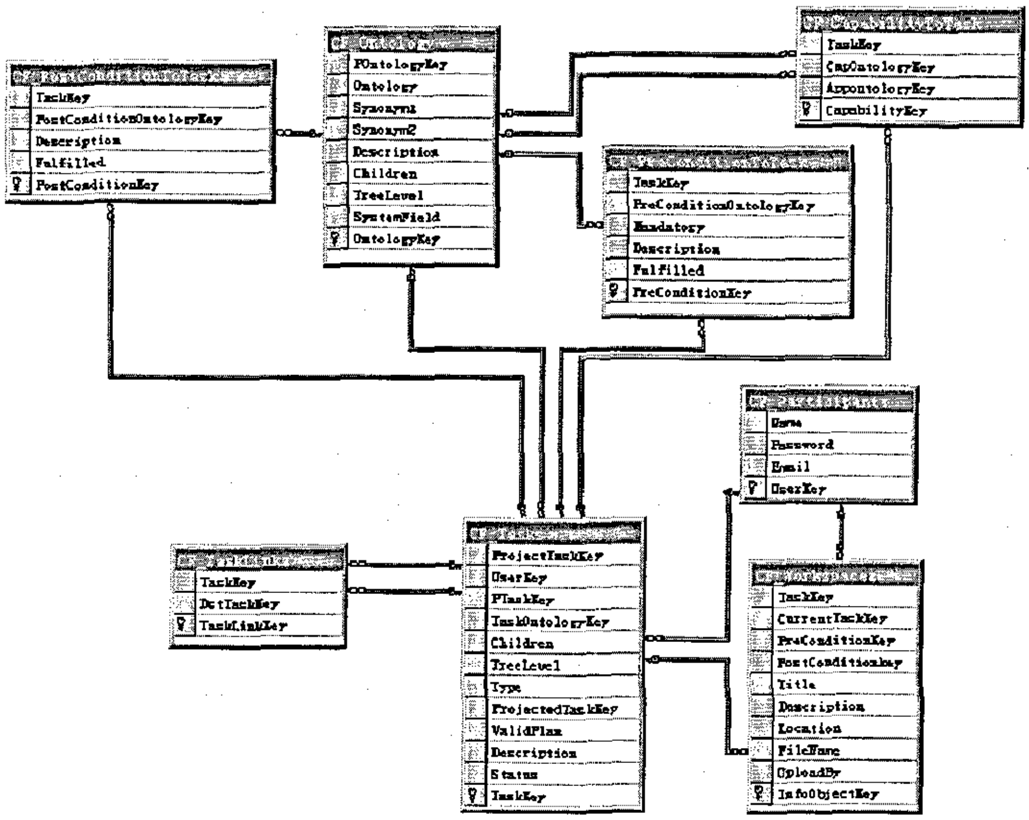
Larry Y.C. Cheung, Paul W.H. Chung, Ray J Dawson, *Managing Process Compliance*, Information Management: Support Systems & Multimedia Technology, Edited by Dr. Ditsa. (In Press)

Paul W.H. Chung and Larry Y.C. Cheung, *Managing the Compliance of Dynamic and Complex Process*, Workflow Handbook 2003, Workflow Management Coalition, 2003. (In Press)

P. W. H. Chung, L. Cheung, J. Stader, P. Jarvis, J. Moore and A. Macintosh, Knowledge-based process management – an approach to handling adaptive workflow, Knowledge-based Systems Journal. (In Press)

Database Schema

The database schema of Compliance Flow system is presented from five different views. They are (1) task management, (2) plan library, (3) standard modelling, (4) task agent and (5) tracking services. The brief description of each table and its fields is given.



Schema view 1. Task management.

CF_PostConditionToTask	
Post-conditions of particular user-defined tasks.	
Field Name	**Description**
TaskKey	GUID key of the task.
PostConditionOntologyKey	GUID key of the post-condition.
Description	Description of the post-condition.
Fulfilled	A Post-condition is either fulfilled or not fulfilled.
PostConditionKey	GUID key of the post-condition.

CF_Ontology	
Ontologies used in naming objects.	
Field Name	**Description**

Ontology	Term.
Synonym1	Synonym of the term.
Synonym2	Synonym of the term.
Description	Description of the term.
Children	The number of the child terms of the ontology.
TreeLevel1	Tree level in the ontology hierarchy.
SystemField	'True' if the term is used by internal system which cannot be changed by user.
OntologyKey	GUID key of the ontology.

CF_CapabilityToTask	
Capability required for performing particular user-defined tasks.	
Field Name	**Description**
TaskKey	GUID key for the task the capability specified.
CapOntologyKey	GUID key for the ontology as the technical part in a capability.
AppOntologyKey	GUID key for the ontology as the application of the technical part.
CapabilityKey	GUID key for the capability.

CF_PreConditionToTask	
Pre-conditions of particular user-defined tasks.	
Field Name	**Description**
TaskKey	GUID key of the task the pre-condition specified.
PreConditionOntology	GUID key of the ontology for naming the pre-condition.
Mandatory	Nature of the pre-condition, either 'Y' or 'N'
Description	Description of the pre-condition.
FulFilled	A post-condition is either fulfilled or not fulfilled.
PreConditionKey	GUID key of the post-condition.

