A Workflow Automation Tool for ISO 9000 Compliant Quality Management with Applications in Software Development

YAN CHING YING

M. PHIL.

The Hong Kong Polytechnic University

1999

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my supervisor, Dr. Keith Chan, for his advice and guidance while this project was in progress. He is helpful and gives me useful suggestions.

Special thanks go to my external examiners, Dr. Michael R. Lyu from The Chinese University of Hong Kong and Dr. H. Q. Wang from The City University of Hong Kong, for their valuable time and effort in reviewing my thesis and providing insightful comments and recommendations.

I am especially grateful for the support that I have received from the director of my company, C.K. To, who gives me the chance to gain practical experiences in software development. And, I have been fortunate to learn and share ideas from my colleagues.

I have been particularly lucky to have the continuing love and support from my dearest friends. It is difficult to name them all, but I will try. Thank you Silva, for her patience and confidence in me, keeping me company. Thank you Janis, for her kindness and help, sharing and laughing with me. Thank you Ewina, for her trust and encouragement. Thank you Venise, for her understanding, care and support. Thank you Jimmy, for his opinions and constructive criticisms. Thank you Boris, for his kindness and patience to listen to me. Thank you Kitty, for her frequent warm visits. Thanks also to Po Kan, Amice and Justina for their care and trust.

Last but not least, I was fortunate to have my parents and grandmother to take care of me and put trust in me. Thank you my sister and brother-in-law, Rachel and Danny, for their confidence and patience. Thank you my brother, On, for his love, continuing support and suggestions.

ABSTRACT

Quality management involves defining quality goals, establishing plans to achieve these goals, and monitoring activities according to the plans. To have an effective quality management system (QMS), recognized international standards, such as that of the popular ISO 9000 series, are often deployed both for business requirement and for product and process excellence. The essence of the ISO standard is to 'say what you do and do what you say'. For a QMS to achieve this, it is not only important to focus on the design of the right processes but also to ensure all tasks are carried out according to the design. In this thesis, we present a workflow automation tool called WAT that can be used to facilitate the construction of such a system. It is a client-server tool equipped with a number of unique features to allow ISO 9000 compliant procedures to be specified and enacted with minimal investment of time and effort. It has two major components : the workflow capturing (WC) component and the workflow enactment (WE) component. The WC component allows users 'say what they do' whereas the WE component makes sure that users 'do what they say'.

The WC component provides users with a graphical user interface (GUI) to define workflow elements such as actors, tasks, forms, and the relationships between them. Users can define ISO 9000 compliant procedures by specifying the sequences of tasks and the actor(s) assigned to each task. Also, the role of each actor and the interactions within and between them can be made explicit. Unlike many workflow software which focus mainly on one particular model, WAT supports three different models : (i) an actor model which makes it easy for the responsibilities of individual members be defined according to the organizational structure; (ii) an information model which allows processes to be defined based on the forms that shared among tasks, and (iii) a process model which allows procedures be defined in terms of task sequence. Users can define processes by the actor or process model. The WC component allows these models to be transformed from one to the other to provide a rich conceptual framework for the capturing of ISO 9000 compliant processes. It allows users to view the details of the defined procedures from different perspectives so that they can understand better the flow of tasks and the interaction among actors.

After the details of the procedures are captured, WAT can translate them into workflow specifications using a modified version of the WIDE language. Based on the specifications, the WE component, which is a groupware application implemented with Lotus Notes/Domino Release 4.65, ensures that all actors 'do what they say' by automating the flow of tasks according to the definition made in the WC component. The WE component, through its many features, brings responsible actors into and out of the process. Actors are informed and reminded of their responsibilities by their to-do list being updated and by receiving email reminders, etc. In addition, this component also provides a set of quality procedure templates and an efficient and reliable document manipulation environment to store information and facilitate automatic document routing. WAT has been completely implemented and has been put to test in different real scenarios in a number of different software development environments. It has been found to be a useful tool for quality management and in addition, it has also been found to be able to help improving and streamlining existing procedures by avoiding redundant data gathering and processing.

TABLE OF CONTENTS

LISTS OF TABLES

LIST OF FIGURES

1. Introduction

Quality management is concerned with the achieving, sustaining and improving of quality in an organization. It includes such things as quality control, quality improvement and quality assurance [*Hoyle, 1994*]. The International Organization of Standardization defines quality control to be the operational techniques and activities that are used to fulfill requirements for quality [*ISO, 1994*]. It deals, in other words, with the maintaining of standards and the prevention of undesirable changes. This differs from quality improvement that is concerned with a process to improve quality by better control of existing standards or by creating new standards. And quality assurance, which is concerned with all the planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.

While finding the best definitions for the many terms concerning quality management, it may not be of concern to many people. The actual requirement for the development of an effective quality management system (QMS) has been regarded by many as an important element for an organization to be competitive and successful in the commercial world nowadays. A QMS is the encapsulation of the management controls needed for helping staff deliver quality products and services in a repeatable manner [*Fulton and Myers, 1995*]. To avoid problems such as project delay, over-budgeting and failure to meet customer requirements, it is important for companies to establish and maintain an effective QMS. This is because this will enable them to achieve, sustain and improve quality. Directly and indirectly, this is also expected to facilitate management and control processes within these companies.

To properly establish and manage a QMS, it is important that a group of collaborating individuals respond to relevant activities at the right time and that proper resources be allocated to them whenever necessary. To ensure that this can be achieved, it involves complex interactions and dependencies among activities and people. Therefore, good workflow, proper document control, effective communication as well as clear roles and responsibilities are essential in implementing a QMS. An organization should support and determine what each staff should do in a process and how they perform their tasks.

To develop effective QMSs, much recent emphasis has been put on the use of international quality standards, and in particular, on the ISO 9000 series of standards. ISO 9000 is an international quality management standard developed by the International Standardization Organization (ISO) which defines what an organization must do to confirm with the standards for a QMS. The ISO 9000 series has become a symbol of good quality and many companies deploy it both for business requirement and for product and process excellence in order to satisfy marketing requirements, to manage quality and achieve continual quality improvement. In this thesis, we describe the effort we have made towards the design and implementation of a tool that can be used to develop ISO 9000 compliant QMS.

1.1. The Problems

The ISO 9000 series of standard states the requirements for a generic quality management system. The essence of ISO 9000 certification is to "say what you do and do what you say" [*Gianluigi 1993, Schmauch, 1995*]. In other words, to comply with this standard, organizations should describe clearly proper procedures that they follow and conform to specified requirements [*Gianluigi 1993*]. Saying what you do is often difficult and this is especially true in the case of software development. In some cases, such as in the case of software development, some tasks are inherently complicated and challenging to implement and control [*Armenise et al, 1992*]. These tasks often involve complex cooperation, coordination and communication of activities that occur within people in an organization [*Gulla and Lindland 1994*]. Also, tasks such as those related to software development and implementation may require constant modification. They may need to be expanded during and after development. And these characteristics make quality management under such circumstances very difficult. For example, it is difficult to document what and how the processes and responsible staff do. And, correct and consistent process execution is not easy to enforce. As a result, how to capture and enact process plays an important role in establishing and maintaining a QMS.

For some cases, such as those related to software development, the practices of QM involve defining, understanding, implementing and managing software processes. Therefore, software processes have to be planned and well defined so that software quality can be managed and controlled [*Chan 1993*]. Since software are recognized to be dynamic which is subjected to frequent changes during and after the development life cycle [*Welsh and Han 1994*], software processes should be visible, flexible and subject to constant improvement. That is, the relationship between the activities, how people perform the activities and interact with others, as well as the resources required and exception-handling mechanisms should be clearly stated. To fulfill the ISO requirements and establish a QMS, a comprehensive process model is thus required to describe processes in order to make them complete and effective. Then, people can easily describe and understand what they do. Therefore, one of our objectives is to build an effective model for processes so as to incorporate process improvement.

The successful completion of a process and its quality depends highly on the coordination of people in completing a set of activities in a particular sequence and within expected time constraints. In software development, one of the common obstacles to the achievement of high product quality is the ineffective communication between people. To ensure adequate and correct information flow throughout development life cycle, another objectives of our project is to provide a balanced and comprehensive approach to facilitate group coordination and communication.

Also, in cases that there is a need to produce various documents and intensive information sharing [*Welsh and Han 1994*], management of quality is all the more difficult. For example, owing to the dynamic nature of software processes, information is also subjected to continued changes. To manage process also means to take care of efficient document flow between actors and the control of creation, modification and referencing of documents. Moreover, a QMS requires documentation of quality manual and a comprehensive set of procedures in the company, as well as associated quality records. Therefore, document control has been regarded as one of the essential activity in a QMS to determine the correctness and quality of processes.

1.2. The Proposed Solution

Since it is important, to achieve ISO 9001, by saying what you do and doing what you say, a good solution to the above problem is to develop a system to facilitate this. To say what you do is to specify ISO 9000 compliant procedures that are to be adopted by the organization. In other words, we not only need to ensure that proper processes be developed, we need also to make sure that they are visible and understandable. Recently, workflow automation has made dramatic impacts in improving productivity and quality of product [*Georgakopoulos et al 1995*]. It is a technology that guides sequences of tasks in a process and performs automatic tasks that do not need human intervention. It is capable of handling rich process structures, controlling complex control flow and automatic task assignment. Therefore, to build a QMS, we propose to enact and manage processes based on a workflow paradigm.

In this thesis, we propose to use a workflow approach to allow ISO 9000 compliant processes to be specified and managed. A process is considered to be a sequence of tasks that depend on the cooperation and communication within and between groups of collaborating individuals. Based on this, a workflow automation tool (WAT) has been developed to assist companies to build a QMS to facilitate the implementation of ISO 9000 compliant procedures with minimal investment of time and effort. The tool makes process visible through well defined workflow elements, structure and representation. And, it is able to support group work by helping actors to cooperate and communicate.

To automate processes in company's workflow, WAT provides a standard environment to capture processes and their explicit information so as to make sure that the required information is accessible by responsible staff members. It provides a central repository of quality templates in shared databases for documentation and data recording. To facilitate document control, WAT provides an effective environment to create, edit, automatic routing as well as keep track of documents. In order to ensure orderly editing of the documents concerned in different tasks, it supports access control of documents, version control mechanism to keep track of document versions and editing histories. Also, it takes advantage of the messaging facilities of groupware for document routing and notification

among members of a project team. All these features provide a group of collaborators a shared, distributed environment to support and control processes.

WAT has two components : workflow capturing (WC) and workflow enactment (WE). The WC component provides graphical interfaces for users to describe workflow. With modification to the WIDE workflow model, WAT provides a rich conceptual model to capture processes and support features that are required, such as organization structure and information model. It supports i) an actor model that captures organizational structure, ii) an information model that records presentation details, iii) a process model which supports workflow automation. Also, transformation between process and actor models is allowed to provide multi-perspectives when capturing and understanding processes and the interaction between actors. This can fulfill *'say what to do'* in the ISO philosophy.

Workflow details are translated into specification with reference to the WIDE workflow specification. Based on these, the WE component ensures consistent process implementation across the organization by implementing procedures through groupware. It takes advantage of useful groupware features to streamline communication, enhance collaboration and facilitate cooperation of team members, as well as manage and control the interactions between processes. Also, it handles task exceptions whenever necessary and provides easy status tracking of processes and tasks. Moreover, it allows distributed process management across time and space, and supports personal task management by sending work reminder. This helps companies to *'do what they say'*.

As a result, the WC component is useful in supporting process modeling, and concepts like process instances and rich organizational models. Whereas, the WE component is responsible for supporting process instances, routing of documents, integration with other applications, and cooperative work in distributed environment.

1.3. Contributions

In this thesis, we propose to use workflow approach to help organization to build an ISO 9000 compliant quality management system. To achieve this, we have developed a workflow model to capture necessary and useful workflow information, a workflow specification language to formally describe workflow so as to facilitate workflow implementation. And, as a proof of our proposed solution, a workflow automation tool (WAT) has been developed, which is an implementation based on our workflow model and specification language. WAT makes process visible and supports group work by well-defined workflow elements and groupware functions. It also provides a central repository of quality templates for documentation (e.g. company policy, work procedures, quality manuals) and data recording [Yan and Chan, 1998]. In summary, WAT is able to provide the following functions :

- workflow capturing functions to retrieve process information
- process control functions to raise exceptions
- work-to-do list function for notification of task assignment
- process status functions to keep tracks of process or task
- process maintenance functions to modify process or task
- data handling functions to retrieve workflow data
- application integration functions to invoke other external applications

WAT provides a shared working environment and graphical user-interface to capture organizational processes. By utilizing object-oriented representation, workflow elements can be defined in an well-organized and systemic way. WAT supports two workflow models : task-based and actor-based. However, there requires no programming background to define workflow and involves user-friendly visual programming development environment in Lotus Notes. From the graphical workflow editor, workflow designers can easily locate loop back problems and aware of incompleteness of workflow. Also, it helps to detect complex part of the workflow (facilitate the evaluation of the feasibility and difficulties in the implementation tasks) [*Chan and Yan 1998, Yan and Chan 1998*].

To ensure consistent process implementation across the entire organization, workflow definitions are centrally stored in a database. It also supports distributed process management across time and space. Apart from that, WAT provides users a lot of workflow functions such as automatic process initiation, task notification and assignment. From the workflow tracking database, users can track status of processes and tasks, trigger exceptions whenever necessary and check current workload and performance of staff easily. To support better navigation between databases, there are links in the workflow specification database to directly points to any application database [*Chan and Yan 1998, Yan and Chan 1998*].

Moreover, as workflow elements are component-based, they can easily be customized and reused. Therefore, respective adjustments and modification to the workflow could be done easily. Since maintenance and modification of processes can be simplified, the time for the management and developers to modify a workflow would be greatly reduced.

1.4. Outline of Thesis

The structure of the thesis is as follow. Chapter 2 is a comprehensive literature survey on different existing commercially available ISO tools, workflow products produced by major software vendors and the attempts that have been made to develop QMS systems with workflow management capabilities for ISO certification. In Chapter 3, a workflow approach will be described as our solution to manage processes. And, Chapter 4 concentrates on workflow specification to facilitate workflow enactment by the WIDE specification language that includes well-defined elements, relationships and model transformation. Chapter 5 describes system implementation of WAT to illustrate how workflow can be enacted. And, several test cases will be discussed in Chapter 6 to demonstrate how the system automates workflow and improves quality. Lastly, Chapter 7 gives the concluding remarks by summarizing all the above mentioned and suggesting possible future work.

2. LITERATURE SURVEY

In the past decade, many software products have been developed for QMS in general and for ISO 9000 certification in particular. However, very few of them have a focus on software development or information technology and even fewer of them utilizes concepts in workflow management. In this chapter, we first present the results of a comprehensive survey we have performed on some currently available commercial ISO tools. We then describe the workflow products produced by major software vendors such as Lotus, IBM, Xerox and Action Technologies. Lastly, we discuss the attempts that have been made to develop QMS systems with workflow management capabilities for ISO certification.

2.1. ISO Tools

There are currently many different tools developed to facilitate ISO certification from different vendors. They have different features focus and functionality. In this survey, we present twenty-two of them. Their features can be grouped into five major categories : i) those that assists users in the documentation of the QMS, such as the provision of facilitates for writing quality manuals, and procedures; ii) those for document control, such as the provision of forms and the support of report generation; iii) those to assess ISO readiness such as the provision of checklists; iv) those that facilitate system audit, such as the maintaining of audit schedule and corrective action request; and v) those of others, such as network-enablement and on-line help. Table 2-1 shows more detailed information of these tools.

Of the twenty-two tools, only five of them have both quality manual and procedure templates to assists users to document their QMSs. They are : i) ISO 9000 Educator/Auditor; ii) Powerway Tools; iii) Quality System Development Kit; iv) Multimedia ISO 9000 Logic for Windows; and v) ISO ^{Plus}. And, a few tools only have one of the features. For example, ISO in a can has only quality manual templates and Quality Control System 9000, Self-Assessment Utility Program, Q-Pulse, ISO 9000 Self Generating Quality Procedures and Essential SET only have procedure templates.

Of the twenty-two tools, seven of them have features to facilitate document control so that document can be created, modified, tracked and controlled easily. These seven tools are : Documentation Control System, ISO in a can, ALI Internal Audit Tracker, Powerway Tools, Quality Workbench, Q-Pulse and QMS/9000+ Compliance Group. Only three of the twenty-two tools provide forms to create quality records. They are Powerway Tools, Q-Pulse and Essential SET. Unlike other features, report generation is a feature that most of the tools have. In fact, except, ISO in a can, ISO 9000 Quality System Checklist, Audit Master Checklist, ISO 9000 Self Generating Quality Procedures, Multimedia ISO 9000 Logic for Windows, ISO plus and Essential SET, the rest of the other tools are all equipped with report generation features.

To evaluate ISO readiness, self-assessment questionnaires and checklists are often used. However, only three of the twenty-two tools, namely ISO 9000 Internal Audit Management, Powerway Tools and Self Assessment Utility Program provides users with both and only eight of them provide checklists : ISO 9000 Quality System Checklist, How to Implement ISO 9000, Audit Master Checklist, Self Assessment Utility Program, ISO 9000 Educator/Auditor, ISO 9000 Internal Audit Management, Quality Workbench and Quality System Development Kit.

Half of the tools surveyed provide facilities for audit against the ISO standard. These tools usually allow all audit documents to be stored in a central repository so that they can be tracked and managed easily. Also, many of them have reporting capabilities to allow summary of audit results. In addition, ten of the twelve tools support corrective action to a certain degree : Corrective Action System, ids Correct Action Tracking System, ISO 9000 Educator/Auditor, ISO in a can, ISO 9000 Quality System Checklist, Audit Master, Quality Workbench, Q-Pulse, QMS/9000+ Compliance Group and ISO plus.

There are nine tools that are network enabled so that information between different divisions, suppliers or customers that are geographically apart can be communicated and exchanged. These tools are : ISO in a can, ISO 9000 Educator/Auditor, ids Correct Action Tracking System, ISO 9000 Internal Audit Management, Documentation Control System, Corrective Action System, Quality Control System 9000, Audit Master and Quality Workbench. Half of these tools have on-line help.

The features and functions mentioned above are useful and important for ISO certification. However, none of the tools surveyed contain all of them. Also, only three out of the twenty-two tools are developed for the software industry but unfortunately they are not developed for all different clauses required. For example, though Document Control System is relatively easy to be adopted in the software industry, it is solely for document control. ISO plus only provides some documentation templates for the documentation of a software QMS. And similarly, Essential SET provides a set of sample document templates for software development, project management and quality assurance. But, in addition, it also has features to help companies plan new projects and track development process. Unfortunately, it does not contain enough to completely meet the requirements for ISO 9000. In other words, therefore, there is currently a lack of comprehensive ISO tools in the market for the software industry.

Product	Application Domain	Platform	Self-assessment	Form	Quality Manuals	Network enable	Online help	Report generation	Procedure	Corrective action	Auditing	Document Control	Checklists	Others	Notes
Quality Control System 9000	Manufacturing	Win 95 Win 31				✗	✗	✗	✗		✗			Bill of material	
Corrective Action System	Manufacturing	Win 95 Win 31				✗	✗	✗		✗					
Documentation Control System	Manufacturing	Win 95 Win 31				✗	✗	✗				✗			applicable to general business
ISO 9000 Internal Audit Management	Manufacturing	Win 95 Win 31	✗			✗		✗			✗		✗		
ids Correct Action Tracking System	Manufacturing	Win 95				✗		✗		✗					
ISO 9000 Educator/Auditor	General Business	Win 95			✗	✗		✗	✗	✗	✗		✗		
ISO in a can	Manufacturing	Win 95 / NT 3.51			✗	✗	✗			✗	✗	✗			
ISO 9000 Quality System Checklist	General Business	Win 3.1/ Win 95								✗	✗		✗		
ALI Internal Audit Tracker	General Business	*Ref Notes						✗			✗	✗			Able to run in all platforms supporting Lotus Note R4.X
How to Implement ISO 9000	General Business	Win 3.1/ Win 95					✗	✗					✗	Training in ISO	
Audit Master Checklist	General Business	Win 3.1/ Win95									✗		✗		Need to incorporate with Audit Master
Powerway Tools	General Business	Win 95	✗	✗	✗		✗	✗	✗			✗			

Table 2-1 Summary of ISO tools.