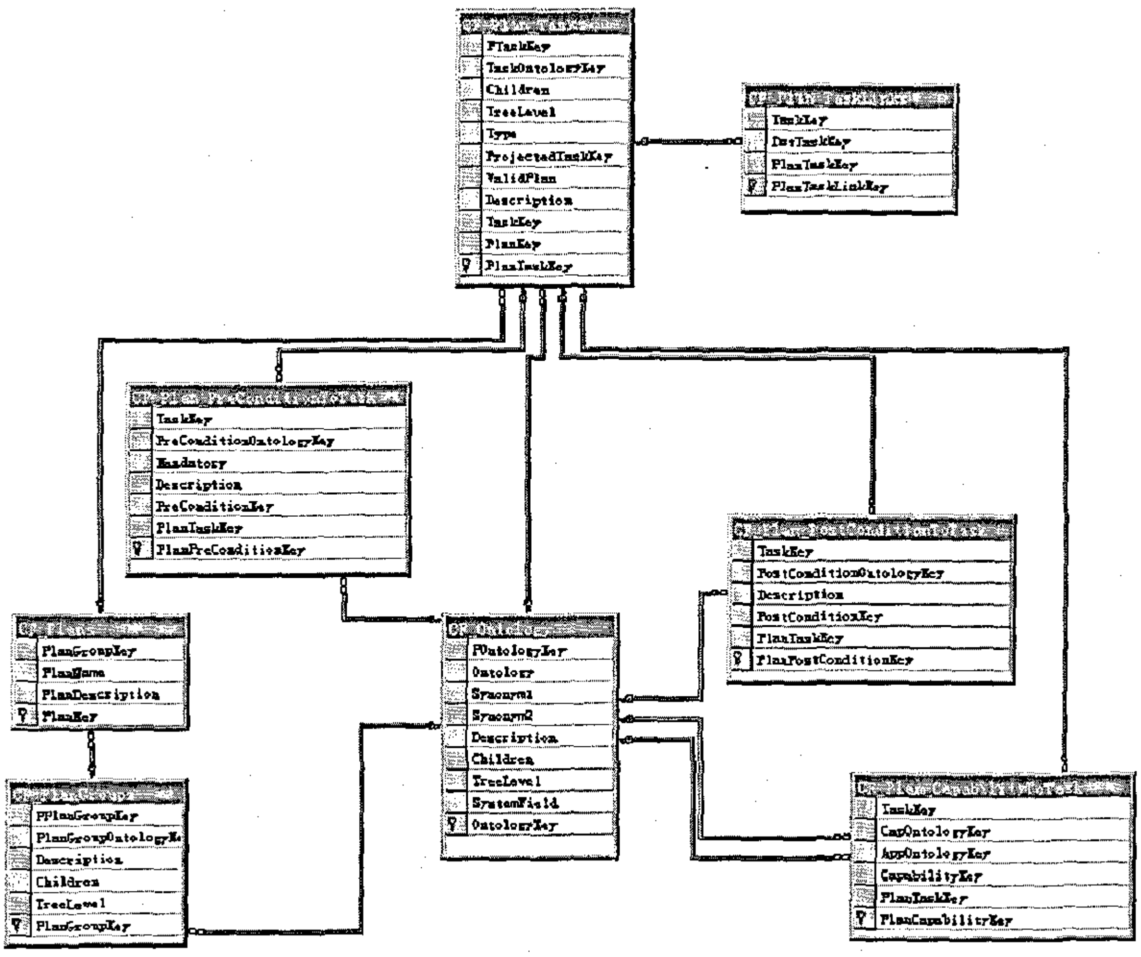
CF_Participants	
System users.	
Field Name	**Description**
Name	User name that is required for system login.
Password	Password that is required for system login.
Email	Email address.
UserKey	GUID key of the user.

CF_TaskLinks	
Links to capture the task flow in processes.	
Field Name	**Description**
TaskKey	GUID key of the 'from' task.
DstTaskKey	GUID key of the 'to' task.

TaskLinkKey	GUID key of the link.

CF_Tasks	
User-defined tasks.	
Field Name	**Description**
ProjectTaskKey	GUID key of the project which will be the same as the task key for the root task.
UserKey	GUID key of the user.
PTaskKey	GUID key of the parent which will be the same as the task key for the root task.
TaskOntologyKey	GUID key of the ontology for naming the task.
Children	Number of child task of the task.
TreeLevel	Number of level in the task hierarchy.
Type	Task type, either 'User-defined' or 'Projected'.
ProjectedTaskKey	*GUID key of the project task that is used by system to maintain the interfaces for task* decompositions.
ValidPlan	Process status, either 'Valid' or 'Not valid'.
Description	Description of the task.
Status	Status of the task. 'Inactive', 'Ready', ...
TaskKey	GUID key of the task.

CF_Workspaces	
Workspace.	
Field Name	**Description**
TaskKey	GUID key of the task the information object associated.
CurrentTaskKey	GUID key of the task the workspace associated, which is the workspace key too.
PreConditionKey	GUID key of the pre-condition that is fulfilled by the information object.
PostConditionKey	GUID key of the post-condition that is fulfilled by the information object.
Title	Title given of the information object.
Description	Description of the information object.
Location	FTP where the information object is stored.
FileName	Physical name of the information object.
UploadBy	GUID key of the user who upload the information object.
InfoObjectKey	GUID key of the information object.



Schema view 2. Plan library.

CF_Plan_Tasks	
Tasks in plans.	
Field Name	**Description**
PTaskKey	GUID key of the parent which will be the same as the task key for the root task.
TaskOntologyKey	GUID key of the ontology for naming the task.
Children	Number of child task of the task.
TreeLevel	Number of level in the task hierarchy.
Type	Task type, either 'User-defined' or 'Projected'.
ProjectedTaskKey	GUID key of the project task that is used by system to maintain the interfaces for task decompositions.
ValidPlan	Process status, either 'Valid' or 'Not valid'.
Description	Description of the task.
TaskKey	GUID key of the task.
PlanKey	GUID key of the Plan.
PlanTaskKey	GUID key of the task in the Plan which Generated by Plan.

CF_Plan_TaskLinks

Links of tasks.	
Field Name	**Description**
TaskKey	GUID key of the 'from' task.
DstTaskKey	GUID key of the 'to' task.
PlanTaskKey	GUID key of the task which is generated by Plan.
PlanTaskLinkKey	GUID key of the link which is generated by Plan.

CF_Plan_TaskLinks	
Links of tasks.	
Field Name	**Description**
TaskKey	GUID key of the 'from' task.
DstTaskKey	GUID key of the 'to' task.
PlanTaskKey	GUID key of the task which is generated by plan.
PlanTaskLinkKey	GUID key of the link which is generated by plan.

CF_Plan_PreConditionToTask	
Post-conditions of particular user-defined tasks.	
Field Name	**Description**
TaskKey	GUID key of the task.
PreConditionOntologyKey	GUID key of the pre-condition.
Mandatory	Nature of the pre-condition, either 'Y' or 'N'
Description	Description of the pre-condition.
PreConditionKey	GUID key of the pre-condition.
PlanTaskKey	GUID key of the task which is generated by plan.
PlanPreConditionkey	GUID key of the pre-condition which is generated by plan.

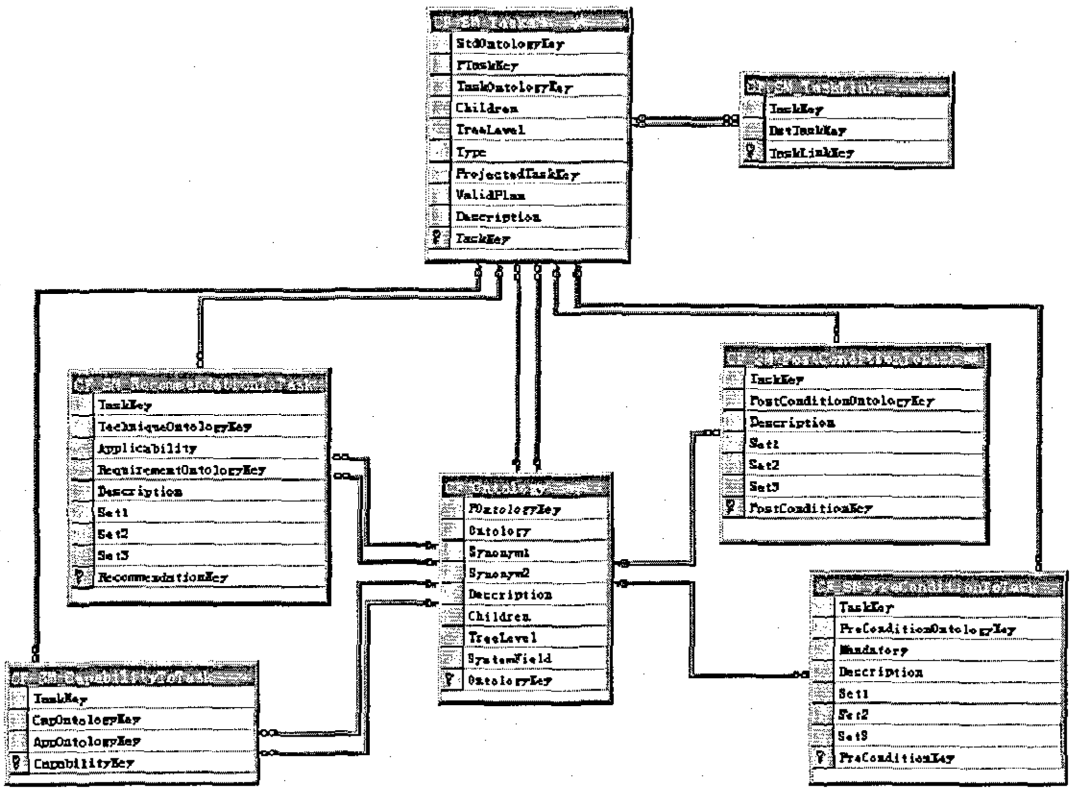
CF_Plan_PostConditionToTask	
Post-conditions of particular user-defined tasks.	
Field Name	**Description**
TaskKey	GUID key of the task.
PostConditionOntologyKey	GUID key of the post-condition.
Description	Description of the post-condition.
PostConditionKey	GUID key of the post-condition.
PlanTaskKey	GUID key of the task which is generated by plan.
PlanPostConditionkey	GUID key of the post-condition which is generated by plan.

CF_Plans

Plans.	
Field Name	**Description**
PlanGroupKey	GUID key of the plan group the plan associated.
PlanName	Name of the plan.
PlanDescription	Description of the plan.
Plankey	GUID key of the plan.

CF_PlanGroups	
Plan groups to store plans.	
Field Name	**Description**
PPlanGroupKey	GUID key of the parent of the plan group.
PlanGroupOntologyKey	GUID key of the ontology for naming the plan group.
Description	Description of the plan group.
Children	Number of the children of the plan group.
TreeLevel	Number of level of the plan group in the group hierarchy.
PlanGroupKey	GUID key of the plan group.

CF_Plan_CapabilityToTask	
Capability required for performing the tasks in the plans.	
Field Name	**Description**
TaskKey	GUID key for the task the capability specified.
CapOntologyKey	GUID key for the ontology as the technical part in a capability.
AppOntologyKey	GUID key for the ontology as the application of the technical part.
CapabilityKey	GUID key for the capability.
PlanTaskKey	GUID key of the task which is generated by plan.
PlanCapabilityKey	GUID key of the capability which is generated by plan.



Schema view 3. Standard modelling.