Product	Application Domain	Platform	Self-assessment	Form	Quality Manuals	Network enable	Online help	Report generation	Procedure	Corrective action	Auditing	Document Control	Checklists	Others	Notes
Self Assessment Utility Program	General Business		✗					✗	✗				✗		
Audit Master	General Business	Win 3.1/ Win95				✗	✗	✗		✗	✗				
Quality Workbench	General Business	Win 95				✗		✗		✗	✗	✗	✗		
Q-Pulse	General Business	Win 3.1/ Win 95		✗				✗	✗	✗	✗	✗		Password protection	
QMS/9000+ Compliance Group	General Business	Win 95					✗	✗		✗	✗	✗		Real time exception reporting	
Quality System Development Kit	General Business	Win 95			✗		✗	✗	✗		✗		✗	Training in ISO9001	Cover some aspect of software industry
ISO 9000 Self Generating Quality Procedures	General Business	Win 3.1/ Win 95							✗						
Multimedia ISO 9000 Logic for Windows	General Business	Win 3.1			✗		✗		✗						Cannot be run in Win 95 environment
ISO Plus	Software Industry	Win 3.1 / Win 95			✗		✗		✗	✗					Word 2.0 or above required
Essential SET (Software Engineering Templates)	Software Industry	Win 3.1 / Win 95		✗					✗						Word 2.0 or above required

Table 2-1 (Cont'd) Summary of ISO tools.

2.2. Workflow Products

The workflow market has been the focus of many major software producers including Lotus, IBM, Xerox and Action Technologies. The reason for their development is mainly because of the recent realization of the need to use computer-based system to smooth out business processes in many organizations. Many of the workflow software have an engine to automate process executions and provide various capabilities, such as task assignment, notification and process tracking, to support the flow of work in an organization. In the following, we discuss the major players in the market and describe what the current state-of-the-art is regarding commercial development of workflow.

Lotus Notes is a product optimized for developing databases. Databases in Notes consist of documents, rather than traditional records. Lotus Notes is ideal for document tracking applications. It offers useful workflow features such as serial routing for notification, automation for scheduled and automatic tasks, and document tracking for status monitoring. In particular, Notes allows the development of agents which, when combined with these features, can emulate rule-based workflow routing and management. To build a workflow application using Notes, the application developers have to understand the details of the organization or be given detailed specifications. Lotus Notes has become existing proof of the concept that business users want to develop group applications that reflect how their businesses run [*Coleman and Khanna, 1995*].

IBM's workflow product is named FlowMark. FlowMark is an object-oriented workflow builder and manager. It has a client-server architecture and it also supports automatic task execution based on the specifications of processes given by its users [*Mohan et al 1995*]. In addition, FlowMark supports flexible assignment by roles and organization. The disadvantage about FlowMark is that, it is not a stand-alone product. It must be integrated with external software to provide a complete document management solution. Furthermore, application access to databases is not modeled in FlowMark and it has limited mechanism to support exception handling. Compared to Notes, FlowMark can only serve as a front-end for the capturing of workflow details. FlowMark lacks database and communication functions and it requires one or more application as a back-end to drive the flow.

Xerox Xsoft also has an object-oriented client-server workflow management system which is called InConcert. It is relational DBMS-based and is thus unlike Notes, which is form-based. InConcert is designed to build document-related workflow applications [*Coleman and Khanna, 1995*]. For a given application, an InConcert solution typically involves configuring workflow process definitions, and integrating desktop and other software tools used in different tasks that comprise the processes. InConcert can help organizations perform their current operations, and to a certain extent support process reengineering and improvement over time. Despite these advantages, unfortunately, it does not provide quality templates and does not handle actor relationships.

ActionWorkflow by Action Technologies does not use DBMS as the basis. It uses Lotus Notes for storage instead. The system specifies workflow based on speech act, i.e. on specific communication patterns between actors that require the performance of tasks and actors. Workflow applications are modeled in business process maps, consisting of interlinked workflow loops specified by the ActionWorkflow Analyst. The ActionWorkflow Builder then takes the map and generates the base code for the application. ActionWorkflow has the advantage of being flexible but the trade-off is a steeper learning curve and a harder formalization of simple, repetitive processes. Task relationships in ActionWorkflow are also difficult to express and understand. And, the process map representation makes complex processes very complicated. Furthermore, task sequences and conditional branching cannot be easily captured.

JetForm Corporation produces workflow product to provide functionalities for designing, filling, sending, printing and managing of forms. It is supposed to help organizations reduce costs and increase efficiency by automating forms processing. JetForm's Print and Output Solutions provide printing capabilities so that quality records and reports can be produced easily. It's Administrative Workflow Solutions allow organizations to automate their internal and administrative processes by defining the flow of documents in the course of workflow. In addition, JetForm's Web Solutions provide end-to-end applications to allow organizations to extend existing systems to Intranet, Extranet, or the Internet. The disadvantage with JetForm is that it is mainly document-driven. It does not focus on actor relationships and quality issues. Also, like all the other tools, it is not tailor-made for the software industry.

It should be noted that all these workflow products have different focuses about workflow. Lotus Notes is document and database oriented and workflow in the Notes environment mainly involves routing of documents and electronic mails. FlowMark is an object-oriented client-server workflow system that cannot be a stand-alone product. A backend database is required for use with it. Since application access to database is not modeled in FlowMark, its support for exception handling is limited. For InConcert, it is data-oriented and it does not address actor relationship and quality management. ActionWorkflow is based on speed act that focuses on communication patterns between actors. It is not developed for task completion and is too complex to model software processes since task sequence and conditional branching are difficult to represent in it. Lastly, JetForm focuses on document control in distributed environment. It does not capture actor relationship. Of all these tools, Lotus Notes is better because it has rich build-in database capabilities as well as electronic messaging functions. This is perhaps the reason why Notes is the most popular workflow and groupware tools now available commercially.

2.3. ISO and Workflow Products

Most of the ISO tools are not developed in respect of workflow. However, some attempts have been made to try to take such aspect into consideration. In particular, QMX Quality Management Software by DPI Services is a suite of integrated workflow-enabled Lotus Notes applications designed to facilitate certain aspects of ISO certification. QMX attempts to help users create, review and control quality system documents such as quality manuals, operating procedures and work instructions. It has built-in procedures to support the management of corrective actions and audit tracking and it allows remote users to participate in the software through Notes communication features. Changes to the documentation can be made remotely through replications. As most of the tools, it does not address relationships between actors. Also, exception-handling mechanism is absence.

Other than QMX, ISO Achiever Plus by Bywater and Triangle claims to be able to help companies to plan and implement a business process-based ISO 9000 quality system. It starts by providing a framework for the design, generation and management of documentation. It helps to create the quality manual, procedures, work instructions and

other controlled documents. Audits can be carried out against procedures and it stores corrective action requests as well as customer complaints, supplier information and training records. In addition, ISO Achiever Plus offers automatic routing of outstanding activity requiring attention. Since it focuses on business processes, much implementation effort is required when applied in the software industry.

In ISO Expert by Management Software International Inc., there are different components that contain the clauses, the forms, reports, templates, and processing logic (or workflow) needed to automate the management process. Templates of quality manuals, procedures and work instructions are provided for companies to document their own quality management system. Furthermore, the project management database provides a suggested implementation plan and the workflow management database allows the definition and control of document flow. ISO Expert provides also some audit questions and allows company to track audit results. However, it does not support task assignment and notification. Like the QMX, actor relationships and exception handling mechanism are not handled.

ISO Pro by Business Challenge Limited has five components which is designed for the implementation of ISO 9000. The seminar module introduces user to the ISO 9000 philosophy and background. The assessment module monitors the quality system through performance, measurement and analysis. The workshop module shows user how to prepare documentation by providing guidance and quality manuals, procedures and quality plans. The implement module contains elements of the implementation process. The configure module, users allow users to customize the system operations. The ISO Pro also prepares internal audit and management review schedules. Although it allows users to customize the system operations, it is not flexible enough to handle complex software processes.

From all the above, we notice that all of the ISO and workflow products have quality templates to facilitate the documentation of QMS and they have system audit facility to assess the current readiness of ISO 9000. As QMX Quality Management Software is a Notes application, it has build-in communication functions and supports remote database access. The ISO Achiever Plus is workflow-enabled and supports routing of outstanding activities. Also, customers, training and suppliers information can be stored in the system. For ISO Expert, it supports project management but lacks of corrective action. And, ISO Pro provides ISO training which other tools do not have. Unfortunately, none of them is tailor-made for the software industry. And, they do not address the importance of actor relationships and the handling of exceptions.

3. Design of The Workflow Automation Tool

3.1. Some Design Considerations

Quality management in general is concerned with managing the organization in order to improve the systems and processes so that continual improvement of quality can be achieved. In this project, we particularly focus on the ISO 9001. It presents a model of quality management activities based on which an organization can use to maintain a quality management system (QMS).

The ISO 9000 requires an organization's quality management system to be established. Each of the elements of a quality management system to be designed, developed, and maintained by the developer are identified, with the objective of ensuring that all products or services meet the requirements of contracts, purchase orders and other agreements. For software development, it involves complex and dynamic interactions of software processes that requires the engagement of many specialists for specific activities. Therefore, the quality of software products is dependent upon the processes used to develop the products. The lack of a well-defined process has thus a direct impact on the organization's quality and productivity. It is hence important that the procedures, tasks and the personnel involved in some complex tasks, such as software development tasks, be precisely defined, established and maintained. We believe that improving process is one of the pre-requisites for achieving the ISO 9001.

Having a QMS in place, both the culture and practices of the organization may be changed. Everyone will understand the principles of quality and have a focus on the quality perception of customers and users. By introducing practical quality techniques, organization will gain substantial benefits. Quality of products and services can be improved which leads to increased competitiveness and better customer satisfaction. Also, rework will be eliminated and costs will be lowered.

To facilitate quality management, we, therefore, propose to automate workflow. Workflow is a notation to describe processes that involve the coordinated execution of multiple tasks performed by different processing entities. It is a concept closely related to reengineering and automating processes in an organization. By capturing tasks in software process, for example, workflow can describe process requirements and task relationships for information system functionality and human skills.

Workflow management is a rapidly growing area that comprises of the automated coordination, control, and communication of work in processes. It coordinates people who interact to complete a process and manages the flow of work so that it is carried out quickly, and efficiency, producing the highest quality of end results. Also, it manages staff and resources in a way that ensures process definition is followed and provides the mechanisms for planning and controlling workflow.

The Workflow Management Coalition (WfMC) defines workflow as "The computerized facilitation or automated component of a process" [Casati et al, 1996]. A workflow management system is a system that completely defines, manages and executes workflow through the execution of software whose order of execution is driven by a computer representation of the workflow logic. Such a system can save time, reduce redundancy, and ensure that processes are followed correctly. Workflow management system ensures that all the information to do the associated work is readily available by routing documents to the relevant workers and make them easier to access. Also, it tracks how far the process has gone towards completion and who it is with at any point in time thereby speeding up and ensuring the appropriate coordination in all processes. Therefore, it is useful to support flexible and repeatable process automation and re-engineering so as to improve quality of processes. To achieve these, process modeling is useful in the analysis of existing processes and how they may contribute towards the design and implementation of new and improved processes.

3.2. The ISO 9000 Approach

International Standards for Quality Assurance ISO 9000 is a set of five universal standards for a Quality Assurance system. The standards allow companies to demonstrate that they have specific processes in place to maintain an efficient quality system. The most comprehensive of the standards is ISO 9001. It applies to industries involved in the design and development, manufacturing, installation and servicing of products or services.

The ISO 9000 approach enforces organizations to create and follow procedures. It requires a documented quality system. The documented processes and procedures together with the quality manual form the bulk of the documented quality system. As most software processes are amenable to change, the documents involved are subject to continued change throughout their lifetime. The need for formal understanding of software process has been critical to the success of the software. Proper documented processes can not only reduce supervision as people can understand what and how they should do instead of having supervisors telling them all the time, but also provide evidence of continued improvement throughout management review. The following points out the role of documentation in the ISO 9000 approach [*Gianluigi, 1993*]:

1. *Say what you do*

   A well documented quality systems and software covering all phases and processes is needed. That is, a comprehensive methodology must exist, cover all phases of the life cycle, including maintenance, and incorporate, besides the constructive tasks, the quality assurance activities.

2. *Record what you did*

   Records are kept of the execution of all quality-related activities and of the outcome of that execution. That means that record of all verification and validation activities; and software defect records are maintained.

Besides documenting and recording what the company do, companies in compliance with the ISO 9001 must ensure that they are doing what the documented procedures said and perform them in a controllable manner. It is important and essential to demonstrate that what the company is doing is efficient and effective. The quality system should be organized in such a way that adequate and continuous control is exercised over all activities affecting quality. And, documented operational procedures coordinating different activities with respect to an effective quality system should be developed, issued and maintained to implement the quality policy and objectives. The interpretation in process enactment can be synthesized as follows [*Gianluigi, 1993*]:

1. *Do what you say*

   The execution of quality related activities should be distributed and well understood throughout the organization. That is, processes are clearly assigned to the appropriate people, and the execution is monitored in order to ensure the conformance of the enacted process to the documented process.

2. *Review your records and act*

   There should be procedures for periodic reviews of the performance of the quality system and for the enactment of subsequent corrective actions. It is important to make sure that processes are executed properly, and if this is not the case, they are appropriately corrected.

3.2.1. Say what you do

Based on a standard groupware interface, WAT provides document manipulation environment through user-friendly interface. It integrates messaging, document storage and a rich application development environment that supports the sharing of all types of data across disparate networks and computing platforms. Also, WAT provides templates of quality manual and procedures in setting up a QMS to facilitate the preparation of ISO required documents. With well-organized databases, quality manual, procedures and records can be referenced and tracked promptly and easily. Table 3-1 and 3-2 list all the templates in quality manual and procedure databases respectively.

Company Policy
Quality Policy Statement
Distribution Revision and Re-issue
Organization - Authority and Responsibilities
Organization Chart
System Outlines
Contract Review
Document Control
Design Control
Purchasing
Purchaser-supplied Product
Product Identification and Traceability
Process Control
Inspection and Testing
Inspection, Measuring and Test Equipment
Inspection and Test Status
Control of Nonconforming Product
Corrective Action
Handling, Storage, Packaging and Delivery
Quality Records
Quality Audits
Training
Servicing
Statistical Techniques

Table 3-1 Templates in quality manual.

Writing Quality System and Work Procedures
Management Responsibility
Quality System
Contract Review
Design Control
Document and Data Control
Purchasing
Customer Supplied Product
Product Identification and Traceability
Process Control
Inspection and Testing
Inspection and Measuring and Test Equipment
Inspection and Test Status
Control of Non-conforming Product
Corrective and Preventive Action
Replication and Delivery and Installation
Quality Records
Internal Quality Audits
Training
Servicing
Statistical Techniques

Table 3-2 Templates in procedures.

Besides, WAT also allows the flow of work within processes in the organization to be specified through a graphical user-interface. It is able to capture information of organizational members and their relationships so that their responsibilities as well as the lines of command can be well defined and understood. Also, it is capable of recording information presentation details and supporting workflow automation and management by sub-models in the workflow model. To better model coordinated activities, it adopts an multi-perspectives approach to capture workflow either by process or actor and support transformation between them. These include well-defined elements, structure and representation scheme to address different dimensions of a process. In this way, it ensures that all information required to perform different tasks in a process is properly documented and be made readily available. In brief, WAT helps users to *say what they do* by providing: i) documentation templates to support preparation of quality manual and records; and ii) graphical user-interface to capture processes and actors information.

3.2.2. Do what you say

As manual implementation of processes is often time consuming and process cannot be easily managed and monitored, WAT employ workflow automation to help companies to do what they said and standardize process execution. In our approach, managing quality starts with automating workflow given the process is known, understood, and defined which is required by ISO 9001. WAT guides sequences of tasks that compose a process and perform automatic tasks that do not need human intervention. Implementation can begin with simple repetitive tasks, such as notification, tracking of status, and scheduling of assignments. Then, additional tasks to consider can be notifying users which task has been initiated when to perform a task, automatic task delegation and handling task exceptions. There is also a need to provide an intelligent document routing capability, as well as an ability to orchestrate access to the database and application tools needed to complete a given task. It is destined to improve organizational productivity and simplify access to conventional database applications. Combining it with electronic mail, sophisticated database support and document-based workflow applications, it is expected to truly take the place of traditional paper-based processes.

In our design philosophy, to provide an unified task management system for groups of people, a flexible mechanism of communication between the system and actors is required. It is seemingly that organizations need large-scale workflow solutions that can accommodate a growing need of distributed environment. It is because organizational and multi-departmental workflow applications are made up of several disparate processes that may include technologies that currently do not interoperate. For example, database or transaction-oriented workflow applications currently do not interoperate with messaging applications that are based on store-and-forward technology. With organizations becoming more distributed, they need to replicate and synchronize data and workflow definitions across departments and physical sites. Distribution is a main issue because workflow management is a distributed application by nature, and distribution opens the way to scalability of the architecture. This can be achieved through a client/server architecture. Messaging can accommodate many of these issues through a variety of technologies. For this reason, there should be a central repository to store the workflow details and process

instances in detail. Fig. 3-1 illustrates these by showing how our system supports and coordinates group work in the course of a workflow in a distributed environment.

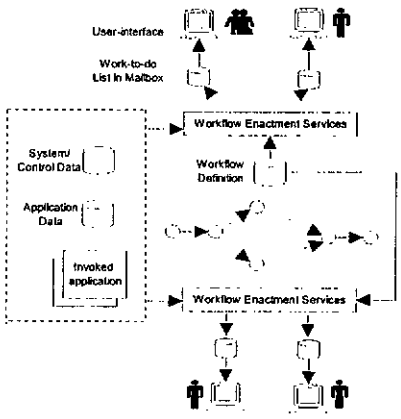


Fig. 3-1 Supporting group works in distributed environment.

In the implementation of WAT, our design is based on the above design philosophies. Workflow details are defined and stored in a central shared database to ensure consistency. Based on them, the workflow enactment services evaluate the routes and determine which actors are responsible for the next task. A task notification will then be sent to the mailboxes of the appropriate actors so that there is a universal inbox (tasks appear in the same inbox as mails) which eliminate the need to check multiple sources of work. In this way, actors can be aware of the task instant independent of their physical locations.

In the course of workflow, different system, control and application data may be required. Actors can retrieve these data through the workflow enactment services and invoke other applications when necessary. It is not only applicable locally, but also in a distributed environment. All workflow details in the organization can either be centralized on one server or distributed over multiple servers and workflow engine can instantiate a process instance based on the location of the definition. Alternatively, multiple workflow engines can be used. Each of them is responsible for part of the workflows. Process actors are not necessary to be restricted within a single domain. Sending task notifications to actors and keeping track of process status in other domains can be achieved by bridging multiple workflow engines. As a result, distributed workflow management across time and space can be supported.

Therefore, to help users *do what they say* and support group work, WAT provides them with numerous benefits including :

- workflow details operations to retrieve process information;
- work-to-do list handling functions for notification of assignment of a task;
- process status functions to keep tracks of a specific process or task;
- process maintenance functions to modify process and/or task definitions;
- data handling functions to retrieve workflow relevant system or control data;
- application integration to invoke other external applications;
- process control functions to raise exceptions when necessary.

Through the above capabilities, our system can keep track of the status of all processes that has been initiated. Users can check status of a particular process instance easily. Information such as the number of a particular process has been initiated, the total number of process that has been initiated on a particular date can be collected. Also, users can check process status from their own point of view and needs [*Yan and Chan 1998, Chan et al 1998*].

3.3. A Survey of Workflow Modeling Techniques

Recently, a number of workflow modeling techniques with different focuses have been proposed. For example, some researchers advocate the use of rules to model work processes [*Ciancarini 1995, Kappel et al 1995, Bussler and Jablonski 1994*]. Typically, rules are used in the specification of different conditions in mutually exclusive cases. Rules are also used to define possible automatic actions so as to trigger the initiation of tasks during a process. In some cases, rules are used to select responsible staff and to manage agent worklists [*Kappel et al, 1995*]. An advantage of rule based techniques is that the rules can be collected in a knowledge base to describe a work. Also, the declarative expressive power of rule-based languages can be an important technology for process specification, modeling, and coordination [*Ciancarini 1995*].

Object-oriented technology has also been used for workflow modeling [*Kappel et al 1995, Hartel 1995, Meichun and Charly 1996*]. A workflow management system on an object-oriented database was proposed in [*Kappel et al, 1995*]. The database provided some functionality for modeling and reusing complex process objects. Another example of the use of object-oriented technology was found in Object-Flow [*Meichun and Charly 1996*]. It supported an object-oriented specification that included different task types (static and dynamic compound tasks). In addition, there were some proposed approaches for the integration of concepts for the modeling of applications and business processes in cooperative information systems. A typical example was that of the framework of a formal object-oriented specification language (TROLL) described in [*Hartel, 1995*].

Over the past decade, many researchers model work processes on the basics of "activities" [*Gary 1994, Kari and Tuula 1992, Raul et al 1992, Gruhn 1995, Christoph and Stefan 1994, Thomas et al 1995, Gerhard and Stefan 1996, Rusinkiewicz and Sheth 1994*]. The Quality Process Language (QPL), for example, which supported ownership of processes, communication and compliance with requirements, was used to represent and analyze processes within an organization [*Gary 1994*]. Also, the Activity Theory that was proposed by [*Kari and Tuula 1992*] was used to identify potential computer supported cooperative work application. In addition, an action workflow approach proposed by [*Raul et al 1992*] was used as a design methodology. Based on the construction of action loops, a new approach to characterizing workflow was developed according to it. Other than these approaches, Gruhn discussed an approach to business process modeling and workflow management that was based on data modeling, activity modeling, and organization modeling [*Gruhn 1995*]. In a model developed at Bellcore, tasks were assigned different characteristics according to their transactional behavior and flow control was determined by scheduling conditions [*Rusinkiewicz and Sheth 1994*]. Similar activity based workflow model could also be found in [*Christoph and Stefan 1994, Thomas et al 1995, Gerhard and Stefan 1996*].

Although activity based models usefully abstract the sequence of work and information flow, they do not adequately explain how and why the actors communicate. Since organizations are viewed as being made up of actors who have motivations, it is suggested that processes be modeled based on actors [*Eric 1995, Kevin 1992, Gulla and Lindland 1994, Alassane et al 1996*]. A framework which supports formal modeling of network of dependency relationships among actors has been proposed [*Eric 1995*]. Also, an actor service model was developed to address actor relationships in coordination activities [*Kevin 1992, Jon and Odd 1994*]. It described how actors interact and cooperate in accordance with intentions and goals. Other than these work, there are also effort to develop system to analyze the action done by actors in their cognitive shared space, derived from a process model, through the events generated by their actions and interactions [*Alassane et al 1996*].

As workflow systems became recognized as a category of computer system, it was recognized that all workflow management products have some common characteristics, giving them the potential to achieve a level of interoperability through the adoption of standards for various functions [*Lawrence, 1997*]. It is important as standards enable interoperability between heterogeneous workflow products and improved integration of workflow applications with other IT services such as electronic mail and document management.

To avoid the potential for conflict in the development of standards, the Workflow Management Coalition (WfMC) has started early in the life of workflow technology. It is an organization of more than 170 members located in 24 countries around the world. It focuses on the advancement of the workflow management technology and its use in industry. One of its initial activities centered on groundwork for the definition of a universal reference model for workflow systems. As the WfMC has a wide coverage of workflow (characteristics, terminology and components of workflow management systems as well as the interfaces and information flows between the major functional components), it has been chosen as the basis of the model and specifications of WAT.

WIDE (Workflow on Intelligent Distributed database Environment) is one of the projects of the WfMC [*Casati et al, 1996*]. The WIDE consortium includes 5 partners from Spain, Italy and The Netherlands. They include the Sema Group sae (Spain) which is the software integrator and the technology providers which are represented by two well known university groups : Politecnico di Milano (Italy) and University of Twente (The Netherlands).

As part of the WIDE project, a rich conceptual workflow model has been proposed. Its objective is to extend the technology of distributed and active databases, in order to extend workflow capabilities of software products according to requirements that were set by collaborating individuals within organizations. The main goals are i) to define an advanced conceptual model for describing both the flow of activities and the organizational environment in which these activities are performed; and ii) to provide an advanced technological support to workflow management through advanced database systems in a distributed environment.

The WIDE workflow model is structured along three loosely coupled sub-models : organizational, process and information models. Flexibility was obtained by incorporating an independence between the three models it consists of, allowing modification of any one of these models without affecting others [*Casati et al, 1996*]. And, such flexibility that provided by the WIDE workflow model allows organizations to quickly anticipate on the ever-changing environment.

The organizational model describes the part of the organization involved in workflow execution in terms of persons, positions, organizational units, organizational roles and authorities [*Lawrence, 1997*]. It determines division of labor, communication and the assignment of organization. This model has sufficient expressive power to describe real enterprise organizational structures.

The information model captures the set of information managed by the workflow. It identifies and describes the documentation elements involved in a process [*Casati et al, 1996*]. Information used in workflows can either be defined at the workflow schema level by variables, based on databases which are shared among all persons involved in the workflow, or consist of documents exchanged through the workflow management system (WFMS).

The process model determines process steps (tasks) and their dependencies in a process. Tasks are the elementary work units that collectively achieve the workflow goal and process flow is modeled by interactions between tasks. The workflow engine determines when a certain task must start being executed, and of assigning it to agents, according to the task assignment rules [*Casati et al, 1996*].

The introduction of this model has improved and strengthened the specification of workflows at the conceptual level, by formalizing within a unique model the interaction and cooperation between tasks, the assignment of tasks to actors and the accesses to external databases. Examples of successful adoption of the model include the enhancement of FORO, an existing workflow management system supporting the management of enterprise-wide processes and their constraints in a cooperative, distributed environment, developed by Sema Group sae. Also, the ING Bank from Amsterdam and the Manresa Hospital (Manresa, Spain) are the major users of WIDE that enables the experimentation and evaluation in different application contexts.

For the above reasons, the WIDE workflow model has been chosen as the basis of our workflow model in implementing the workflow system. However, our workflow model has been enhanced to address relationships between people as well as task relationships. In order to provide a better understanding of workflow in an organization, it supports model transformation that is capable to capture relationships between people from task relationships and vice versa.

3.4. The Proposed Workflow Model

As previously discussed, a company say what it does is important in the ISO philosophy. This is closely related to the definition and execution of workflow. In this section, we will describe how WAT models workflow.

To model workflow, WAT address processes, tasks, dependencies among tasks, the actors that perform the tasks and forms used in the tasks. With some modifications and enhancements to the WIDE workflow model by the WfMC, WAT is able to capture all the workflow details by three models. The information model to identify and describe the documentation elements involved in a workflow. The actor model to capture actor relationships and facilitate assignment of tasks to actors. And, the process model to model cooperation and dependency among tasks. Workflow can be captured by either process or actor models and WAT supports transformation from one to another. This provides multi-perspective to capture and understand processes, actors and their relationships in an organization.

WAT defines objects and connectors to link objects in the process and actor models. In process model, connectors define task sequence and conditional task branching. While in actor model, connectors show different relationships between actors. For each type of connector, there are one or more attributes associated with it, which defines the properties of that connector. In the following sections, the information model, actor model and process model as well as the transformation between process and actor models will be discussed.

3.4.1. Information Model

The information model identifies the information objects involved in workflow. Information is identified here as useful data in processes. The object in this model is defined in WAT as a set of forms which is characterized by : form name, description, field labels and fields. Owing to the interactive nature of workflow system, there often involves the routing of physical documents and a lot of information are presented and captured using forms displayed on user-interface. System and workflow definition as well as control data are stored to facilitate the flow of process, such as the process definition, organization data, workflow control data and work-to-do list, etc. Forms are the documents or records used in the quality management system. For example, in an ISO 9000 compliance system, the various forms provide the ability to create the required QMS documentation and to record the actual level of quality performance via the records. After forms are defined, they are eventually linked to tasks and workflow participants will use information contained in them.

A form in WAT mainly consists of static texts and fields. Static texts are used as annotation and field labels. And fields are used to reference workflow data. With all the form elements defined and positioned, the layout of the form can be described. To enhance readability and enforce access control, part of the form can be grouped to form section. Sections can be collapsed or expanded under different circumstances and based on different users. This provides different focuses and facilitates quick information retrieval. For access-controlled sections, they can control edit right for different members in a group to prevent unauthorized editing.

3.4.2. Actor Model

In order for the organization members to perform within the environment as expected, it is essential to model these individuals. Flexibility is needed in an ever-changing organizational environment. It can be accommodated through the uncoupling of the actor model from the other models. Actors are the building block and processing entities of an organization in which tasks are passed around. Manual tasks can only be performed with the cooperation of actors. Interactions between actors are thus inevitable and essential in the quality of work. Therefore, it is valuable and interesting to investigate into the relationships and dependencies between actors in an organization. It will facilitate the understanding of how actors work together and help to characterize the working model and behavior of actors. Actor definition entails more than identifying people at the organization. For each person defined, WAT specifies his/her department, subordinates, superordinates and roles.

This model adopts the concept of role for specifying the responsibility for an actor to perform a given task. It is valuable in independence between actors and definition of tasks to support dynamic task assignment to actors with suitable role. In the actor model, WAT allows three types of actor relationship between roles to be defined : i) information sharing, ii) task cooperation and iii) task dependency.

INFORMATION SHARING

As discussed in the information model, information is useful data that are needed in tasks. They may exist in different forms, such as a physical document, a file or an electronic mail. Information sharing refers to the referencing of document content and knowledge among actors. It is the time for actors to retrieve information for them to accomplish certain tasks. For example, a system specification is shared among system analysts and programmers. Therefore, referencing the same document implies actors are doing the same or at least related tasks. And information sharing relationship is then useful in analyzing workflow.

TASK COOPERATION

Since several actors with different roles may engage in the same task, they are work in cooperation. For example, both project manager and system analyst may participate in project planning and scheduling. Therefore, if more than one actor is responsible for a task, we consider they have task cooperation relationship. As the number of involved actors in a task increases, the effort needed to ensure and control the quality of work increases as well. Also, if a actor work and cooperates with many other actors, it may imply an uneven workload distribution, insufficient human resources or extra on-the-job training is required. As a result, task cooperation relationship among actors can also provide meaningful and significant information in the current workflow.

TASK DEPENDENCY

In a workflow, there is well-defined possible ordering of tasks in a process. A task can only start after the completion of its preceding task(s). So, when actor(s) responsible for the coming task(s) can start working depends on when the actor(s) in the preceding task(s) finished their work. Our system considers the actor(s) in the coming task depend on actor(s) in the preceding task(s). For example, the programmer can start coding if the system analyst has finalized the design. It should be notice that task dependency has different scope from task cooperation. Task cooperation focuses on actors in a task while task dependency stresses on the sequence among consecutive tasks. This type of relationship is important especially in those time critical tasks.

3.4.3. Process Model

The process model supports the modeling of process in organizations. WAT captures process by a flowchart and process is modeled as a combination of tasks with well-defined sequences. This model expresses execution of tasks in a process and focuses on modeling of the work instead of the commitments among humans. Object in this model is a task. Apart from general attributes, such as task name and description, runtime relevant attributes may be defined for each task. For example, task type (manual, automatic), maximum time allowed, time unit (day, hour, minute), form created, modified and referenced, actors (in terms of roles) responsible for the task as well as actors that will be notified and method to handle exceptions (ignore, stop immediately, send error message, start recovery task). Runtime evaluation of these attributes allows tasks to be dispatched on a dynamic basis and allows exceptions to be raised. For example, the system handles manual and automatic tasks differently. Manual tasks must have at least one actor to perform the job while automatic tasks required an agent, an automatic element in our system, to get the job done. So, actor(s) will be informed in manual tasks and agent will start in automatic tasks. However, in both types of task, it is allowed to notify other actors for the initiation of that task. WAT navigates a process by evaluating task relationships. It allows three types of task relationship to be defined : i) sequential, ii) parallel and iii) conditional.

SEQUENTIAL RELATIONSHIP

Sequential relationship is the fundamental and simplest task relationship that involves consecutive execution of task. A series of sequential relationship forms a possible path in a process. It models working procedures that people pass the work to the other after they have finished their own jobs. It is mandatory in process model in order to represent the order of tasks and indicate the flow of work or data. For two tasks A and B connected sequentially, task B is ready to start only if task A completes successfully.

PARALLEL RELATIONSHIP

Parallel relationship involves simultaneous task execution. There will be more than one task in their active state. Usually, this kind of relationship is established between independent but cooperative tasks. Executing one of the tasks does not depend on others and will not affect others execution as well. If task A is connected to task B and C in parallel and task D follows, both task B and C are ready for execution after task A accomplishes. Then, task D can be initiated after both tasks B and C have been completed. This relationship supports task parallelism and tries to minimize investment of time.

CONDITIONAL RELATIONSHIP

Conditional relationship selects process execution path is based on conditions. This relationship splits process execution path into two or more. Although there may be multiple execution paths, only one of them can be in active state at a time depending on different circumstances. For instance, after executing task A, task B may follow at this time but at the next time, task C may start instead. A task-based workflow can have many conditional task relationships and the number of relationship determines the number of execution paths.

3.4.4. Transformation between Process and Actor Model

WAT supports transformation between process and actor models. Such a transformation enables the analysis of different aspects in an organization and accommodates a greater variety of organizational needs. Given a process model, with all the actors, form used and task sequences or branches defined, the system is able to construct respective actor model showing actor relationships if there is any as described in the previous sub-sections. On the other hand, we can also build a process model from an actor model. Respective task relationships can be deduced from well-defined actor relationships by specifying cooperated tasks, referenced forms and dependent tasks. In this way, WAT allows two different approaches to describe an organization and processes involved.

Specifically, in the process model, adding actors in a task creates task cooperation relationship and defining form referenced builds information sharing relationship among actors. That is, if there is more than one actor in a task, task cooperation relationship is built between each pair of actor. And, when several actors are responsible for a task and there is at least one referenced form, each pair of actor will have an information sharing relationship. For task cooperation, connector links two actors having task name as attribute. Whereas, for information sharing, connector attributes are the referenced form name and the task names that the two actors participate and reference the same form. These two relationships are non-directional. As there is no dependency between actors, connector indicates only a relationship between them.

However, task dependency is directional. When two tasks are connected, it is important to specify actors in the succeeding task depends on actors in preceding task. Apart from its directional nature, task dependency is further classified as conditional and unconditional. In sequential and parallel task relationships, task dependency is unconditional. And, in conditional task relationship, depends on which condition is matched in a particular instance of process, actor will work differently and different actors would be followed. Then, task sequence will change accordingly and connector attributes as well as values are thus varied. WAT supports conditions to be specified and evaluated in terms of agent, automatics event-driven tasks, and form value. Using agent to evaluate condition, the agent name and name of next task should be given. After successful running of the agent,

it returns a true/false value for the system to determine whether the next task should be started. For example, an agent called 'CheckResource' is used to check if all the resources are available. If a true value is returned, the task 'NotifyActors' will be started to inform all the responsible actors to perform their works. On the other hand, when condition is specified using form value, form name, field name, operator (includes, =, <>, >, <, >=, <=), field value and name of next task are required. It is used to check if the content of a particular field in a particular form satisfies a particular value so that a particular task can be followed. Here is an example of such a condition : if ***field*** *'severity'* in ***form*** *'problem report'* ***equals*** *(=)* to *high*, then ***task*** *'report to project manager'* starts. Similarly, different condition statements can be built to control execution path. Currently, WAT allows a maximum of five different conditions to be specified in each task. Conditions will be evaluated in their input sequence. Whenever a condition is satisfied, the next task for that condition will be followed and other conditions will not be evaluated.

In the unconditional case, connector specifies which actor depends on which actor and the names of preceding and succeeding tasks are included. Whereas in the conditional case, apart from all the unconditional attributes just described, condition for that dependency is specified as well. With all these information, an actor model can be built from a process model. The above is summarized as follows:

Task creation	Task connection
Assigning actors ⇒ Task cooperation **Actor 1** Task1, Task2 **Actor 2**	Unconditional task relationships ⇒ Task dependency **Actor 1** Task 1 ▶ **Actor 2** Task 3
Defining form referenced ⇒ Information sharing **Actor 1** form 1 **Actor 2** Task 1 form 2 Task 2, Task 4	Conditional task relationships ⇒ Task dependency **Actor 1** conditon **Actor 3** Task 1 statement ▶ Task 4

Through transformation from actor model to process model, task properties and relationships can be deduced from actor relationships. Defining task cooperation and information sharing relationships captures task properties. Actors are determined through task cooperation and forms referenced are defined when creating information sharing relationship. That is, whenever two actors have task cooperation relationship, they must be responsible for at least one common task. And, if there is information sharing relationship between them, they must reference at least one common form in the same or different task.

Through task dependency relationships, our system determines the sequence of tasks and possible conditional branches. Unconditional task dependencies (sequential and parallel task relationships) ensure a restricted task sequence that an actor must follow the actors in the preceding task. However, conditional task dependency only provides possible task dependency by defining different combination of condition and next task. Therefore, in different instances of the same process, different condition will be satisfied which leads to different task to be followed. Task sequences is then varied. Take the following transformation as an example. Given the actor model in Fig. 3-2, the process model as in Fig. 3-3 can be deduced.

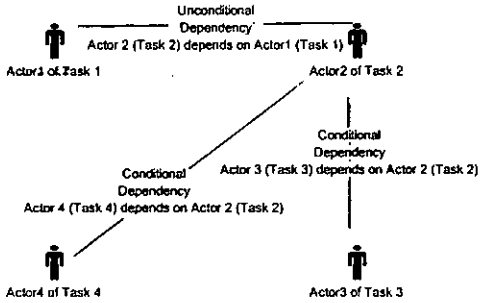


Fig. 3-2 An illustrative example of actor model.

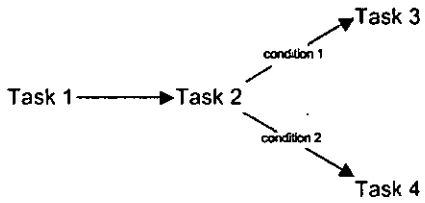
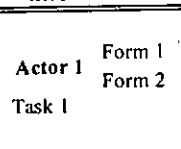
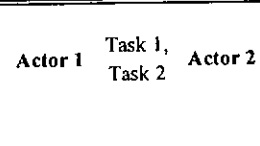
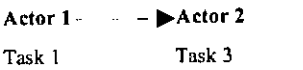
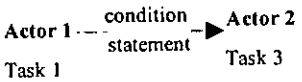


Fig. 3-3 An illustrative example of process model.

Actors in Task 2 must depend on Actor 1 in Task 1 because there is a sequential task relationship between Task 1 and Task 2. However, based on different conditions, either Task 3 or Task 4 will follow Task 2. So, if condition 1 is true, Actor 3 in Task 3 will depend on Actor 2 in Task 2. And if condition 2 is true, Actor 4 in Task 4 will depend on Actor 2 in Task 2 instead.

Connector attributes in actor model are similar to process model. Having all the task properties, ordering and branching, our system can build a process model from an actor model. The transformation algorithm is summarized as follows :

Information sharing	Task cooperation	Task dependency
 Actor 1 is an actor in Task 1 and Actor 2 is an actor in Task 2 and 4. Form 1 and 2 are the referenced forms of Task 1, 2 and 4.	 Actor 1 and 2 are one of the actors in Task 1 and Task 2.	 Actor 1 is an actor in Task 1 and Actor 2 is an actor in Task 3. Task1 can starts only after Task 3.  Actor 1 is an actor in Task 1 and Actor 2 is an actor in Task 3. Task 1 starts after Task 3 if the condition is satisfied.

3.5. Major differences between Our Model and WIDE

Although the WIDE workflow model has been chosen as the basis of our workflow model, there are number of differences between them. One of the major enhancements in our workflow model is the development of actor relationships. Most of the current workflow models, as well as WIDE, only address the importance of tasks. However, in our workflow model, apart from capturing task relationships, the importance of relationships between people (actors) is also addressed. Therefore, three types of actor relationship have been established, namely information sharing, task dependency and task cooperation. Moreover, in order to provide a better understanding of workflow in an organization, it supports model transformation that is capable to capture relationships between people from task relationships and vice versa. As a result, in a given task-based workflow, respective actor relationships can be deduced. Also, a set of actor relationships can be transformed to a task-based workflow [*Chan and Yan 1998, Yan and Chan 1998*].

WIDE model aims at modeling and implementing workflow in an organization. It does not address quality issues. In our workflow model, our objectives are not only model workflow, but also aims at building an ISO compliance quality management system. Therefore, our actor model supports different relationships to capture clear responsibilities and lines of command between organizational staff. Apart from that, our form model has also been enhanced. It captures the layout and information of form as well as linking forms to task. In this way, users are aware of different form(s) created/ modified/ referenced throughout the course of workflow [*Chan and Yan 1998, Yan and Chan 1998*].