CF_SM_Tasks	
Standard tasks.	
Field Name	**Description**
ProjectTaskKey	GUID key of the project which will be the same as the task key for the root task.
UserKey	GUID key of the user.
PTaskKey	GUID key of the parent which will be the same as the task key for the root task.
TaskOntologyKey	GUID key of the ontology for naming the task.
Children	Number of child task of the task.
TreeLevel	Number of level in the task hierarchy.
Type	*Task type, either 'User-defined' or 'Projected'.*
ProjectedTaskKey	GUID key of the project task that is used by system to maintain the interfaces for task decompositions.
ValidPlan	Process status, either 'Valid' or 'Not valid'.
Description	Description of the task.
Status	Status of the task. 'Inactive', 'Ready', ...
TaskKey	GUID key of the task.

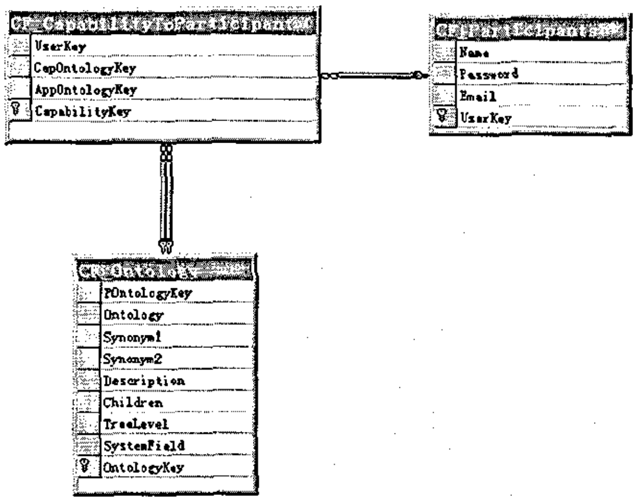
CF_SM_TaskLinks	
Links to capture the task flow in standard processes.	
Field Name	**Description**

TaskKey	GUID key of the 'from' standard task.
DstTaskKey	GUID key of the 'to' standard task.
TaskLinkKey	GUID key of the link.

CF_SM_PostConditionToTask	
Post-conditions of particular standard tasks.	
Field Name	**Description**
TaskKey	GUID key of the standard task.
PostConditionOntologyKey	GUID key of the post-condition.
Description	Description of the post-condition.
Set1	Not Used.
Set2	Not Used.
Set3	Not Used.
PostConditionKey	GUID key of the post-condition.

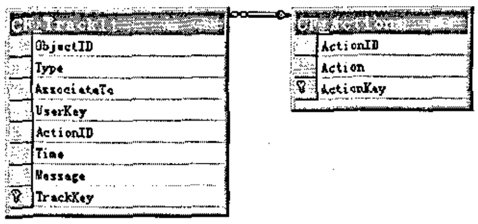
CF_SM_PreConditionToTask	
Pre-conditions of particular standard tasks.	
Field Name	**Description**
TaskKey	GUID key of the standard task.
PreConditionOntologyKey	GUID key of the pre-condition.
Description	Description of the pre-condition.
Set1	Not Used.
Set2	Not Used.
Set3	Not Used.
PreConditionKey	GUID key of the pre-condition.

CF_SM_CapabilityToTask	
Capability required for performing particular standard tasks.	
Field Name	**Description**
TaskKey	GUID key for the standard task the capability specified.
CapOntologyKey	GUID key for the ontology as the technical part in a capability.
AppOntologyKey	GUID key for the ontology as the application of the technical part.
CapabilityKey	GUID key for the capability.



Schema view 4. Task agent.

CF_CapabilityToParticipant	
Capability possessed by users.	
Field Name	**Description**
UserKey	GUID key for the user who has the capability.
CapOntologyKey	GUID key for the ontology as the technical part in a capability.
AppOntologyKey	GUID key for the ontology as the application of the technical part.
CapabilityKey	GUID key for the capability.



Schema view 5. Tracking Service.

CF_TrackIt	
Keep track the user operations.	
Field Name	**Description**
ObjectID	The object the user operated on.
Type	Type of operation.
AssociateTo	GUID key of the task the object associated.
UserKey	GUID key of the user perform the operation.
ActionID	Key of the pre-defined actions.
Time	Operation time.
Message	Reason of the operation, if given.
TrackKey	GUID key of the tracking.

CF_Action	
Action types for operations tracking.	
Field Name	**Description**
ActionID	Key of the action.
Action	Description of the action.
ActionKey	GUID key of the action.

Object Operation Descriptions

Followings are the operation descriptions of the system objected described in Chapter 8, Design and Implementation.

Second Tier – Framework Components

StandardModeller	
Operation	Description
ProcessModelling()	To model the proposed framework of a standard.

OrganisationServer	
Operation	Description
CapabilityMatching()	To assess the Goodness Of Fit (GOF) of the available agents against the required capability.

Workspace	
Copy()	To copy a document from one workspace to another. A document in a workspace is a hyperlink linking to the document stored in a secure place. Thus, only a hyperlink will be copied instead of the document itself.
Download()	To download a document from a workspace to the client machine.
Remove()	To remove a document from a workspace.
Upload()	To upload a document from a client machine to a workspace.

OntologyServer	
GetOntology()	To allow user to select an ontology from the ontology hierarchy.
Translate()	To translate a term based on its pre-defined synonyms.

TaskManager	
CompleteTask()	To complete a task by setting its state to 'Completed'.
FreezeTask()	To freeze a task by setting its state to 'Suspended'.
ProcessModelling()	To specify a process.
ProvisionTask()	To set the state of a task to 'Provisioned'.
ResumeTask()	To resume a suspended task to its previous state.
SelectAgent()	To select a suitable agent for a selected task.
StartTask()	To start a task by setting its state to 'Active'.
TerminateTask()	To terminate a task by setting its state to 'Terminated'.

Inspector	
CapabilityCheck()	To check the compliance of a task agent with a standard in terms of capability.
CompletenessCheck()	To check the required activities are concerned by a user-defined process.
CorrectnessCheck()	To ensure the sequence of the tasks in a process is complied with a standard.
ErrorPrevention()	To detect the compliance error on a task and freeze it if any error is found.
InformationCheck()	To identify any tasks which may require the information that is just uploaded, and copy that information to the workspaces.
PlanningAssistance()	To look for information which may be useful in specifying a selected task.
RecommendationCheck()	To ensure the standard recommended techniques of a task is concerned during its specification and execution.

ModelOfStandards	
FindCorrespondingTask()	To identify the corresponding standard tasks of a user-defined task.
GetCorrespondingTask()	To retrieve the corresponding standard tasks of a user-defined task.
GetRequiredPostCondition()	To retrieve the standard specified post-conditions of a user-defined task.
GetRequiredPreCondition()	To retrieve the standard specified pre-conditions of a user-defined task.
GetRequiredRecommendation()	To retrieve the standard specified recommendations of a user-defined task.
GetRequiredStandardCapability()	To retrieve the required capability to perform a user-defined task.

TrackingServer	
DisplayLog()	To retrieve show the operation history related to a task.
LogOperation()	To store a user operation into the database.

PlanLibrary	
LoadPlan()	To load a Plan the visual process editor.
SavePlan()	To save a process structure as a Plan in Plan Library.
RemovePlan()	To remove a plan from Plan Library.

Third Tier – Workflow Objects

Workspace	
GetDocument()	To receive a document and store it into the workspace.
RemoveDocument()	To remove a document from the workspace.
SendDocument()	To send the copy (hyperlink) of a document to another workspace.

TaskAgent	
GetMyTask()	To identify all the tasks belonging to a task agent.
GetCapability()	To retrieve the capability of a task agent.

Task	
GetCapability()	To retrieve the required capability of a task.
GetChild()	To retrieve the immediate child tasks of a task.
GetLink()	To retrieve the links associated to a task.
GetNextTask()	To retrieve the immediate subsequent tasks of a task.
GetParent()	To retrieve the immediate parent task of a task.
GetPostCondition()	To retrieve the post-conditions of a task.
GetPreviousTask()	To retrieve the immediate previous tasks of a task.
GetStatus()	To retrieve the current status of a task.

TaskCapability	
GetAgent()	To retrieve the agents who possess a specific capability.

Link	
GetDestination()	To retrieve the destination task of a link. The destination task is one of the immediate subsequent tasks of the task.

Pre- and Post-condition	
GetTaskAsPostCondition()	To retrieve the tasks which have the same post-condition.
GetTaskAsPreCondition()	To retrieve the tasks which have the same pre-condition.

StandardTask	
GetRecommendation()	To retrieve the recommendations of a task.

Recommendation	
GetTask()	To retrieve the tasks for which the recommendation can be applied.

Plan

Load()	To load a Plan and its components.
Move()	To move a Plan to another Plan Group.

Managing Hierarchical Data

The user-defined tasks, standard tasks, ontologies, and the plan group folders have hierarchical structures. While XML handles hierarchical data quite well, the rational database doesn't. The handling of hierarchical data therefore becomes critical to support the system.

The hierarchical structure is presented using an advanced adjacent model. A simplified data schema is presented in Table-1.

Field:	**Type:**	**Comments:**
GUID	GUID	The object's ID
Ontology	GUID	The object Name, drawing from an ontology term.
ParentID	GUID	The GUID of the parent to this object. ParentID = GUID for the root object.
TreeLevel	Integer	The number of level of this object in the object hierarchy.
NoOfChild	Integer	The number of child of this object.

Table-1. Data schema of hierarchical structure objects.

To deal with object hierarchy, a number of functions are required to support the operations of the objects in the third tier. These functions are outlined and discussed.

Adding a leaf object into a hierarchy. An example is to create a new term in an ontology where the term is a specification of another term. Adding a leaf object is simple; the parent-child relationship can be established by filling the ParentID field.

```
Function AddLeafObject(Obj)
      INSERT INTO
            TABLE (GUID, Ontology, ParentID, TreeLevel, NoOfChild)
            VALUES      (Obj.GUID,      Obj.OntologyGUID,      Obj.ParentGUID,
            Obj.TreeLevel, Obj.NoOfChild);
End Function
```

Removing a leaf object from a hierarchy. An example is to delete an outdated term from an ontology.

```
Function RemoveLeafObject(Obj)
     DELETE TABLE WHERE GUID = Obj.GUID;
End Function
```

Adding a parent object into a hierarchy. An example is to insert a new high level term into an ontology. While the adding object becomes the parent of some existing objects, it also becomes a child of another objects, as depicted in Figure-1. Adding a parent object involves two steps: (1) Adding the new object to the table and link to the parent; (2) Linking the original child objects to the new object.

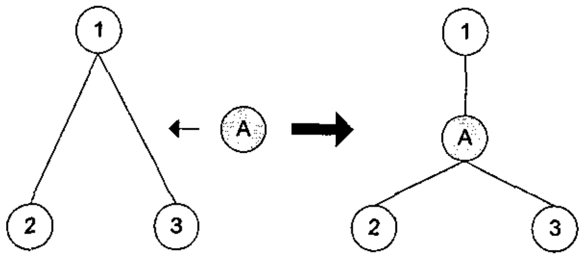


Figure-1. Adding a parent object into a hierarchy.

```
Function AddParentObject(Obj, ParentObj)
      INSERT INTO
            TABLE (GUID, Ontology, ParentID, TreeLevel)
            VALUES      (Obj.GUID,      Obj.OntologyGUID,      ParentObj.GUID,
      Obj.TreeLevel);
      UPDATE TABLE
            SET ParentID=Obj.GUID WHERE
            GUID=ParentObj.GUID;
End Function
```

Removing a parent object from a hierarchy. An example is to remove an unsuitable conceptual term from an ontology. Once the parent is removed, the grandparent will take care of the children, as depicted in Figure-2. It involves two steps: (1) Link the child objects to the grandparent; (2) Remove the parent object.