Our workflow model not only focuses on actor relationships, but also aware of the importance of collaboration, cooperation and communication between group of people. For example, in our process model, it captures people who are responsible for accomplishing each task as well as other people that should be notified. In our implementation, automatic notifications will be sent to responsible staff and people who should be notified. In this way, workflow applications are able to bring responsible people to the correct task and maximize the value of the existing information by delivering them to the people who need them at the correct time [*Chan and Yan 1998, Yan and Chan 1998*].

4. Workflow Specification

4.1. Our Workflow Specification

Workflow specification is required to act as an interface between the two components of WAT : the workflow capturing component and workflow enactment component. To define a new workflow, actor and information specifications must first be imported from the workflow enactment component to the workflow capturing component. After the workflow has been defined, respective process specification can be generated from the workflow capturing component and import to the workflow enactment component. Our workflow specification references to but simplifies the WIDE specification. With similar approach to the WIDE specification and based on our workflow model previously described, our workflow specification is divided into i) actor specification, ii) information specification, and iii) process specification. In the following subsections, these three specifications will be presented and illustrated with examples.

4.1.1. The Actor Specification

For each actor in an organization, our specification captures the following attributes using the ACTOR definition :

1. *Name* : Fully distinguishable Lotus Notes user name of the actor
2. *Role* : Position which the actor holds
3. *Department* : Department or location that the actor works at
4. *Subordinates* : List of actor names that are his/her subordinate
5. *Superordinates* : List of actor names that are his/her superordinate

Each attribute is assigned to a data type specified after the attribute and separated with a colon. For example, 'name' is in string type whereas 'subordinates' is a list of string. Therefore, the complete actor definition is as follows :

```
ACTOR HAS
        name : STRING;
        role : STRING;
        department : STRING;
        subordinates : LIST OF STRING;
        superordinates : LIST OF STRING;
END_ACTOR;
```

Moreover, each actor is assigned to a role. All the roles in the organization are captured under the ROLE definition. For each role, there are a role name and description. For example, the role definition for a project director is like this :

```
ROLE Project Director
        desc : Manage and Control Projects;
END_ROLE;
```

With all these actor information well defined, the model can be populated with instances using the REGISTER ACTOR statements with all the attributes. Then, all the actors in the organization can be characterized. Given four sample actors in Table 4-1, the actor specification is shown as in Table 4-2.

Name	Role	Depart.	Subordinates	Superordinates
Ying/POLYU	Test Manager	ITS	User1/POLYU, User2/POLYU	
Keith Chan/POLYU	Project Director	Computing	Ka Fai Cheng/POLYU, Ricky Lee/POLYU	
Gloria Cheung/POLYU	Tester	Computing		Ricky Lee/POLYU

Table 4-1 Sample Actors.

```
ACTOR_MODEL DEVELOPMENT
        ACTOR HAS
                name : STRING;
                role : STRING;
                department : STRING;
                subordinates : LIST OF STRING;
                superordinates : LIST OF STRING;
        END_ACTOR;

        ROLE Project Director
                desc : Manage and Control Projects;
        END_ROLE;

        ROLE Test Manager
                desc : Unit and System Test;
        END_ROLE;

        ROLE Tester
                desc : General testing;
        END_ROLE;
END_MODEL

REGISTER ACTOR OF DEVELOPMENT
        (name, role, department, subordinates, superordinates)
        [(Ying/POLYU,Test Manager,ITS,[User1/POLYU,User2/POLYU] , [] ) ,
         (Keith Chan/POLYU,Project Director,Computing,[Ka Fai Cheng/POLYU,Ricky Lee/POLYU] , [] ) ,
         (Gloria Cheung/POLYU,Tester,Computing,[] , [Ricky Lee/POLYU] ) ];
```

Table 4-2 Actor Specification.

4.1.2. The Information Specification

For each form used in the process, our specification captures the following attributes using the FORM definition :

1. *Name* : The name of the form
2. *Description* : Detail description about the form
3. *Procedures* : List of procedure names that will create, modify or reference this form.

Similar to the actor definition, the form definition is formalized like this :

```
FORM HAS
        name : STRING;
        description : STRING;
        procedures : LIST OF STRING;
END_FORM;
```

In Lotus Notes, form design is completely hidden from other applications. It is so far impossible to obtain the layout of a Notes form (such as the text captions, field names and their respective position) and export to the other applications. Therefore, our information specification is different from the WIDE information specification. As in the actor specification, the information model can also be populated with instances using the REGISTER statements with all the forms attributes. For example, Table 4-3 shows all the forms used in the design process. And, Table 4-4 shows the information specification.

Name	Description	Procedures
Discussion Record	For Discussion	Document and Data Control, Process Control
Functional Specification	Detail description of functions	Design Control, Inspection and Testing
Technical Specification	Detail specification	Design Control, Statistical Techniques

Table 4-3 Forms used in the design process.

```
INFORMATION_MODEL Design
        FORM HAS
                name : STRING;
                description : STRING;
                procedures : LIST OF STRING;
        END_FORM;
END_MODEL

REGISTER FORM OF Design
        (name, description, procedures)
        [(Discussion Record,For Discussion,[Document and Data Control,Process Control] ) ,
         (Functional Specification,Detail description of functions,[Design Control,Inspection and Testing] ) ,
         (Technical Specification,Detail specification,[Design Control,Statistical Techniques] ) ];
```

Table 4-4 Information Specification.

4.1.3. The Process Specification

The process specification captures task relationships, as well as control flows and data flows in process. For each process, our specification captures the following as parameters :

1. *Name* : The name of the process
2. *Description* : Detail description about the process
3. *Database* : Name of application database used in the process

For example, the process specification of a software change request process implemented in database wf\demo is like this :

```
WORKFLOW_MODEL Software Change Request (name, description, database)
[Software Change Request, Standard procedures to handle change request, wf\demo]
```

Relevant actor and information model have to be imported into this specification using the USES statement. In the software change request process, actor model ReqActor and information model ReqForm are used. Therefore, the following statements are required :

```
USES ACTOR_MODEL ReqActor;
USES INFORMATION_MODEL ReqForm;
```

The START statement is used to specify the first task to in the process. That is, if "Initiate change request" is the first task in the software change request process, the START statement should like the following :

```
START Initiate change request;
```

Task is specified using the TASK definition, the following GENERAL information will be captured :

1. *Name* : The name of the task
2. *Description* : Detail description about the task
3. *Type* : Type of the task
4. *Time* : Maximum time allowed for the task
5. *Unit* : Time unit used

In addition, the following information will also be included :

1. *ACTOR* : List of actor names who is responsible for the task
2. *NOTIFY* : List of names that will be informed apart from the responsible actors
3. *FORM_CREATE* : List of form names that will be created in the task
4. *FORM_MODIFY* : List of form names that will be modified in the task
5. *FORM_REF* : List of form names that will be referenced in the task
6. *EXCEPTION* : Method to handle exception and respective argument
7. *COMPLETE* : How to proceed upon completion of the task. It is used to determine the control flow and task sequences.

For each type of information, the required arguments are included between a pair of bracket with comma as separator. And, the actual content of the arguments are listed in the same order as the arguments between a pair of square bracket and separated by comma. The following is the task specification of Initiate change request :

```
TASK Initiate change request
        GENERAL (description, type, precede, time, unit) [test, , 1, 30, Minute]
        ACTOR (actor list) [User]
        NOTIFY (actor list) []
        FORM_CREATE (form list) [Change Request Form]
        FORM_MODIFY (form list) []
        FORM_REF (form list) []
        EXCEPTION (method, argument) [Ignore, ]
        COMPLETE (method, argument) [Start Single Task, Approve change request]
END TASK;
```

Except COMPLETE, the number of arguments for all information is fixed. Based on the workflow captured in the workflow editor, after the completion of a task, there are four possible ways to proceed : i) End Process; ii) Start Single Task; iii) Start Single Task Based on Condition and iv) Start Multiple Tasks. Therefore, the number of arguments will vary. No argument is required for End Process. Like this :

```
COMPLETE (method, argument) [End Process, ]
```

For Start Single Task and Start Multiple Task, the task name or list of task names is need. The number of arguments in Start Multiple Task depends on the number of tasks that run in parallel. For example :

COMPLETE (method, argument) [Start Single Task, Approve change request]
COMPLETE (method, argument) [Start Multiple Task, Schedule change request, Schedule UAT]

For Start Single Task Based on Condition, the number of conditions and the detailed conditions are required. Conditions are specified by agent or form value as described in Chapter 3. And, they are included inside a pair of bracket. For example, there are two possible ways to proceed :

COMPLETE (method, argument) [Start Single Task Based on Condition, 2,
(By Form, Change Request Form, status, =, disapproved, Archive change request),
(By Form, Change Request Form, status, =, approved, Impact analysis)]

The following example is used to illustrate the complete workflow specification described above. Given a workflow as in Fig. 4-1, its workflow specification is shown in Table 4-5.

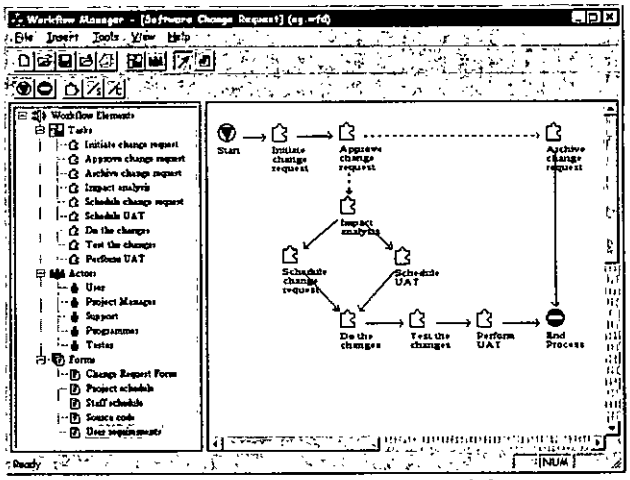


Fig. 4-1 A sample process model.

```
WORKFLOW_MODEL Software Change Request (name, description, database)
[Software Change Request, Standard procedures to handle change request, wfdemo]
USES ACTOR_MODEL ReqActor;
USES INFORMATION_MODEL ReqForm;

START Initiate change request;

TASK Initiate change request
   GENERAL (description, type, precede, time, unit) [test, , 1, 30, Minute]
   ACTOR (actor list) [User]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Change Request Form]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Approve change request]
END TASK;

TASK Approve change request
   GENERAL (description, type, precede, time, unit) [test, , 1, 2, Day]
   ACTOR (actor list) [Project Manager]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Change Request Form, status, =, disapproved,
            Archive change request), (By Form, Change Request Form, status, =, approved, Impact analysis)]
END TASK;

TASK Archive change request
   GENERAL (description, type, precede, time, unit) [test, , 1, 2, Day]
   ACTOR (actor list) [Support]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [End Process, ]
END TASK;

TASK Impact analysis
   GENERAL (description, type, precede, time, unit) [test, , 1, 4, Day]
   ACTOR (actor list) [Support]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Multiple Task, Schedule change request, Schedule UAT]
END TASK;

TASK Schedule change request
   GENERAL (description, type, precede, time, unit) [test, , 1, 1, Day]
   ACTOR (actor list) [Project Manager]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Staff schedule, Project schedule]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Do the changes]
END TASK;
```

Table 4-5 Process Specification.

```
TASK Schedule UAT
   GENERAL (description, type, precede, time, unit) [test, , 1, 1, Day]
   ACTOR (actor list) [Project Manager]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Staff schedule]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Do the changes]
END TASK;

TASK Do the changes
   GENERAL (description, type, precede, time, unit) [test, , 2, 10, Day]
   ACTOR (actor list) [Programmer, Support]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form, Source code]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Test the changes]
END TASK;

TASK Test the changes
   GENERAL (description, type, precede, time, unit) [test, , 1, 8, Day]
   ACTOR (actor list) [Programmer, Tester]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Source code, User requirements]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Perform UAT]
END TASK;

TASK Perform UAT
   GENERAL (description, type, precede, time, unit) [test, , 1, 5, Day]
   ACTOR (actor list) [User]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [End Process, ]
END TASK;
```

Table 4-5 (Cont'd) Process Specification.

5. System Implementation

5.1. The Workflow Capturing Component

The workflow capturing (WC) component of WAT is designed based on object-oriented concepts and implemented using Visual C++. It consists of a workflow editor which provides a easy-to-use graphical interface for the definition of objects such as processes, tasks, actors, forms and the connections between them.

In order to define a workflow, a user should start the WC component by clicking on the icon shown in Fig 5-1. Once the user brings up the WC component, he/she will be given access to the workflow editor (Fig. 5-2). With the workflow editor, the user can either define a new process or modify a previously defined one.

Fig. 5-1 Icon of the WC component.

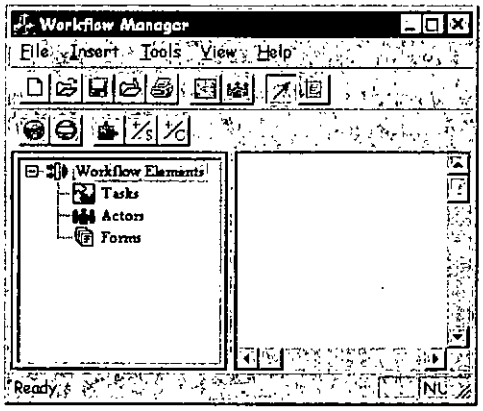


Fig. 5-2 The workflow editor.

To define a new process, the user will be asked to give the process a name and a brief description. He will also be asked to give the name of the database in which this process is linked to (Fig. 5-3). The database, which is maintained by Lotus Notes, contains the implementation logic and design of forms in which the process may need to create, modify or refer to. Then, the WC component will prompt user to input the actor and form files, which are the actor and information specification generated from the WE component (Fig. 5-4 and Fig. 5-5). It is because actors are registered and forms and designed in the WE component. Based on the specifications, a lists of actors and forms may be involved in the process will be imported in the WC component for the purpose of process definition. An example is given in Fig. 5-6.

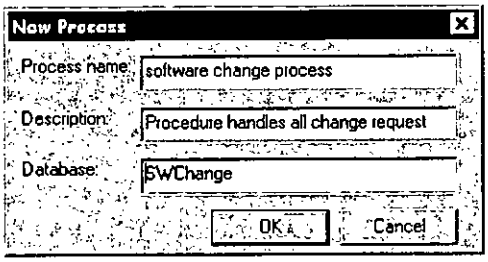


Fig. 5-3 Defining a new process.

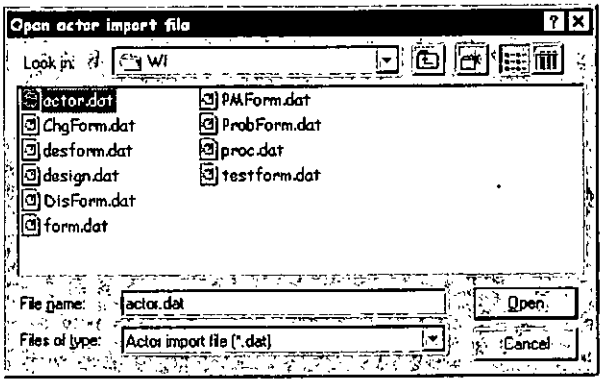


Fig. 5-4 Importing actor file.

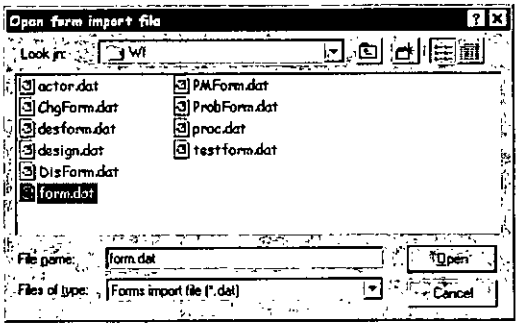


Fig. 5-5 Importing form file.

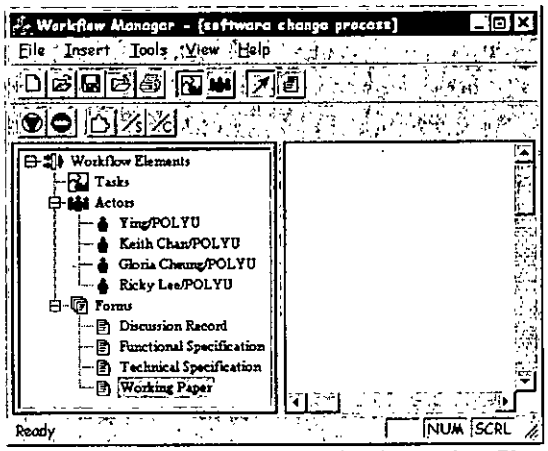


Fig. 5-6 The workflow editor after importing files.

If a user needs to modify a previously defined process instead of creating a new one, a user can open a workflow file defaulted to have type "wfd". An example of it is given in Fig. 5-7. The file change.wfd contains a previously defined change request process. Once it is opened, the features of the workflow editor become more obvious. Based on this example, we discuss these features in the following.

The workflow editor is divided into two panels as in Fig. 5-7. The left-hand panel shows the workflow elements of tasks, actors and forms. The nodes on the left-hand panel can be expanded and collapsed when the user double clicks on them. If it is expanded, a user can view all the currently available tasks, actors and forms. By double clicking on any of these workflow elements, or by clicking the right button to bring up a shortcut menu, a user can bring up the details of the properties associated with it (Fig. 5-8). Once this shortcut menu is brought up, the user can choose to view or modify the properties of the element or delete them from the process. A user can also delete a workflow element by selecting it in the left panel and pressing the DEL key.

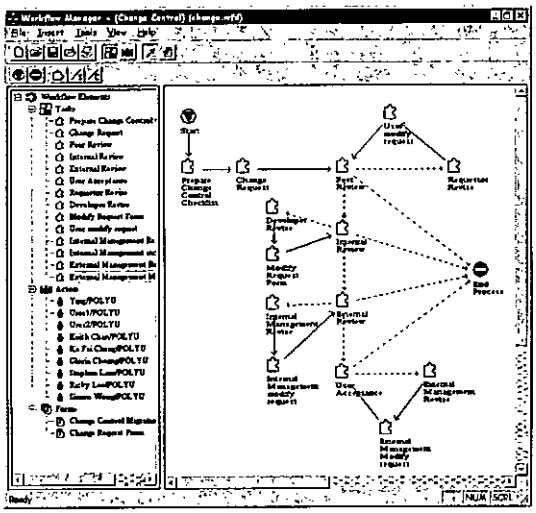


Fig. 5-7 A previously defined change request process.

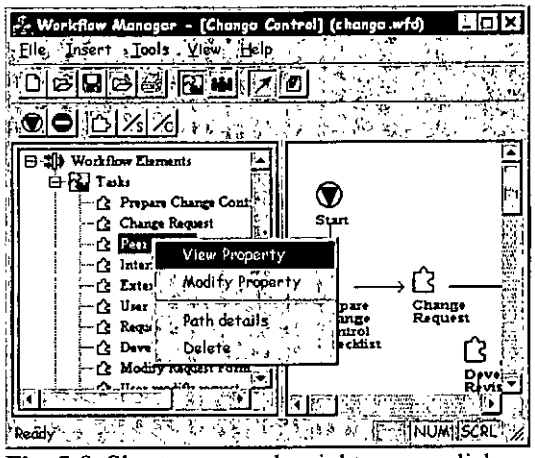


Fig. 5-8 Shortcut menu by right mouse click.

The right hand panel, as shown in Fig. 5-7, shows a task-based view of a process. In this view, users can find several different types of icons. They can use the start and end icons (Entries 1 and 2 respectively in Table 5-1) to indicate the beginning and end of a process. The task icon (Entry 3 in Table 5-1) is used to create tasks and the "arrows" icons (Entries 4 and 5 in Table 5-1) establish sequential and conditional relationships between the tasks.

Entry	Icons	Explanations
1		Create the Start icon
2		Create the End icon
3		Create a new task
4		Establish unconditional (sequential or parallel) task relationship
5		Establish conditional task relationship
6		Create a new workflow
7		Open an existing workflow
8		Save the current workflow
9		Close the existing workflow
10		Switch to task view
11		Switch to actor view
12		Selection pointer
13		Create a new form

Table 5-1 Icons in the toolbar of the task view.