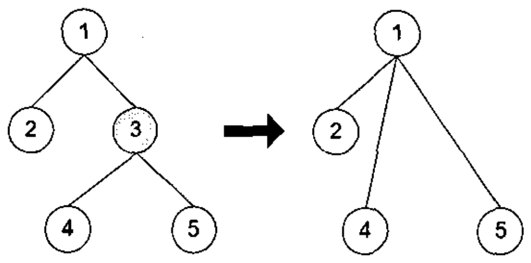


Figure-2. Removing a parent object.

```
Function RemoveParentObject(Obj)
      UPDATE TABLE
            SET ParentID=Obj.ParentGUID
            WHERE ParentID=Obj.GUID;
      DELETE TABLE WHERE GUID=Obj.GUID;
End Function
```

Moving a branch. An example is to move a term with its children to become a child to another term. Moving a branch can be simply achieved by changing the parent of the top object in the branch.

```
Function MoveBranch(Obj, NewParentObj)
      UPDATE TABLE
            SET ParentID=NewParentObject.GUID
            WHERE GUID = Obj.GUID;
End Function
```

Removing a branch. An example is to remove an abstract term with its specific terms. As the Microsoft SQL server does not support internal recursion which is support by Oracle using CONNECTED BY keyword, removing a branch requires an external recursive operation.

```
Function RemoveBranch(Obj)
      SET ChildObj IS AN OBJECT;
      IF Obj.NoOfChild > 0 THEN
            FOR EACH ChildObj in Obj
                  RemoveBranch(ChildObj);
            NEXT ChildObj
      End IF
      DELETE TABLE WHERE ParentID = OBJ.GUID;
End Function
```

Updating the TreeLevel and NofOfChild. The values of tree level and number of child of some objects will be changed because of the adding, moving or deleting an object in the hierarchy. Thus, the recalculation of TreeLevel and NoOfChild has to be performed if the object hierarchy is changed. The SQL server will trigger the following two stored procedures the table is updated.

```
Function UpdateTreeLevel()
      UPDATE TABLE
            SET TREELEVEL=CASE WHEN ParentID=GUID THEN 0 ELSE -1 END;
      WHILE EXISTS(SELECT * FROM TABLE WHERE TREELEVEL=-1)
      BEGIN
            SET @TreeLevel=@TreeLevel+1;
            UPDATE CF_TASKS
                  SET TREELEVEL=@TreeLevel WHERE TREELEVEL=-1 AND EXISTS
                  (
                  SELECT * FROM TABLE Tmp
                  WHERE Tmp.GUID=TABLE.ParentID AND TREELEVEL=@TreeLevel-1
                  );
      END
End Function

Function UpdateNoOfChild()
      UPDATE TABLE
            SET NoOfChild=(SELECT COUNT(*) FROM TABLE AS Tmp
            WHERE Tmp.ParentID=TABLE.GUID);
End Function
```

Creating an object hierarchy. An example is to create ontology hierarchy to allow user to select a term for naming a workflow object. In this case, all the objects in the hierarchy are loaded. Another case is to load just a branch. Loading a process that is part of a project is an example. The two cases have to deal with appropriately.

Loading the whole object hierarchy only requires a single query to the database.

```
Function LoadHierarchy(ObjTree)
      SET Obj IS AN OBJECT;
      SELECT * FROM TABLE
            ORDER BY TREELEVEL;
      FOR EACH Obj in TABLE
            ObjTree.Add Obj;
      NEXT Obj
End Function
```

Loading a branch is recursive operation, which is performed by retrieving objects level by level.

```
Function LoadBranch(ParentObj)
      SET Obj IS AN OBJECT;
      SELECT * FROM TABLE WHERE ParentID = ParentObj.GUID;
      FOR EACH Obj IN TABLE
            ObjTree.Add Obj;
            IF Obj.NoOfChild > 0 THEN LoadBranch(Obj);
      NEXT Obj
End Function
```

Loading the child objects' associated objects of high level object in a hierarchy. An example is to see the documents inside a workspace of a high level task. The documents inside the workspace include the documents inside the workspaces of its child tasks. Loading the associated objects includes two steps: (1) retrieving the object hierarchy by recursion; (2) retrieving the associated objects of each object in the hierarchy.

```
Function LoadAssociatedObject(Obj)
      SET ChildObj IS AN OBJECT;
      SELECT * FROM TABLE WHERE ParentID = Obj.GUID;
      FOR EACH ChildObj IN TABLE
            IF Obj.NoOfChild > 0 THEN LoadAssociatedObject(Obj);
            SELECT * FROM WORKSPACE WHERE ObjGUID=ChildObj.GUID
      NEXT Obj
End Function
```