To create a new task, a user can simply click on the task icon on the toolbar and then click on any place on the right panel. Once the user releases the button, he/she will be prompted for the detailed attributes of the task. Fig 5-9 shows the dialog box containing six different tab pages on which the user can specify the task attributes. In our system, the attributes of all the tasks are group into six different categories and the six tab pages are organized accordingly. These six tab pages are : "General", "Forms Created", "Forms Modified", "Forms Referenced", "Actors" and "Exception Handling". The "General" tab page allows users to enter task name, a brief description of the task, task type, expected maximum duration (see Fig. 5-10). The "Forms Created" tab page allows users to select from a list the form(s) that is or are to be created during the task (see Fig. 5-11). The "Forms Modified" tab pages enables users to select from a list the form(s) that is or are to be modified during the task (see Fig. 5-12). The "Forms Referenced" supports users to select from a list the form(s) that is or are to be referenced during the task (see Fig. 5-13). The "Actors" tab page allows users to select from a list of actor(s) that is or are responsible for the task (see Fig. 5-14). And, the "Exception Handling" tab page helps users to specify how the task will be proceeded when exception is raised.

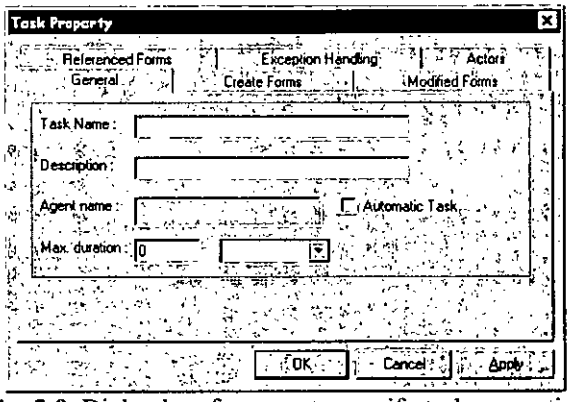


Fig. 5-9 Dialog box for users to specify task properties.

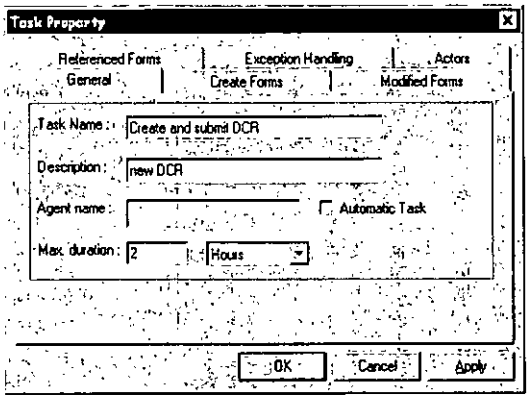


Fig. 5-10 General tab page of task property.

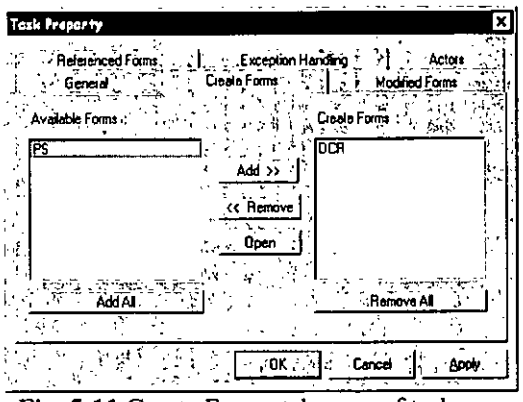


Fig. 5-11 Create Forms tab page of task property.

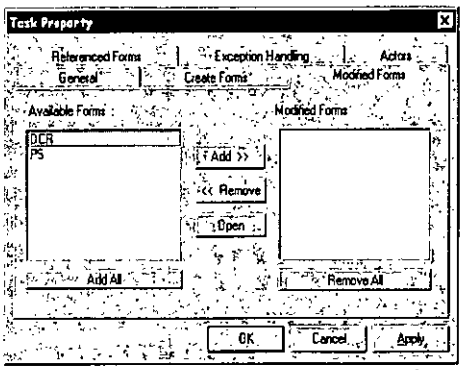


Fig. 5-12 Modified Forms tab page of task property.

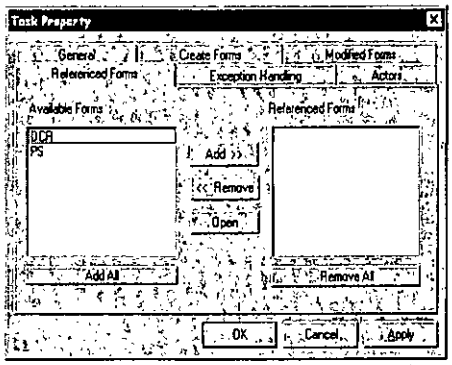


Fig. 5-13 Referenced Forms tab page of task property.

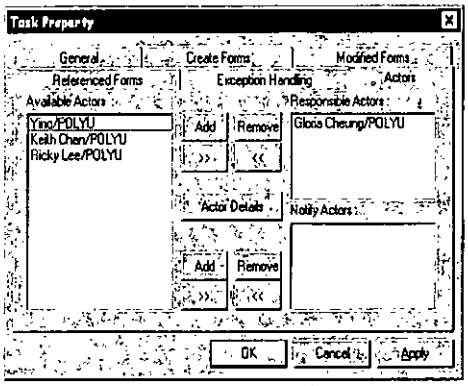


Fig. 5-14 Actors tab page of task property.

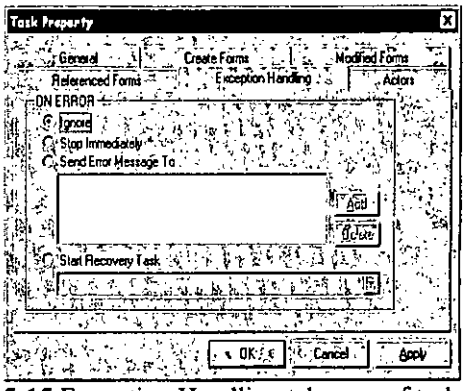


Fig. 5-15 Exception Handling tab page of task property.

Once entered, a new task icon will be added in the right panel and the user can move the icon around by drag and drop. Also, task properties can be viewed or modified by double clicking on the task icon or by right clicking on it to bring up the shortcut menu. The task properties can only be modified through the tab pages associated with the task icon.

To specify the flow of work from tasks to tasks, a user can define the relationship between them. The relationship between two tasks is sequential if one task is to be executed after the other unconditionally. To define such a relationship between two tasks, a user can first click on the "solid-arrow" icon on the toolbar (Entry 4 in Table 5-1). Then, user can click on the task icon of the one that is to be executed first and then click on the icon of the one to be executed after. The relationship between two tasks is conditional if one task is executed after the other only when a certain condition occurs. In such case, to define a relationship between two tasks, a user can click on the "broken-arrow" icon (Entry 5 in Table 5-1). And then, user should click on the task icon of the one to be executed first and then click on the icon of the one to be executed after when some conditions are met. As soon as the user releases the button, a dialog box will be created requesting the user to give the conditions under which the second task should be executed. Condition can be specified

by agent or form value as mentioned in the previous chapter. Fig. 5-16 shows a sample dialog box of conditional task relationship. This condition is evaluated by a form called 'DCR' and when the field 'continue' equals (=) to 'adopt', the result is true. Otherwise, the result is false.

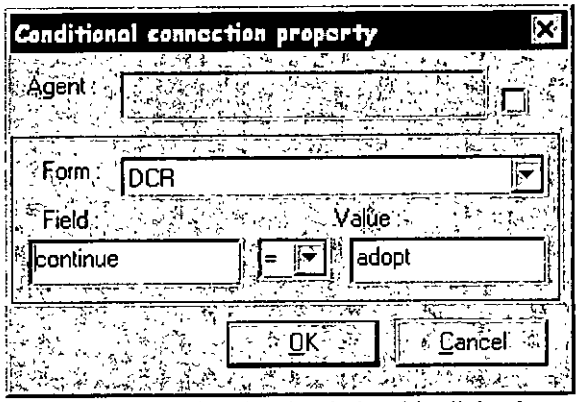


Fig. 5-16 Conditional task relationship dialog box.

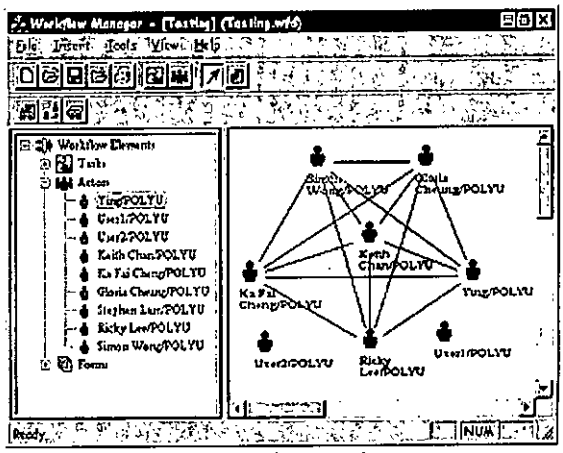


Fig. 5-17 An example of actor view.

As discussed in the previous chapters, a user can swap between the task view and the actor view any time during editing. To switch from the task view to the actor view, a user can click on the actor view icon (Entry 11 in Table 5-1). Once in the actor view, the user will see different icons corresponding to the different actors on the left-hand panel (see Fig. 5.17).

The actor property dialog box is shown from Fig. 5-18 to Fig. 5-20. The three tab pages includes : i) "General" on which a user enters actor name, title, department and task involved (see Fig. 5-18); ii) "Subordinates" on which a user select from a list of the actor(s) that is or are the subordinates of the current actor (see Fig. 5-19); and iii) "Superordinates" on which a user select from a list of the actor(s) that is or are the superordinates of the current actor (see Fig. 5-20). And, Fig. 5-21 is a sample of form property dialog box. It has only one tab page that is "General". On this page, a user can specify the title, description and related procedures of the form.

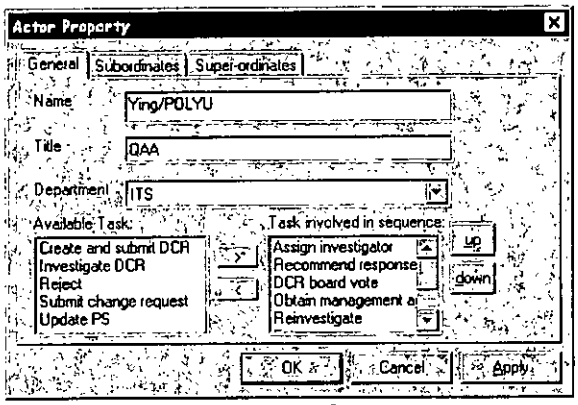


Fig. 5-18 General tab page of actor property.

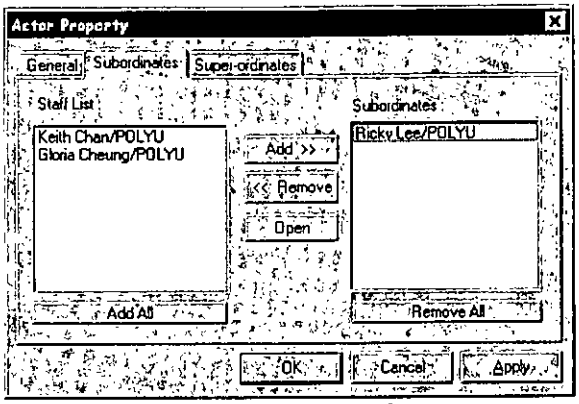


Fig. 5-19 Subordinates tab page of actor property.

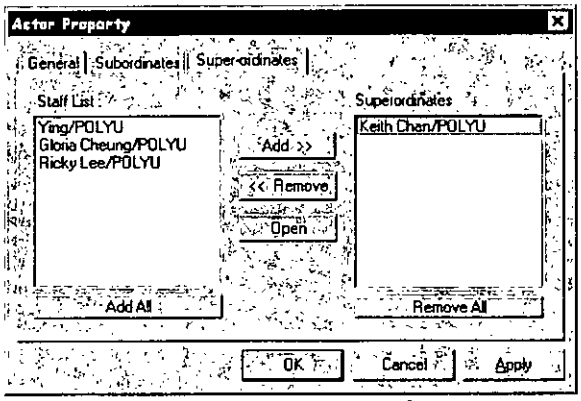


Fig. 5-20 Superordinates tab page of actor property.

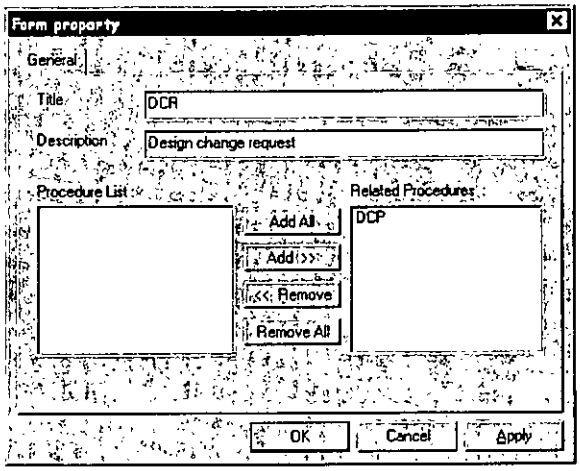


Fig. 5-21 General tab page of form property.

Some of the actors may be connected if they are defined in the task view to i) be responsible for the same task; ii) depend on the other; or iii) reference the same form(s). Fig. 5-22 to Fig. 5-25 are examples of tab pages in an actor relationship dialog box. This four tab pages are : i) "Task cooperation" on which it lists all the available tasks in the current process and all the tasks that the two actors are involved (see Fig. 5-22); ii) "Task dependency" on which they indicate which actor depends on which actor and their corresponding tasks (see Fig. 5-23 and Fig. 5-24); and iii) "Information sharing" on which it lists all the available forms in the process and those are referenced by both actors in different tasks (see Fig. 5-25). And, Table 5-2 lists three more icons in the toolbar in this view for users to specify these kinds of actor relationships.

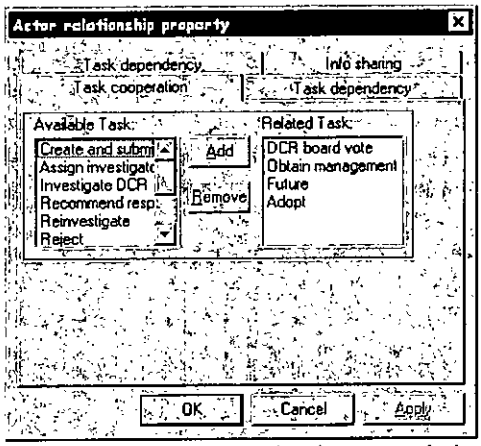


Fig. 5-22 Task cooperation in actor relationship.

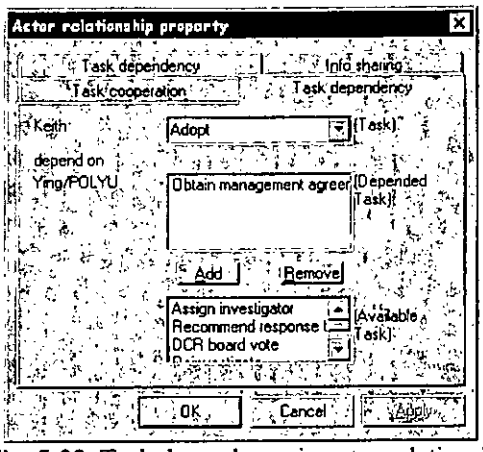


Fig. 5-23 Task dependency in actor relationship.

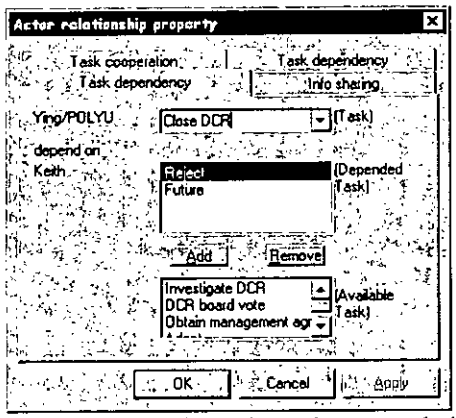


Fig. 5-24 Task dependency in actor relationship.

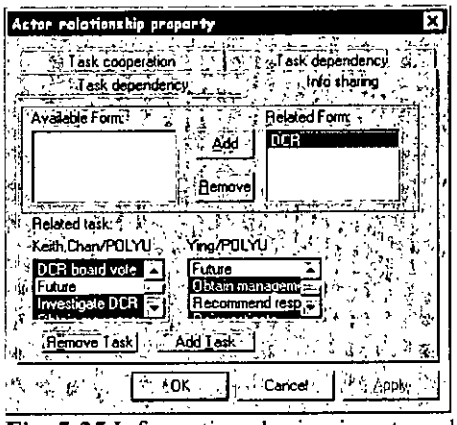


Fig. 5-25 Information sharing in actor relationship.

Entry	Icons	Explanations
1		Create information sharing relationship
2		Create task cooperation relationship
3		Create task dependency relationship

Table 5-2 Icons in the actor view.

Since all valid users must first be created in Lotus Notes and all actor information will then imported to the WC component, therefore, users need not create a new actor in the workflow editor. In the actor view, users can check whether actors have any kind of relationship between them and if so, how they are related. As all these actor relationships are derived from the workflow defined in the task view, they will be changed accordingly when the properties or sequence of tasks change. As a result, users are able to view updated actor relationships. In order to support different viewpoints and requirements of different users, users can drag actor icons around in the actor view as if task icons in the task view.

To specify the relationships between actors, a user can define the links between the actor icons. As described in the previous chapter, two actors have information sharing relationship if they reference the same form in the course of the process. To define such a relationship between two actors, a user can first click on the "information sharing" icon on the toolbar (Entry 1 in Table 5-2) and then click on the actor icon of the one of the actors and then click on the icon of the other. As soon as the user releases the button, a link will be created between the two actors. User can select the link and invoke the shortcut menu by the right mouse click to specify the name of the form(s) shared and the name of the tasks which each actor is responsible for. Similarly, two actors have task cooperation relationship if they work in the same task. In such case, to define a relationship between two actors, a user can click on the "task cooperation" icon (Entry 2 in Table 5-2), and then click on the actor icon of the one of the actors and then click on the icon of the other. To specify the task cooperation properties, user can use the shortcut menu to give the task names(s) that both actors are responsible for. When one actor works after the actor, they have task dependency relationship. In this case, a user should click on the "task dependency" icon (Entry 3 in Table 5-2), and then click on the actor icon of the one to start working first and then click on the icon of the one to work afterwards. Through the shortcut menu, user can specify the name of the task(s) that start first and later.

After viewing or modifying the actor relationships, user can click on the task view icon (Entry 10 in Table 5-1) to switch back to the task view. And when the flow of work and all actor relationships have been confirmed, user can export the process specification, which will be imported and used in the workflow enactment component later, by the 'File - Export' menu.

5.2. The Workflow Enactment Component

The workflow enactment (WE) component is an application implemented on top of Lotus Notes Release 4.6. This is to take advantage of some of the unique features of the groupware. In addition to the provision of a medium for communication, Lotus Notes also provides an easy way to track processes and a secure environment for the sharing of information.

Lotus Notes has a document oriented data model for compound multimedia documents, and a Notes application typically consists of documents, views, and forms [*Reinwald and Mohan 1996*]. In Notes, documents are used to refer to data stored in databases, views are used to refer to lists of documents taken from different viewpoints, and forms are used to refer to the interfaces for the display and editing of the documents.

For enactment of the workflow defined in the WC component, we have to make use of several Notes features including *links*, *agents* and *LotusScripts*. As far as *links* is concerned, Notes has three different types : document links, view links and database links. Document links are used to link a Notes document to another Notes document in the same or in a different database. These links allow users to have direct access from one document to another most effectively. The document link concept used in Notes can be considered as 'call by reference' in programming language design. In a similar manner, Notes also provides view links and database links to link together views and databases respectively.

Notes agents are event-driven automation design elements. They are programmable and can be programmed in such a way that they can be triggered automatically to accomplish pre-defined tasks when certain events occurred. Lotus Notes allows users to decide when to start an agent, which documents the agent should act on and what tasks the agent performs. Agents can also be executed manually, according to schedules or whenever certain changes occur. Based on different combinations of criteria and settings, Notes agents can be used to extend an application's workflow capability.

LotusScript is an object-oriented basic-like language. It can be used for the programming of agents or applications in response to user actions. For script programming, Notes provides a set of object class libraries that are accessible at script-level. In addition, it also provides OLE2 automation support and an integrated environment for development and debugging purposes [*Reinwald and Mohan 1996*].

With these Lotus Notes features, the WE component was constructed. It uses such features as links, agents and scripts in forms and databases and it consists of six core databases: (i) the quality manual database; (ii) the quality procedures database; (iii) the organizational structure database; (iv) the workflow specification database; (v) the application database; and (vi) the users' mailboxes. Fig. 5-26 shows these databases in the Notes environment.

Access to these databases can be made through double clicking on the respective icons. For example, by double clicking on the quality manual database, users can view the quality manual as shown in Fig 5-27. The window is divided into the left and right panel. In the left panel, there are nodes that users can double click to expand or collapse the different views in the database. By expanding on a node, the users can view information displayed in different logical format. If a user clicks on any view, e.g. on "Folders and Views", all the documents in the respective view (documents which match the selection criteria of that particular view) will be displayed in the right panel. By double clicking a document/row, e.g. when "1.3 Organization - Authority and Responsibilities" is double clicked, the document will be opened so that users can see the details as in Fig. 5-28.

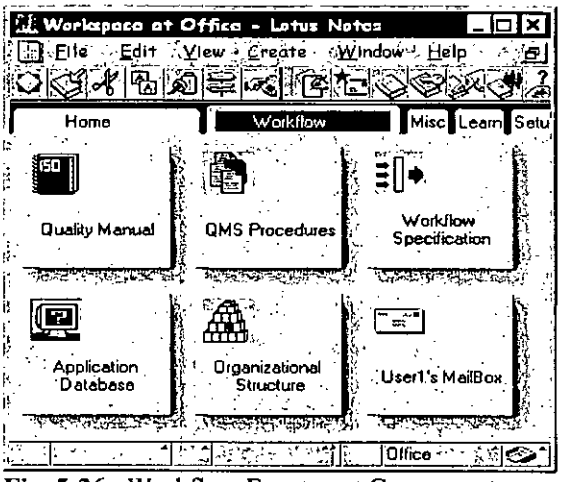


Fig. 5-26 Workflow Enactment Component.

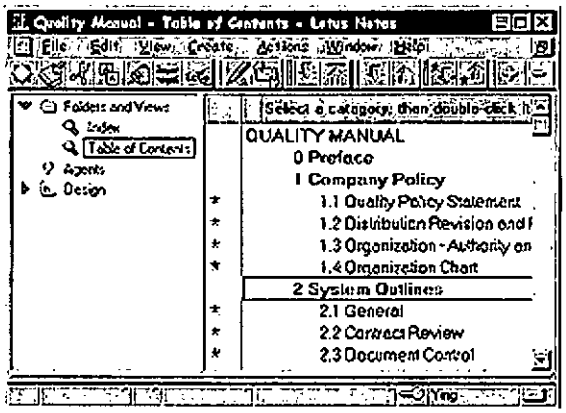


Fig. 5-27 Quality Manual view.

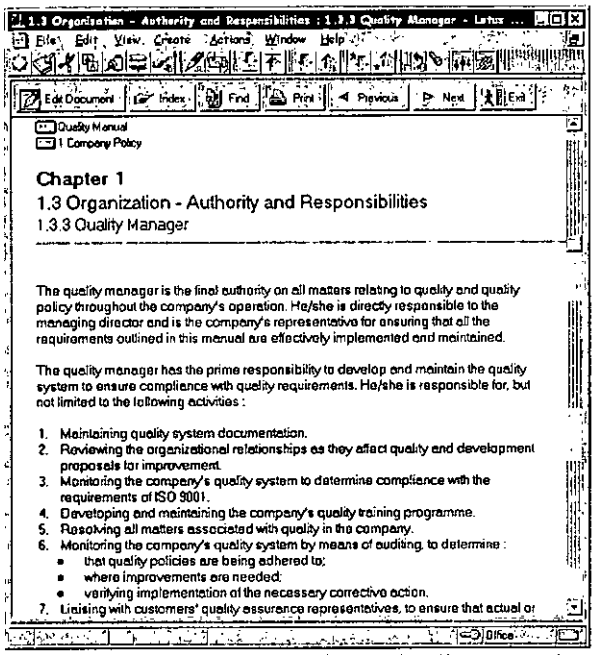


Fig. 5-28 Sample document in the Quality Manual.

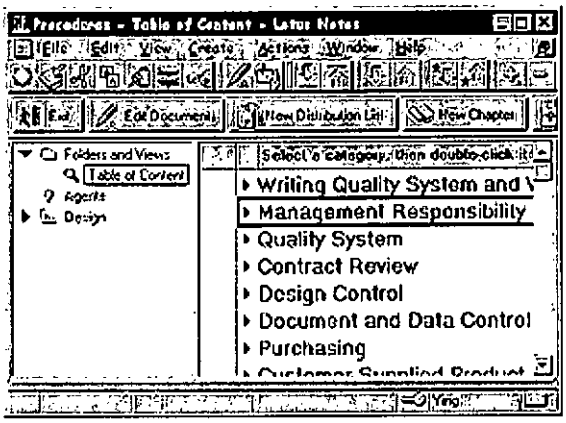


Fig. 5-29 QMS Procedures view.

Similarly, by double clicking on the icon of the "QMS Procedures" database (see Fig. 5-26), users can view all the procedures in the organization. Again, by expanding and collapsing the nodes on the left panel as shown in Fig 5-29, users can view document containing different procedures and users can click on a triangle in the right panel to expand or collapse that category to see all documents belonging to that category. An example of document in the Procedures database is shown in Fig. 5-30. Note that the tool bar contains icons for users to edit the current procedure, find keywords, print out the current procedure, and go to previous and next procedures as well as exit from the current procedure.

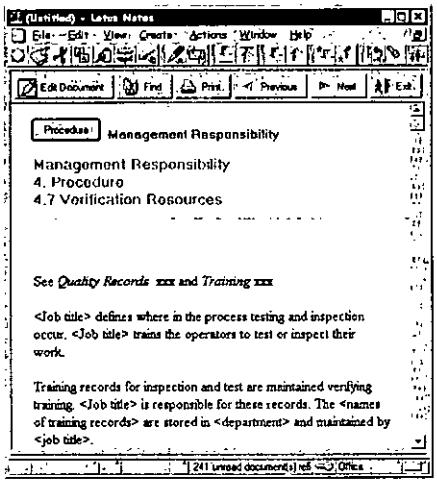


Fig. 5-30 An example of document in Procedures.

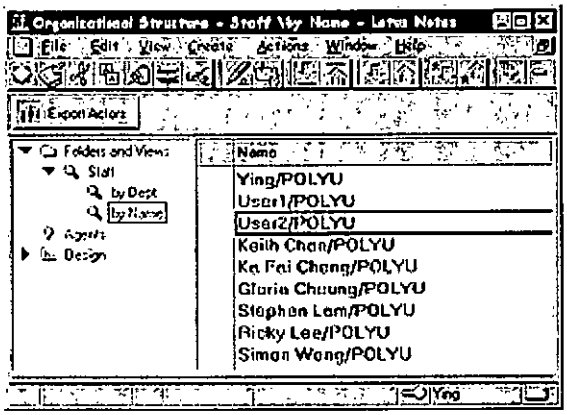


Fig. 5-31 Actor view in Lotus Notes.

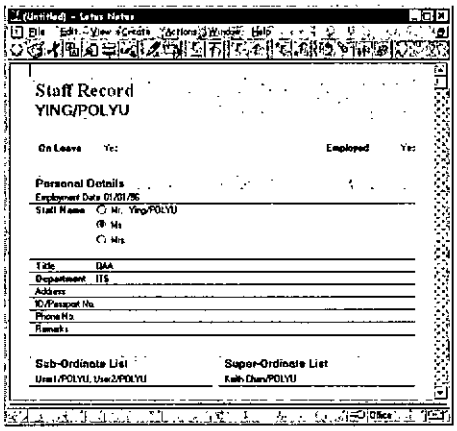


Fig. 5-32 An example of document in Organizational Structure.

By double clicking on the icon for the "organizational structure" database, users can view all the actors in the organization. As shown in Fig 5-31, users are given both the left and right panel. As you double-click on a name in the right panel, a staff record as shown in Fig. 5-32 is displayed. It is an example of document describing an actor in the organizational structure database. As described before, users are required to export actor specifications from the WE component to the WC component for the definition of a workflow and the organizational structure database is the place where actor specifications are generated. To facilitate the export of the specification, we provide users with an "Export Actor" button on the toolbar so that they can export actor specifications from any view in the database.

The workflow defined in the WC component are imported in the workflow specification database of the WE component in the form of a set of workflow specifications described in the previous chapter. For security purpose, only process administrator (a special role defined in the workflow specification database) can import workflow specifications into the WE component. Once a user double-clicks on the workflow specification database icon, the user will see its contents (Fig. 5-33). The specification includes high-level process information as well as low level task specifications. The view lists all the processes in the organization with all the tasks that belong to each process. By clicking the triangle next to a process name in the right panel, the process can be expanded and collapsed to show and hide all the tasks in that process. To see the detailed definition of a process or task, users can double click the selected process/task document in the view. Fig. 5-34 and Fig. 5-35 show an example of a process and task definition respectively.

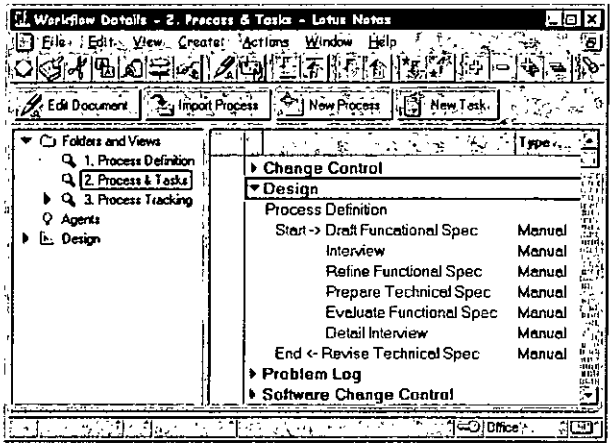


Fig. 5-33 Workflow Specification in Database.

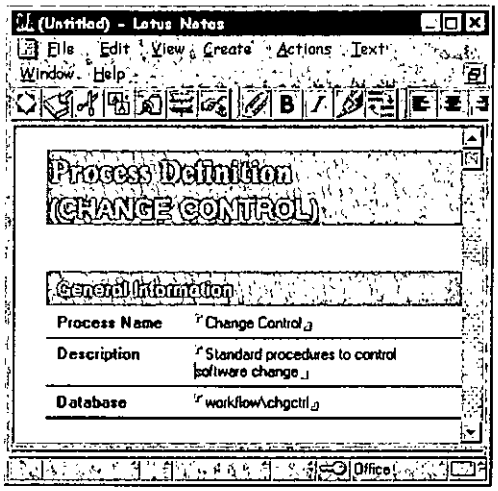


Fig. 5-34 An example of process definition.

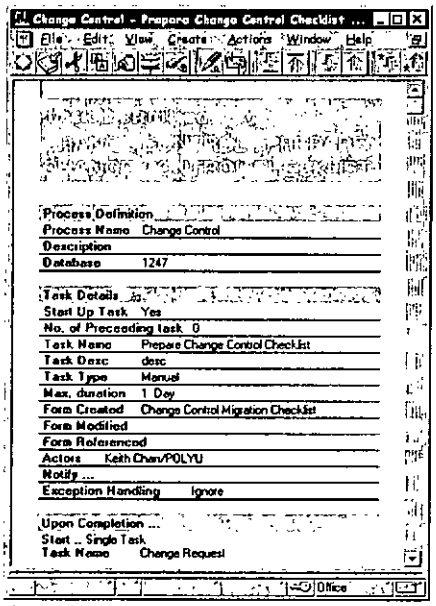


Fig. 5-35 An example of task definition.

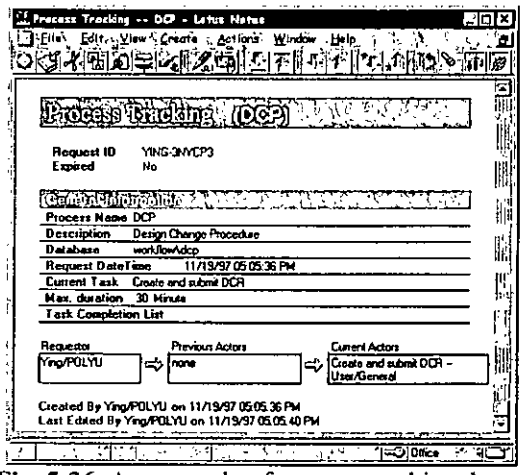


Fig. 5-36 An example of process tracking document.

Having workflow specifications defined, instance of the process can be initiated by the process owner (another role of the workflow specification database) whenever required. Upon process initiation or task completion, WAT generates a process tracking document for that process instance. It is a Notes document that contains all the necessary information to start the process. Also, a unique reference number called Request ID will be created and stored for each process tracking document for document indexing and searching. In the course of the process, this document will be updated and additional information will be inserted accordingly. Information includes process name, description, name of application database, maximum allowable duration, requester, previous as well as current tasks and actors, and completed tasks. Fig. 5-36 shows an example of such a process tracking document.

Apart from the process tracking document, a task notification message will be generated. Each responsible staff will receive one message through electronic mail so that they know the task that they are responsible for has been started and requires them to work for. They will receive this message in their mailboxes as a task notification. Such a notification includes appropriate 'pointer' (Doclink in Notes) so that users can get required information directly. Fig. 5-37 is an example of task notification.

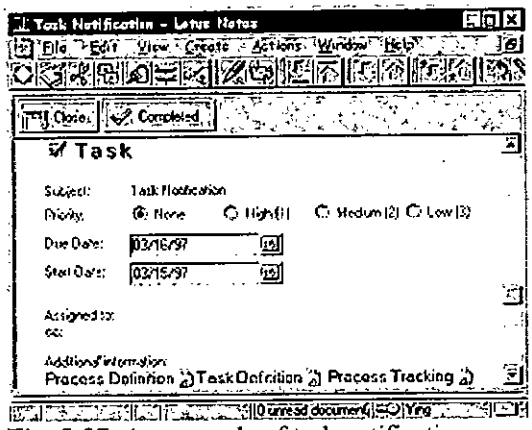


Fig. 5-37 An example of task notification.

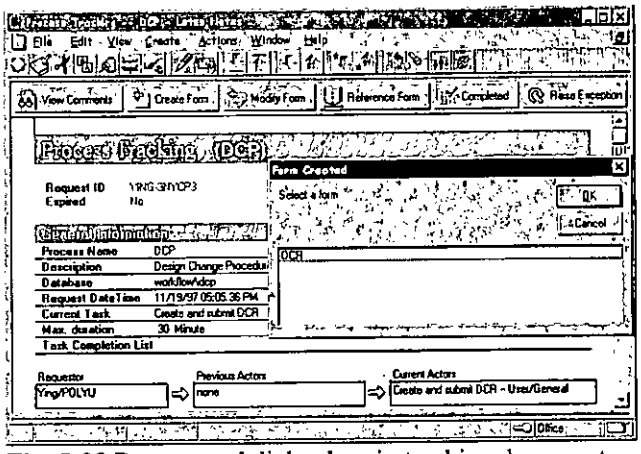


Fig. 5-38 Buttons and dialog box in tracking document.

Responsible actors can go directly to the process tracking document through Doclinks in the task notification message or from the process tracking view in the workflow specification database. Through buttons on the top of the slip, actors can raise exception when necessary, mark task to be completed, and view previous comments from other actors if there is any. Also, actors can create, modify and reference forms. By looking up the current task definition, the system can check if there is any form to create/ modify/ reference.

If there is any form to be created in the current task, a dialog box as Fig. 5-38 will be prompted to let users to choose which form to create. Otherwise, a message will be prompted to inform users that no form is required to create in the current task. Similarly, the 'Modify Form' and 'Reference Form' buttons behave the same way. Using the Request ID as a key, WAT will put actors directly to the required or selected form. All documents created under the same process instance will inherit the same Request ID. However, only current actors that have not finish their respective tasks will see these buttons and be authorized to perform all the above functions. Other actors can only reference the information.

Upon task completion, actor should mark the task as completed by clicking the "Completed" button in the process tracking document. Then, WAT will check if all the forms to be created, if there is any, before proceeding to the next task. Actor will be prompted if some required forms have not been created. Otherwise, WAT will prompt actor to enter his/her comment. Fig. 5-39 shows the dialog box to input comment. Then, there is a message box such as Fig. 5-40 indicating that the current actor has completed the task.

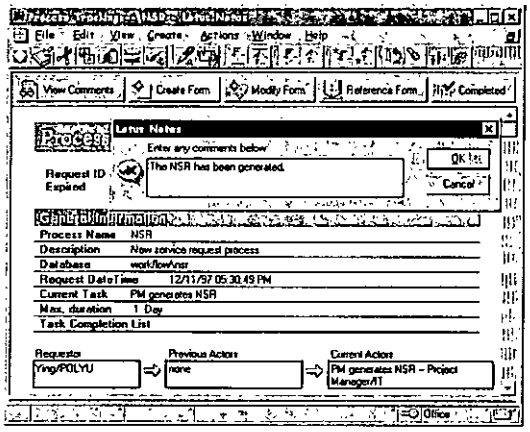


Fig. 5-39 Comment Dialog Box.

Fig. 5-40 Task Completed Message Box.

WAT will wait until all the responsible actors have completed their works and mark completed in the process tracking document. In order to start next task(s), all the forms required to create in the current task must be created. Also, WAT will check the availability of responsible actors, to see if any of them is currently on leave or has already not work for the company. If so, current actor will be prompted to decide whether to select other actor for substitution or just skip that actor. After checking all these, the next task(s) can be started.

If there are several parallel tasks running and the current actor involved in more than one task, the system will ask which task has been completed by the actor. When all the actors of the task have finished their tasks, the process will navigate to the next task according to the process specification.

The process tracking document will be updated by looking up the next task to follow. All actors responsible for the next task will receive a task notification with links to that process tracking document and respective process and task definitions. Similarly, process instance will propagate from task to task until completion. Through WAT, all tasks can be completed according to pre-defined sequence, by the right people and based on different conditions. Users only have to concentrate on human interaction - without involving in any repetitive and administrative tasks that is handled automatically by the system.

The WE component is able to keep track of the status of all processes that have been initiated in the organization by different views, such as a sorted or a categorized view by the current responsible actors, process name or request date (see Fig. 5-41 to Fig. 5-43). Information such as the number of a particular process that has been initiated, the total number of process that has been initiated on a particular date can also be obtained.

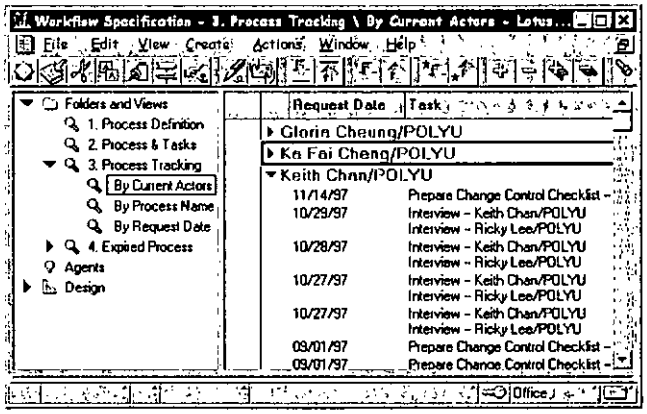


Fig. 5-41 The process tracking view by current actors.

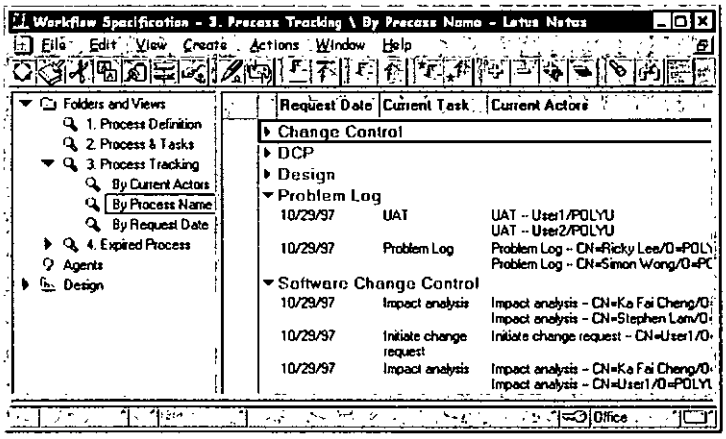


Fig. 5-42 The process tracking view by process name.