Loading a Plan into a project. A Plan is created by saving an existing process as a template into the Plan Library. The GUID of the task objects in a Plan are taken from the original tasks stored in project database. The primary key of plan tasks is PlanGUID + TaskGUID. Thus no duplication will occur even a process is used to create two similar Plans. However, when loading a Plan into a project as a sub-process, a new GUID must be assigned to every newly created task, and thus the GUID of the parent of a newly created task is also changed. Loading a Plan involves two parts: (1) Copy Plan tasks into the project as user-defined tasks, and record both new and old GUID of the newly created tasks in a hash table. (2) Copy the associated objects into the project and link them to the corresponding tasks by referring to the hash table.

```
Function LoadPlanToProject(PlanObj, ParentTaskObj)
      SET ChildPlanObj IS AN OBJECT;
      SET AssObj IS AN ASSOCIATED_OBJECT;
      SET TaskObj AS NEW PlanObj WITH New GUID;
      SET TaskObj.ParentID=(SELECT NewGUID FROM HashTable
            WHERE OldGUID=PlanObj.GUID);
      AddLeafObject(TaskObj,ParentTaskObj);
      INSERT INTO FASHTABLE (NewGUID,OldGUID)
            VALUE(TaskObj.GUID, PlanObj.GUID);
      SELECT * FROM PLAN_ASSOCIATE_TABLE WHERE LinkObjGUID=PlanObj.GUID;
      FOR EACH AssObj IN PLAN_ASSOCIATED_TABLE
            AssObj.LinkObjGUID = TaskObj.GUID;
            INSERT INTO PROJECT_ASSOCIATE_TABLE (GUID,LinkObjGUID,…)
                  VALUE(AssObj.GUID, AssObj.LinkObjGUID,…);
      NEXT AssObj
      IF PlanObj.NoOfChild > 0 then
            FOR EACH ChildPlanObj IN PlanObj
                  LoadPlanToProject(ChildPlanObj,TaskObj);
            NEXT ChildPlanObj
      END IF
End Function
```

Capability Specification and Matching Evaluation

Evaluation Part 1

Instruction: Please outline capability and application ontology based on the following scenario, and give the knowledge overlap of the concepts of interest according to your understanding.

Scenario:

ABC Consultants provides software development services, specialising in engineering systems but also perform customised software applications for businesses. The supporting operating systems include Windows, Linux and AS400. The development tools can be categorised into three groups: object-oriented, structure and logic. Object-oriented tools include C++, Java, Visual Basic. Structure tools include Assembly Language, COBOL and RPG. Logic tools include LISP and Prolog. For each language, different version may be included.

The company has three key departments: Customer Service, IT and Accounting. There has no high degree of interdependence among the three departments.

There are four kinds of technical expertises in the company: programming, system analysis, testing, and project management. The system development is divided into three layers: presentation layer, process logic layer and database layer. Most of the programmers are familiar with more than one programming languages, but they are only used to programming a single system layer.

A system analyst is familiar with one or two kinds of application areas. Such as safety plant, risk assessment, accounting and stock management. Different system development approaches, such as RAD, SDLC and SSADM, will be used depends on the project types.

The test engineers are independent from development team. They will perform the independent test according to the system requirements during and after the programming stage. Most of the test engineers and some programmers possess professional qualifications that are required for particular type of system assessment, based on the customer's requirements. The professional qualifications include C.Eng (chartered software engineer), MBCS (member of British Computer Society), MCAD (Microsoft Certified Application Developer), MCSD (Microsoft Certified System Developer) etc.

A senior staff is assigned to manage a project. He/she is responsible to plan the project and select suitable staffs to perform particular tasks. Any technical staff may be involved in more than one project at the same time.

Part 2

Instruction: Please describe following capability specifications based on the given scenario and ontologies.

Capability(Java, ProcessLogic)

Capability(VB, Database)

Capability(Java, ProcessLogic, 70, 30, 100)

Capability(Java, ProcessLogic, 70, 30, 40)

Capability(Programmer, Windows, 50, 50, 20)

Capability(MBCS, Programmer, 50, 50, 40)

Evaluation Part 3

Instruction: Please formally specify following capabilities based on the given scenario and ontologies.

1. A staff who knows VB and experienced in database programming.

2. A programmer who knows VB.Net and experienced in database programming under Windows environment.

3. A task requires a system analyst who is experience in designing workflow system.

4. A task requires a Window based programmer who is required to use J.Builder to develop a workflow system.

5. A task requires a Borland C++ programmer. The programmer must be a chartered software engineer and is experienced in developing safety system. The programming will be performed under Windows environment.

Evaluation Part 4

Instruction: Please select the most suitable agent for each task.

Task	Wc	Wa	Ws	Result	Available Agent

Task-1
p1-Capability(ProjectManager,IT);
p2-Capability(ProjectManager,ERP);
	
	50
	50
		
		50
		50
			
			60
			40
				☐	**Agent-1:**
					Capability(ProjectManager, IT);
					Capability(ProjectManager, SSADM);
					Capability(VB, Database)

Task-2
p1-Capability(Programmer, IT);
p2-Capability(VB.Net, DataBase);
p3-Capability(VB.Net, Workflow);
	
	50
	70
	50
		
		50
		30
		50
			
			40
			40
			20
				☐	**Agent-2:**
					Capability(ProjectManager, IT);
					Capability(ProjectManager, Safety);

Task-3
p1-Capability(Analyst, IT);
p2-Capability(Analyst, CEng);
	
	50
	50
		
		50
		50
			
			50
			50
				☐	**Agent-3:**
					Capability(Programmer, IT);
					Capability(Java, Presentation);
					Capability(Java, ProcessLogic)
					Capability(VB,Database)
					Capability(VB,Accounting)

				☐	**Agent-4:**
					Capability(Programmer, IT);
					Capability(Programmer, MBCS);
					Capability(C++,ProcessLogic)
					Capability(C++,Safety)

				☐	**Agent-5:**
					Capability(Programmer, IT);
					Capability(VB.Net, Accounting);
					Capability(VB.Net,Presentation);

				☐	**Agent-6:**
					Capability(Analyst,IT);
					Capability(Analyst,RAD);
					Capability(Analyst,ERP);
					Capability(Analyst,MBS);

				☐	**Agent-7:**
					Capability(Analyst, IT);
					Capability(Analyst, SDLC);

				☐	**Agent-8:**
					Capability(Tester, IT);
					Capability(Tester, CEng);

Followings are the GOFs of every available agent against each task which are calculated by the proposed fuzzy matching algorithm.