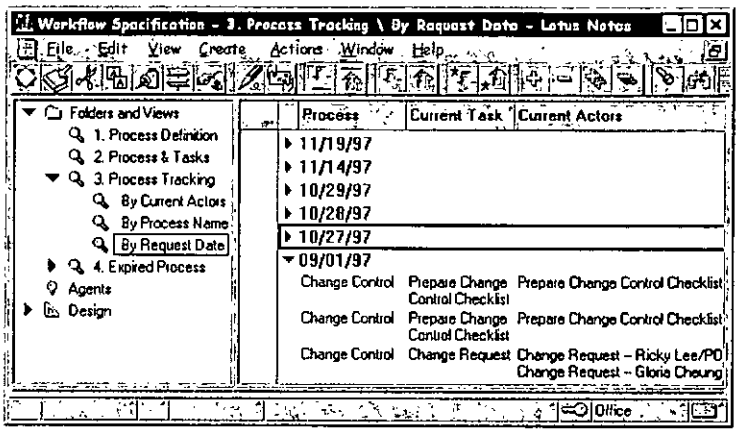


Fig. 5-43 The process tracking view by request date.

By comparing the maximum duration defined in the task definition and the elapsed duration since the task has started, a process instance is said to be 'expired' if the elapsed time exceeds the maximum duration. To monitor these expired process instances, an agent is designed to run on a scheduled basis (such as daily or weekly). After running the agent, those process instances satisfy the expiration condition will be marked as expired and moved to another view, expired process view. As more and more process instances start and execute, it is useful to help administrators to better manage and control all the processes, as well as prompt them to take action. Fig. 5-44 is an example of the expired process view.

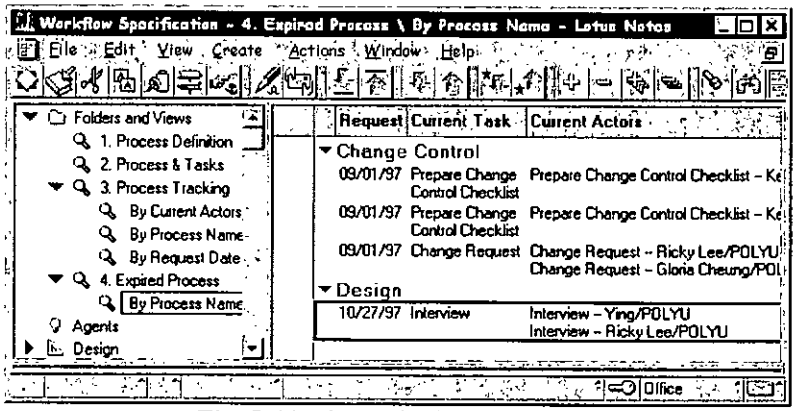


Fig. 5-44 The expired process view.

6. Test Cases

In this chapter, four test cases will be given to illustrate how WAT can be used to build a QMS and automate documented procedures in an organization. They are categorized into real and simulated cases. Real cases are processes that are practically implemented and used, while simulated cases are well-designed processes. For the real cases, a new service request process and a software development life cycle will be described. The new service request process is used in a bank to automate the initiation and approval of service requests. It starts from the initiation of a new service request and ends with sign-off of new service agreement. For the software development cycle, it is used by a software company in Hong Kong. It involves multiple processes from design, testing to user acceptance. Also, project management, change control and discussion will be included. It gives an integrated approach towards process quality and management.

For simulated cases, a design change procedure and a software change control process will be described. The design change procedure can be used to ensure that changes to the programming specifications are managed and controlled to maintain product quality and development productivity. And, the software change control process serves as another example to demonstrate the implementation of a process and how WAT works.

6.1. Real Cases

6.1.1. New Service Request Process

In this section, a new service request (NSR) process used in a bank will be discussed to demonstrate how a process can be implemented and automated by WAT. The following lists all the tasks in this NSR process. And, the process will be described and explained by a task-based workflow with detailed task descriptions.

- Project Manager (PM) generates new service request (NSR)
- Agreement of NSR from Business Sponsor (BS)
- PM revises NSR
- New service office (NSO) receives NSR
- PM provides additional information to NSO
- NSO forwards NSR to unit heads (UHs) for estimates
- UHs do the estimates
- UHs make changes to their estimates
- NSO generates costing agreement
- Financial controller (Fincon) signoff of agreement
- BS signoff agreement
- Fincon comments on agreement
- NSO marks completion of project

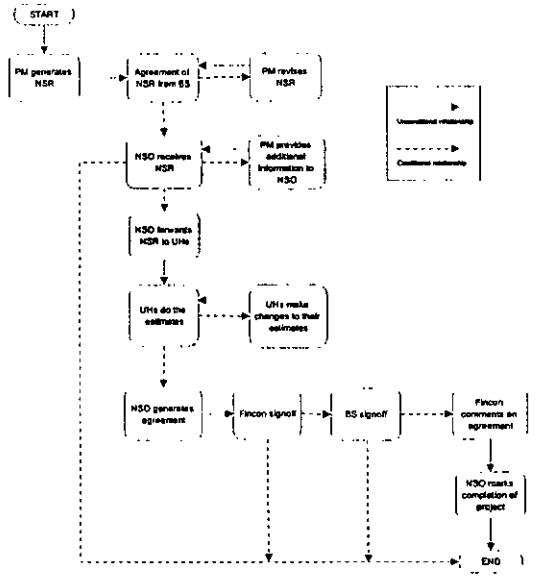


Fig. 6-1 The task-based workflow of the NSR process.

Fig. 6-1 shows the task-based workflow of the NSR process. Table 6-1 lists the task description of the process and Table 6-2 summarizes the actors and forms referenced in all the tasks.

Project Manager (PM) generates new service request (NSR)	NSR can be generated by any Project Manager (PM). Only he/she and the Business Sponsor (BS) for this project are allowed to read or edit the NSR. Upon the NSR is saved, all mandatory fields will be validated. Once the NSR is complete, it will be forwarded to Business Sponsor through mail for his/her agreement.
Agreement of NSR from Business Sponsor (BS)	After receiving the NSR from NSO, the BS will notify the NSO for his/her agreement to sponsor the project. At that time, only BS is able to edit the NSR. If the BS agrees to this NSR, a mail will be sent to the PM and NSO. Otherwise, a mail will be sent to the PM for further action, such as revisions.
PM revises NSR	PM should revise the NSR if BS rejects the NSR and requests he/she to revise it. After revising the NSR, the NSR will send back to the BS for his/her agreement.
New service office (NSO) receives NSR	Once agreed by the BS, the NSR will be reviewed by NSO. Only NSO can edit and perform actions.
PM provides additional information to NSO	The NSO can send the NSR back to the PM to request more information. After the PM has provided additional information, NSR will be forwarded to the NSO. Only PM is allowed to make revision on the NSR.
NSO forwards NSR to Unit Heads (UHs) for estimates	NSO will identify appropriate UHs to provide cost and resource estimates on the NSR. Selected UHs and members of the support teams will receive mail notification with escalation dates included.
UHs do the estimates	UHs will provide the effort and resource estimates and the estimates can only be confirmed by them. Once costing agreement is generated, UH is not allowed to update any estimate.
UHs make changes to their estimates	There should be a mechanism to make changes to resource estimates. Once the agreement has been generated, UH is not allowed to change his estimates.
NSO generates costing agreement	Once all estimates are received, costing agreement can be generated. It can only be edited by NSO and Financial controller (Fincon).
Financial controller (Fincon) signoff of agreement	Only Fincon can signoff or reject the agreement. Upon signoff, a mail will be sent to PM and BS to indicate the NSR has been financially approved. The mail will also indicate that the agreement has been forwarded to the BS for his/her signoff. If the agreement is rejected, the NSR will be canceled. No further action is required.
BS signoff agreement	Only BS can signoff or reject the agreement after Fincon has signed off. A mail will be sent to the Fincon, NSO, Direct SRPC, support team members and PM indicating that the signoff of agreement. If the agreement is rejected, the NSR will be canceled. No further action is required on this NSR.
Fincon comments on agreement	When agreement is signed off by BS, Fincon should add comments about when and how the MISing is done. The comment part can only be edited by the Fincon.
NSO marks completion of project	After the agreement has been signed off by the BS, the NSO can mark the completion of the project to freeze the NSR. No further action is required on this NSR.

Table 6-1 Task descriptions of NSR process.

Task Name	Actors	Form Referenced
PM generates NSR	PM	-
Agreement of NSR from BS	BS	NSR
PM revises NSR	PM	NSR
NSO receives NSR	NSO	-
PM provides additional information to NSO	PM	NSR
NSO forwards NSR to UHs for estimates	UH	-
UHs do the estimates	UH	NSR
UHs make changes to their estimates	UH	Estimate
NSO generates costing agreement	NSO	NSR, Estimate
Fincon signoff of agreement	Fincon	Agreement
BS signoff agreement	BS	Agreement
Fincon comments on agreement	Fincon	Agreement
NSO marks completion of project	NSO	-

Table 6-2 Actors and forms referenced in NSR process.

After importing the actors (such as PM, BS, NSO, UH, Fincon) and forms (such as NSR, estimate, agreement) from the databases of workflow enactment component, the workflow, actors and forms created, modified and referenced can be specified. Fig. 6-2 is the startup screen of the workflow editor after importing these actors and forms details. And, Fig. 6-3 is the task-based workflow of NSR process in the workflow editor.

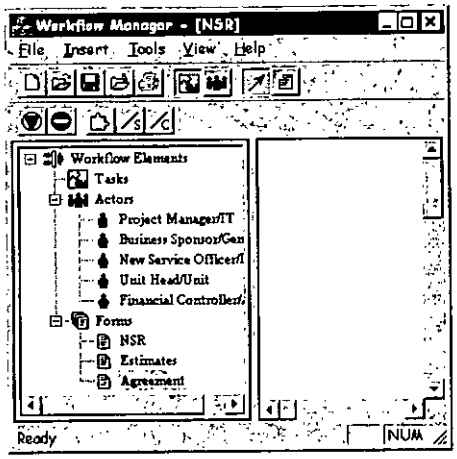


Fig. 6-2 Startup screen of NSR process.

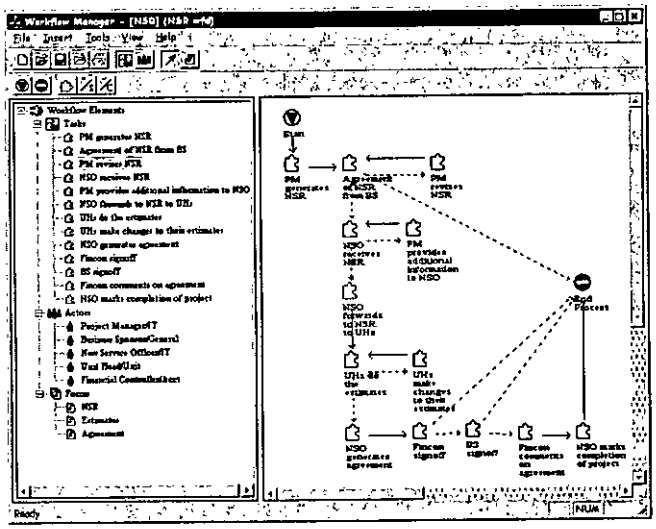


Fig. 6-3 Task-based workflow of NSR process.

Based on the above information, respective actor view can be built as in Fig. 6-4 and Table 6-3 illustrates all the actor relationships in Fig. 6-5 with appropriate attributes.

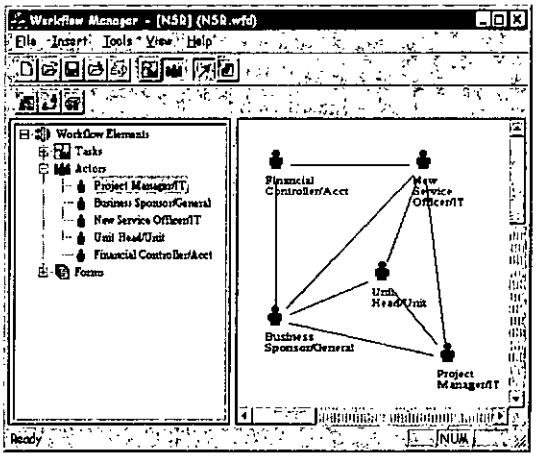


Fig. 6-4 Actor view of NSR process.

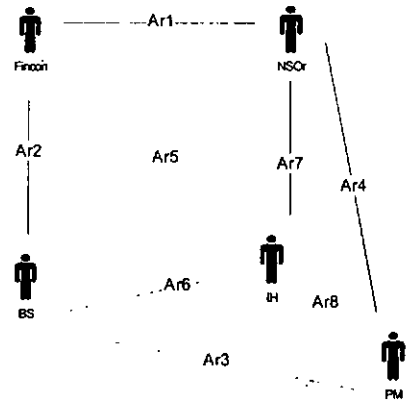


Fig. 6-5 Illustrative actor diagram of NSR process.

Label	Connector Attributes
Ar1	**Unconditional Task Dependency** Fincon (Fincon signoff) → NSO (NSO generates agreement) NSO (NSO marks completion of project) → Fincon (Fincon comments on agreement)
Ar2	**Information Sharing** Agreement - BS (BS signoff) - Fincon (Fincon comments on agreement , Fincon signoff) **Conditional Task Dependency** BS (BS signoff) → Fincon (Fincon signoff) Form Agreement field FinconSign operator = field value Y next task BS signoff Fincon (Fincon comments on agreement) → BS (BS signoff) Form Agreement field BSSign operator = field value Y next task Fincon comments on agreement
Ar3	**Information Sharing** NSR - PM (PM provides additional information to NSO, PM revises NSR) - BS (Agreement of NSR from BS) **Unconditional Task Dependency** BS (Agreement of NSR from BS) → PM (PM generates NSR, PM revises NSR) **Conditional Task Dependency** PM (PM revises NSR) → BS (Agreement of NSR from BS) Form NSR field BSDecision operator = field value Revise next task PM revises NSR

Table 6-3 Summary of connector in actor-based workflow of NSR process.

Label	Connector Attributes
Ar4	**Information Sharing** NSR - NSO(NSO generates agreement) - PM (PM provides additional information to NSO, PM revises NSR) **Unconditional Task Dependency** NSO (NSO receives NSR) → PM (PM provides additional information to NSO) **Conditional Task Dependency** PM (PM provides additional information to NSO) → NSO (NSO receives NSR) Form NSR field AddInfo operator = field value Y next task PM provides additional information to NSO
Ar5	**Information Sharing** NSR - NSO (NSO generates agreement) - BS (Agreement of NSR from BS) **Conditional Task Dependency** NSO (NSO receives NSR) → BS (Agreement of NSR from BS) Form NSR field BSDecision operator = field value OK next task NSO receives NSR
Ar6	**Information Sharing** NSR - UH (UHs do the estimates) - BS (Agreement of NSR from BS)
Ar7	**Information Sharing** NSR - UH (UHs do the estimates) - NSO (NSO generates agreement) Estimates - NSO (NSO generates agreement) - UH (UHs make changes to their estimates) **Unconditional Task Dependency** UH (UHs do the estimates) → NSO (NSO forwards NSR to UHs) **Conditional Task Dependency** NSO (NSO generates agreement) → UH (UHs do the estimates) Form Estimates field NeedChanges operator = field value N next task NSO generates agreement
Ar8	**Information Sharing** NSR - UH (UHs do the estimates) - PM (PM provides additional information to NSO, PM revises NSR)

Table 6-3 (Cont'd) Summary of connector in actor-based workflow of NSR process.

After capturing workflow in the workflow editor, workflow specification is imported into the database of the WE component. Only authorized actors have rights to import and modify workflow specification (process and task information). Table 6-4 is the workflow specification of the NSR process. And, Fig. 6-6 is a sample NSR in the application database.

```
WORKFLOW_MODEL NSR (name, description, database) [NSR, New service request process, workflow\nsr]

USES ACTOR_MODEL DEVELOPMENT;
USES INFORMATION_MODEL NSR;
START PM generates NSR;

TASK PM generates NSR
   GENERAL (description, type, precede, time, unit) [create NSR, , 1, 1, Day]
   ACTOR (actor list) [Project Manager/IT]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [NSR]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Agreement of NSR from BS]
END TASK;

TASK Agreement of NSR from BS
   GENERAL (description, type, precede, time, unit) [BS agrees/rejects NSR, , 2, 2, Day]
   ACTOR (actor list) [Business Sponsor/General]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [NSR]
   FORM_REF (form list) [NSR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 3, (By Form, NSR, BSDecision, =, OK, NSO receives NSR), (By
            Form, NSR, BSDecision, =, Revise, PM revises NSR), (By Form, NSR, BSDecision, =, Cancel, End Process)]
END TASK;

TASK PM revises NSR
   GENERAL (description, type, precede, time, unit) [revises NSR after rejection, , 1, 3, Day]
   ACTOR (actor list) [Project Manager/IT]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [NSR]
   FORM_REF (form list) [NSR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Agreement of NSR from BS]
END TASK;

TASK NSO receives NSR
   GENERAL (description, type, precede, time, unit) [Complete NSR after agreement, , 2, 2, Hour]
   ACTOR (actor list) [New Service Officer/IT]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [NSR]
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, NSR, AddInfo, =, Y, PM provides additional
            information to NSO), (By Form, NSR, AddInfo, =, N, NSO forwards NSR to UHs)]
END TASK;
```

Table 6-4 The workflow specification of the NSR process.

```
TASK PM provides additional information to NSO
   GENERAL (description, type, precede, time, unit) [Gives additional info, , 1, 1, Day]
   ACTOR (actor list) [Project Manager/IT]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [NSR]
   FORM_REF (form list) [NSR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, NSO receives NSR]
END TASK;

TASK NSO forwards NSR to UHs
   GENERAL (description, type, precede, time, unit) [forwards for estimates, , 1, 30, Minute]
   ACTOR (actor list) [New Service Officer/IT]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, UHs do the estimates]
END TASK;

TASK UHs do the estimates
   GENERAL (description, type, precede, time, unit) [Provides estimates, , 2, 3, Day]
   ACTOR (actor list) [Unit Head/Unit]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Estimates]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [NSR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Estimates,
            NeedChanges, =, Y, UHs make changes to their estimates), (By Form, Estimates, NeedChanges,
            =, N, NSO generates agreement)]
END TASK;

TASK UHs make changes to their estimates
   GENERAL (description, type, precede, time, unit) [Change estimates, , 1, 2, Day]
   ACTOR (actor list) [Unit Head/Unit]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Estimates]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, UHs do the estimates]
END TASK;

TASK NSO generates agreement
   GENERAL (description, type, precede, time, unit) [Create agreement, , 1, 1, Day]
   ACTOR (actor list) [New Service Officer/IT]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Agreement]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Estimates, NSR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Fincon signoff]
END TASK;
```

Table 6-4 (Cont'd) The workflow specification of the NSR process.

```
TASK Fincon signoff
   GENERAL (description, type, precede, time, unit) [agreement signed off by Fincon, , 1, 1, Day]
   ACTOR (actor list) [Financial Controller/Acct]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Agreement]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Agreement, FinconSign,
            =, Y, BS signoff), (By Form, Agreement, FinconSign, =, N, End Process)]
END TASK;

TASK BS signoff
   GENERAL (description, type, precede, time, unit) [Agreement signed off by BS, , 1, 1, Day]
   ACTOR (actor list) [Business Sponsor/General]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [Agreement]
   FORM_REF (form list) [Agreement]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Agreement, BSSign, =, Y,
            Fincon comments on agreement), (By Form, Agreement, BSSign, =, N, End Process)]
END TASK;

TASK Fincon comments on agreement
   GENERAL (description, type, precede, time, unit) [Gives comments on agreement, , 1, 2, Day]
   ACTOR (actor list) [Financial Controller/Acct]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [Agreement]
   FORM_REF (form list) [Agreement]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, NSO marks completion of project]
END TASK;

TASK NSO marks completion of project
   GENERAL (description, type, precede, time, unit) [Freeze NSR, , 1, 2, Day]
   ACTOR (actor list) [New Service Officer/IT]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [NSR]
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [End Process, ]
END TASK;
```

Table 6-4 (Cont'd) The workflow specification of the NSR process.

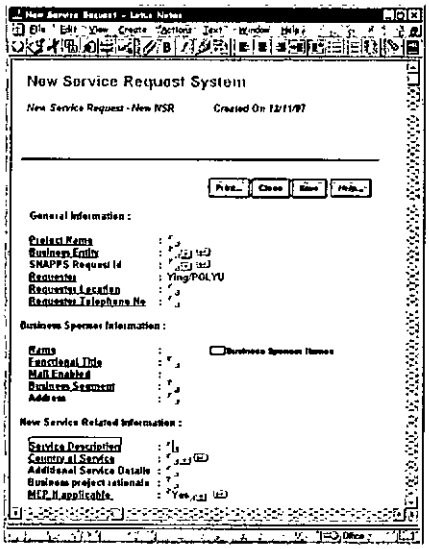


Fig. 6-6 NSR in the application database.