Task	Wc	Wa	Ws	Result	GOF (%)	Available Agent

Task-1
p1-Capability(ProjectManager,IT);
p2-Capability(ProjectManager,ERP);
	
	50
	50
		
		50
		50
			
			60
			40
				✓	**Final = 80**
					p1 = 100
					p2 = 50
						Agent-1:
						Capability(ProjectManager, IT);
						Capability(ProjectManager, SSADM);
						Capability(VB, Database)

				✓	**Final = 82.4**
					p1 = 100
					p2 = 56
						Agent-2:
						Capability(ProjectManager, IT);
						Capability(ProjectManager, Safety);

				✗	**Final = 0**
					p1 = 0
					p2 = 0
						Agent-3:
						Capability(Programmer, IT);
						Capability(Java, Presentation);
						Capability(Java, ProcessLogic)
						Capability(VB,Database)
						Capability(VB,Accounting)

				✗	**Final = 0**
					p1 = 0
					p2 = 0
						Agent-4:
						Capability(Programmer, IT);
						Capability(Programmer, MBCS);
						Capability(C++,ProcessLogic)
						Capability(C++,Safety)

				✗	**Final = 0**
					p1 = 0
					p2 = 0
						Agent-5:
						Capability(Programmer, IT);
						Capability(VB.Net, Accounting);
						Capability(VB.Net,Presentation);

				✗	**Final = 0**
					p1 = 0
					p2 = 0
						Agent-6:
						Capability(Analyst,IT);
						Capability(Analyst,RAD);
						Capability(Analyst,ERP);
						Capability(Analyst,MBS);

				✗	**Final = 0**
					p1 = 0
					p2 = 0
						Agent-7:
						Capability(Analyst, IT);
						Capability(Analyst, SDLC);

				✗	**Final = 0**
					p1 = 0
					p2 = 0
						Agent-8:
						Capability(Tester, IT);
						Capability(Tester, CEng);

Task	Wc	Wa	Ws	Result	GOF	Available Agent

Task-2
p1-Capability(Programmer, IT);
p2-Capability(VB.Net, DataBase);
p3-Capability(VB.Net, Workflow);
	
	50
	70
	50
		
		50
		30
		50
			
			40
			40
			20
				☑	**Final = 50**
					p1 = 0
					p2 = 100
					p3 = 50
						Agent-1:
						Capability(ProjectManager, IT);
						Capability(ProjectManager, SSADM);
						Capability(VB, Database)

				☒	**Final = 0**
					p1 = 0
					p2 = 0
					p3 = 0
						Agent-2:
						Capability(ProjectManager, IT);
						Capability(ProjectManager, Safety);

				☑	**Final = 84**
					p1 = 100
					p2 = 100
					p3 = 70
						Agent-3:
						Capability(Programmer, IT);
						Capability(Java, Presentation);
						Capability(Java, ProcessLogic)
						Capability(VB,Database)
						Capability(VB,Accounting)

				☑	**Final = 46.8**
					p1 = 100
					p2 = 11
					p3 = 12
						Agent-4:
						Capability(Programmer, IT);
						Capability(Programmer, MBCS);
						Capability(C++,ProcessLogic)
						Capability(C++,Safety)

				☑	**Final = 72**
					p1 = 100
					p2 = 50
					p3 = 60
						Agent-5:
						Capability(Programmer, IT);
						Capability(VB.Net, Accounting);
						Capability(VB.Net,Presentation);

				☑	**Final = 28**
					p1 = 70
					p2 = 0
					p3 = 0
						Agent-6:
						Capability(Analyst,IT);
						Capability(Analyst,RAD);
						Capability(Analyst,ERP);
						Capability(Analyst,MBS);

				☑	**Final = 28**
					p1 = 70
					p2 = 0
					p3 = 0
						Agent-7:
						Capability(Analyst, IT);
						Capability(Analyst, SDLC);

				☑	**Final = 28**
					p1 = 70
					p2 = 0
					p3 = 0
						Agent-8:
						Capability(Tester, IT);
						Capability(Tester, CEng);

Task	Wc	Wa	Ws	Result	GOF	Available Agent

Task-3
p1-Capability(Analyst, IT);
p2-Capability(Analyst, CEng);
	
	50
	50
		
		50
		50
			
			50
			50
				☒	**Final = 0**
					p1 = 0
					p2 = 0
						Agent-1:
						Capability(ProjectManager, IT);
						Capability(ProjectManager, SSADM);
						Capability(VB, Database)

				☒	**Final = 0**
					p1 = 0
					p2 = 0
						Agent-2:
						Capability(ProjectManager, IT);
						Capability(ProjectManager, Safety);

				☑	**Final = 35**
					p1 = 70
					p2 = 0
						Agent-3:
						Capability(Programmer, IT);
						Capability(Java, Presentation);
						Capability(Java, ProcessLogic)
						Capability(VB,Database)
						Capability(VB,Accounting)

				☑	**Final = 35**
					p1 = 70
					p2 = 0
						Agent-4:
						Capability(Programmer, IT);
						Capability(Programmer, MBCS);
						Capability(C++,ProcessLogic)
						Capability(C++,Safety)

				☑	**Final = 35**
					p1 = 70
					p2 = 0
						Agent-5:
						Capability(Programmer, IT);
						Capability(VB.Net, Accounting);
						Capability(VB.Net,Presentation);

				☑	**Final = 86**
					p1 = 100
					p2 = 72.5
						Agent-6:
						Capability(Analyst,IT);
						Capability(Analyst,RAD);
						Capability(Analyst,ERP);
						Capability(Analyst,MBS);

				☑	**Final = 75**
					p1 = 100
					p2 = 50
						Agent-7:
						Capability(Analyst, IT);
						Capability(Analyst, SDLC);

				☑	**Final = 35**
					p1 = 70
					p2 = 0
						Agent-8:
						Capability(Tester, IT);
						Capability(Tester, CEng);