6.1.2. Software Development Cycle

To build a more comprehensive test case, we base on similar mechanism in the previous test case to model a software development cycle in a software company in Hong Kong. The development cycle consists of the following processes and the details of will be discussed in the following sub-sections:

Design This covers the design stage that is responsible for the production of functional and technical specifications. Meeting reports with users as well as other working papers will also be stored.

Testing This is the testing stage to detect program variations from specification. Unit test and system test will be covered and modifications to program codes are required.

Problem Logging This handles user acceptance, problem logging, problem fixing and testing after problem is fixed.

Change Control This involves four levels of change control approval - i) peer review by developers; ii) internal review by internal management; iii) external review by external management; and vi) user acceptance by users.

Project Management This is responsible for project management tasks. Management can create escalation and exception reports when necessary. Also, management bi-weekly report can be used to monitor progress of projects.

DESIGN

Fig. 6-7 shows the task-based workflow of the design process and Table 6-5 summarizes the actors and forms referenced in all the tasks. Based on this information, respective actor-based workflow is built as in Fig. 6-8 and Table 6-6 illustrates all the actor relationships with appropriate attributes. And, Table 6-7 is the workflow specification of the described design process.

Task Name	Actors	Form Referenced
Draft Functional Spec.	System Engineer	-
Interview	System Engineer, Project Manager	Functional Spec.
Refine Functional Spec.	System Engineer	Discussion Record, Working Paper
Evaluate Functional Spec.	Project Manager	Functional Spec., Discussion Record Working Paper
Detail Interview	Project Manager, System Engineer	Functional Spec.
Prepare Technical Spec.	System Engineer, Software Engineer	Functional Spec., Discussion Record Working Paper
Evaluate Technical Spec.	Project Manger	Functional Spec., Technical Spec.
Detail Analysis	Project Manager, System Engineer Software Engineer	Technical Spec.
Revise Technical Spec.	Software Engineer, System Engineer	Technical Spec., Discussion Record Working Paper

Table 6-5 Tasks in design process with respective actors and form referenced.

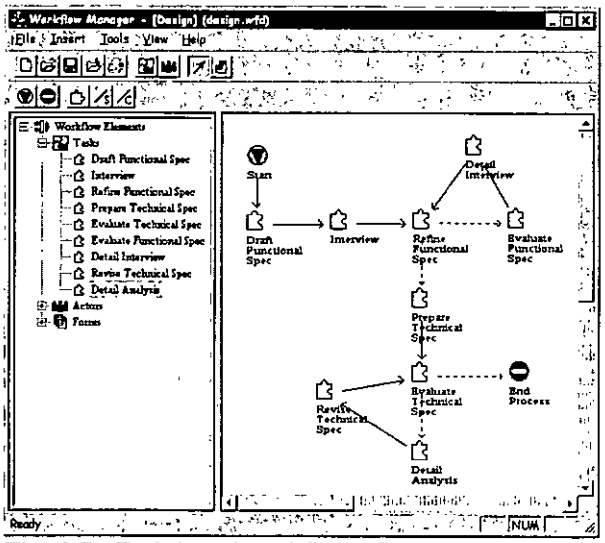


Fig. 6-7 Task-based workflow of design process.

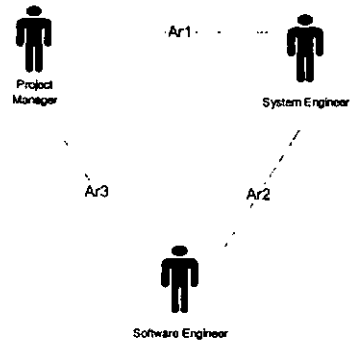


Fig. 6-8 Actor-based workflow of design process.

Label	Connector Attributes
Ar1	**Information Sharing** Functional Spec - System Engineer (Interview, Detail Interview, Prepare Technical Spec) - Project Manager (Interview, Evaluate Functional Spec, Detail Interview, Evaluate Technical Spec) Discussion Record - System Engineer (Refine Functional Spec, Prepare Technical Spec, Revise Technical Spec) - Project Manager (Evaluate Functional Spec) Working Paper - System Engineer (Refine Functional Spec, Prepare Technical Spec, Revise Technical Spec) - Project Manager (Evaluate Functional Spec) Technical Spec - System Engineer (Detail Analysis, Revise Technical Spec) - Project Manager (Evaluate Technical Spec, Detail Analysis) **Task Cooperation** Interview, Detail Interview, Detail Analysis **Unconditional Task Dependency** Project Manager (Interview) → System Engineer (Draft Functional Spec) System Engineer (Refine Functional Spec) → Project Manager (Interview) System Engineer (Detail Interview) → Project Manager (Evaluate Functional Spec) System Engineer (Refine Functional Spec) → Project Manager (Detail Interview) Project Manager (Evaluate Technical Spec) → System Engineer (Prepare Technical Spec) System Engineer (Revise Technical Spec) → Project Manager (Detail Analysis) Project Manager (Evaluate Technical Spec) → System Engineer (Revise Technical Spec) **Conditional Task Dependency** Project Manager (Evaluate Functional Spec) → System Engineer (Refine Functional Spec) Form: Functional Spec field: status operator: = field value: Need to revise next task: Evaluate Functional Spec
Ar2	**Information Sharing** Functional Spec - System Engineer (Interview, Detail Interview, Prepare Technical Spec) - Software Engineer (Prepare Technical Spec) Discussion Record - System Engineer (Refine Functional Spec, Prepare Technical Spec, Revise Technical Spec) - Software Engineer (Prepare Technical, Revise Technical Spec) Working Paper - System Engineer (Refine Functional Spec, Prepare Technical Spec, Revise Technical Spec) - Software Engineer (Prepare Technical, Revise Technical Spec) Technical Spec - System Engineer (Detail Analysis, Revise Technical Spec) - Software Engineer (Detail Analysis, Revise Technical Spec) **Task Cooperation** Prepare Technical Spec, Detail Analysis, Revise Technical Spec **Unconditional Task Dependency** Software Engineer (Revise Technical Spec) → System Engineer (Detail Analysis) System Engineer (Revise Technical Spec) → Software Engineer (Detail Analysis) **Conditional Task Dependency** Software Engineer (Prepare Technical Spec) → System Engineer (Refine Functional Spec) Form: Functional Spec field: status operator: = field value: Completed next task: Prepare Technical Spec

Table 6-6 Summary of connector in actor-based workflow of design process.

Label	Connector Attributes
Ar3	**Information Sharing** Functional Spec - Project Manager (Interview, Evaluate Functional Spec, Detail Interview, Evaluate Technical Spec) - Software Engineer (Prepare Technical Spec) Discussion Record - Project Manager (Evaluate Functional Spec) - Software Engineer (Prepare Technical, Revise Technical Spec) Working Paper - Project Manager (Evaluate Functional Spec) - Software Engineer (Prepare Technical, Revise Technical Spec) Technical Spec - Project Manager (Evaluate Technical Spec, Detail Analysis) - Software Engineer (Detail Analysis, Revise Technical Spec) **Task Cooperation** Detail Analysis **Unconditional Task Dependency** Project Manager (Evaluate Technical Spec) → Software Engineer (Prepare Technical Spec) Project Manager (Evaluate Technical Spec) → Software Engineer (Revise Technical Spec) Software Engineer (Revise Technical Spec) → Project Manager (Detail Analysis) **Conditional Task Dependency** Software Engineer (Detail Analysis) → Project Manager (Evaluate Technical Spec) Form: Technical Spec field: status operator: = field value: Need to revise next task: Detail Analysis

Table 6-6 (Cont'd) Summary of connector in actor-based workflow of design process.

```
WORKFLOW_MODEL Design (name, description, database)
[Design, Standard procedure for system design, Workflow\design]

USES ACTOR_MODEL Design_Actors;
USES INFORMATION_MODEL Design_Forms;
START Draft Functional Spec;

TASK Draft Functional Spec
   GENERAL (description, type, precede, time, unit) [Create first draft of functional specification, M, 1, 3, Day]
   ACTOR (actor list) [System Engineer/ Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Functional Specification]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Interview]
END TASK;

TASK Interview
   GENERAL (description, type, precede, time, unit) [Review spec. with users, M, 1, 1, Day]
   ACTOR (actor list) [System Engineer/ Development, Project Manager/ Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Discussion Record, Working Paper]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Functional Specification]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Refine Functional Spec]
END TASK;
```

Table 6-7 The workflow specification of the design process.

```
TASK Refine Functional Spec
   GENERAL (description, type, precede, time, unit) [Refine Functional Spec, M, 2, 2, Day]
   ACTOR (actor list) [System Engineer/ Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [Functional Specification]
   FORM_REF (form list) [Discussion Record, Working Paper]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Functional Specification,
            status, =, Completed, Prepare Technical Spec), (By Form, Functional Specification, status, =, Need
            to revise, Evaluate Functional Spec)]
END TASK;

TASK Prepare Technical Spec
   GENERAL (description, type, precede, time, unit) [Prepare Technical Spec, M, 1, 3, Day]
   ACTOR (actor list) [System Engineer/ Development, Software Engineer Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Technical Specification]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Functional Specification, Discussion Record, Working Paper]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Evaluate Technical Spec]
END TASK;

TASK Evaluate Technical Spec
   GENERAL (description, type, precede, time, unit) [Evaluate Technical Spec, M, 2, 2, Day]
   ACTOR (actor list) [Project Manager/ Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [Technical Specification]
   FORM_REF (form list) [Functional Specification, Technical Specification]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Technical Specification,
            status, =, Completed, End Process), (By Form, Functional Specification, status, =, Need to revise,
            Detail Analysis)]
END TASK;

TASK Evaluate Functional Spec
   GENERAL (description, type, precede, time, unit) [Evaluate Functional Spec, M, 1, 2, Day]
   ACTOR (actor list) [System Engineer/ Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [Functional Specification]
   FORM_REF (form list) [Discussion Record, Working Paper, Functional Specification]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Detail Interview]
END TASK;

TASK Detail Interview
   GENERAL (description, type, precede, time, unit) [Detail Interview, M, 1, 2, Hour]
   ACTOR (actor list) [Project Manager/ Development, System Engineer/ Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Discussion Record, Working Paper]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Functional Specification]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Refine Functional Spec]
END TASK;
```

Table 6-7 (Cont'd) The workflow specification of the design process.

```
TASK Revise Technical Spec
   GENERAL (description, type, precede, time, unit) [Revise Technical Spec, M, 1, 5, Day]
   ACTOR (actor list) [Software Engineer/ Development, System Engineer/ Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [Technical Specification]
   FORM_REF (form list) [Working Paper, Discussion Record, Technical Specification]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Evaluate Technical Spec]
END TASK;

TASK Detail Analysis
   GENERAL (description, type, precede, time, unit) [Detail Analysis, M, 1, 3, Day]
   ACTOR (actor list) [Project Manager/Development, System Engineer/ Development, Software Engineer/
                       Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Technical Specification, Discussion Record, Working Paper]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Revise Technical Spec]
END TASK;
```

Table 6-7 (Cont'd) The workflow specification of the design process.

TESTING

The task-based workflow of the testing process is shown in Fig. 6-9. Table 6-8 summarizes the actors and forms referenced in all the tasks. The respective actor-based workflow is built as in Fig. 6-10 and Table 6-9 illustrates all the actor relationships with appropriate attributes. The workflow specification of this testing process is shown as in Table 6-10.

Task Name	Actors	Form Referenced
Program variation	Software Engineer	-
Unit Test	Tester, Programmer	Program variation from Spec
Plan System Test	Project Manager	Program variation from Spec
Evaluate Unit Test Result	Software Engineer, Programmer	Unit Test result
System Test	Tester, Programmer	System Test Plan
Change Code after unit test	Programmer	Unit Test result
Evaluate system test result	Software Engineer	System Test result
Change code after system test	Programmer	System Test result
Integration Test	Tester, Programmer	-

Table 6-8 Tasks in testing process with respective actors and form referenced.

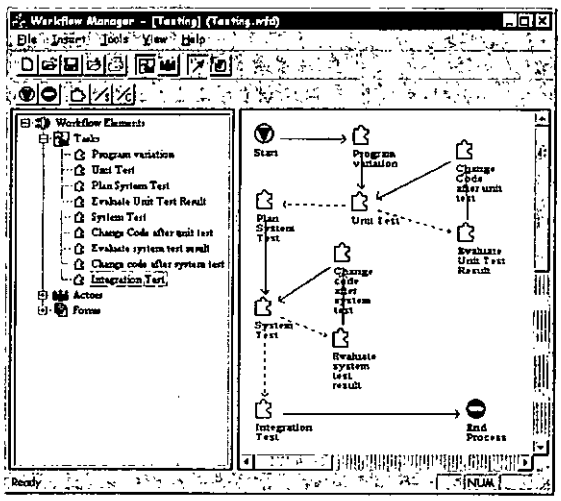


Fig. 6-9 Task-based workflow of testing process.

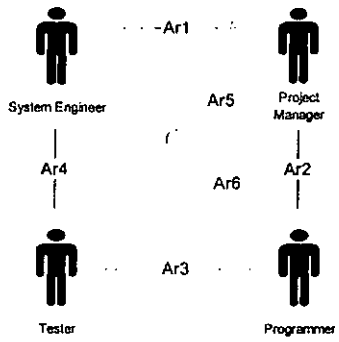


Fig. 6-10 Actor-based workflow of testing process.

Label	Connector Attributes
Ar1	**Information Sharing** System Test Result - Project Manager (Evaluate system test result) - Software Engineer (Evaluate system test result) **Task Cooperation** Evaluate System Test Result
Ar2	**Information Sharing** Program variation from Spec - Project Manager (Plan System Test) - Programmer (Unit Test) System Test Result - Project Manager (Evaluate system test result) - Programmer (Change code after system test) **Unconditional Task Dependency** Programmer (System test) → Project Manager (Plan system test) Programmer (Change code after system test) → Project Manager (Evaluate system test result) **Conditional Task Dependency** Project Manager (Plan System Test) → Programmer (Unit Test) Form Unit Test Result field RevStatus operator = field value Completed next task Plan System Test Project Manager (Evaluate System Test Result) → Programmer (System Test) Form System Test Result field RevStatus operator = field value Need to revise next task Evaluate System Test Result

Table 6-9 Summary of connector in actor-based workflow of testing process.

Label	Connector Attributes
Ar3	**Information Sharing** Program variation from Spec - Tester (Unit Test) - Programmer (Unit Test) System Test Plan - Tester (System Test) - Programmer (System Test) **Task Cooperation** Unit Test, System Test, Integration Test **Unconditional Task Dependency** Tester (Unit Test) → Programmer (Change code after unit test) Tester (System test) → Programmer (Change code after system test) **Conditional Task Dependency** Programmer (Evaluate Unit Test Result) → Tester (Unit Test) Form: Unit Test Result field: RevStatus operator: = field value: Need to revise next task: Evaluate Unit Test Result Programmer (Integration Test) → Tester (System Test) Form: System Test Result field: RevStatus operator: = field value: Completed next task: Integration Test Tester (Integration Test) → Programmer (System Test) Form: System Test Result field: RevStatus operator: = field value: Completed next task: Integration Test
Ar4	**Unconditional Task Dependency** Tester (Unit Test) → System Engineer (Program variation) **Conditional Task Dependency** Software Engineer (Evaluate Unit Test Result) → Tester (Unit Test) Form: Unit Test Result field: RevStatus operator: = field value: Need to revise next task: Evaluate Unit Test Result Software Engineer (Evaluate Unit Test Result) → Tester (System Test) Form: System Test Result field: RevStatus operator: = field value: Need to revise next task: Evaluate System Test Result

Table 6-9 (Cont'd) Summary of connector in actor-based workflow of testing process.

Label	Connector Attributes
Ar5	**Information Sharing** Program variation from Spec - Project Manager (Plan System Test) - Tester (Unit Test) **Unconditional Task Dependency** Tester (System test) → Project Manager(Plan system test) **Conditional Task Dependency** Project Manager (Plan System Test) → Tester (Unit Test) Form: Unit Test Result field: RevStatus operator: = field value: Completed next task: Plan System Test Project Manager (Evaluate System Test Result) → Tester (System Test) Form: System Test Result field: RevStatus operator: = field value: Need to revise next task: Evaluate System Test Result
Ar6	**Information Sharing** Unit Test Result - Software Engineer (Evaluate Unit Test Result) - Programmer (Evaluate Unit Test Result , Change code after unit test) System Test Result - Software Engineer (Evaluate system test result) - Programmer (Change code after system test) **Task Cooperation** Evaluate Unit Test Result **Unconditional Task Dependency** Programmer (Unit Test) → System Engineer (Program variation) Programmer (Change code after unit test) → System Engineer (Evaluate unit test result) Programmer (Change code after system test) → System Engineer (Evaluate system test result) **Conditional Task Dependency** Software Engineer (Evaluate Unit Test Result) → Programmer (Unit Test) Form: Unit Test Result field: RevStatus operator: = field value: Need to revise next task: Evaluate Unit Test Result Software Engineer (Evaluate System Test Result) → Programmer (System Test) Form: System Test Result field: RevStatus operator: = field value: Need to revise next task: Evaluate System Test Result

Table 6-9 (Cont'd) Summary of connector in actor-based workflow of testing process.

```
WORKFLOW_MODEL Testing (name, description, database) [Testing, , workflow\test]

USES ACTOR_MODEL Test_Actors;
USES INFORMATION_MODEL Test_Forms;
START Program variation;

TASK Program variation
   GENERAL (description, type, precede, time, unit) [Program variation, M, 1, 2, Day]
   ACTOR (actor list) [Software Engineer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Program Variation from Spec.]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Unit Test]
END TASK;

TASK Unit Test
   GENERAL (description, type, precede, time, unit) [Perform Unit Test, M, 2, 5, Day]
   ACTOR (actor list) [Tester/Development, Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Test Result]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Program Variation from Spec.]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Test Result, RevStatus,
            =, Need to revise, Evaluate Unit Test Result), (By Form, Test Result, RevStatus, =, Completed,
            Plan System Test)]
END TASK;

TASK Plan System Test
   GENERAL (description, type, precede, time, unit) [Create System Test, M, 1, 2, Day]
   ACTOR (actor list) [Project Manager/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [System Test Plan]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Program Variation from Spec.]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, System Test]
END TASK;

TASK Evaluate Unit Test Result
   GENERAL (description, type, precede, time, unit) [Evaluate Unit Test Result, M, 1, 1, Day]
   ACTOR (actor list) [Software Engineer/Development, Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Test Result]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Change Code after unit test]
END TASK;
```

Table 6-10 The workflow specification of the testing process.

```
TASK System Test
   GENERAL (description, type, precede, time, unit) [Perform System Test, M, 2, 3, Day]
   ACTOR (actor list) [Tester/Development, Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Test Result]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [System Test Plan]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Test Result, RevStatus,
            =, Need to revise, Evaluate system test result), (By Form, Test Result, RevStatus, =, Completed,
            Integration Test)]
END TASK;

TASK Change Code after unit test
   GENERAL (description, type, precede, time, unit) [Change Code after unit test, M, 1, 10, Day]
   ACTOR (actor list) [Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Test Result]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Unit Test]
END TASK;

TASK Evaluate system test result
   GENERAL (description, type, precede, time, unit) [Evaluate system test result, M, 1, 3, Day]
   ACTOR (actor list) [Software Engineer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Test Result]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Change code after system test]
END TASK;

TASK Change code after system test
   GENERAL (description, type, precede, time, unit) [Change code after system test, M, 1, 10, Day]
   ACTOR (actor list) [Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Test Result]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, System Test]
END TASK;

TASK Integration Test
   GENERAL (description, type, precede, time, unit) [Perform Integration Test, M, 1, 5, Day]
   ACTOR (actor list) [Tester/Development, Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Test Result]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [End Process, ]
END TASK;
```

Table 6-10 (Cont'd) The workflow specification of the testing process.

PROBLEM LOGGING

The task-based workflow of the problem logging process is shown in Fig. 6-11. Table 6-11 summarizes the actors and forms referenced in all the tasks. Fig. 6-12 shows the respective actor-based workflow and Table 6-12 lists all the actor relationships with appropriate attributes. And, Table 6-13 is the workflow specification of this process generated by the WC component.

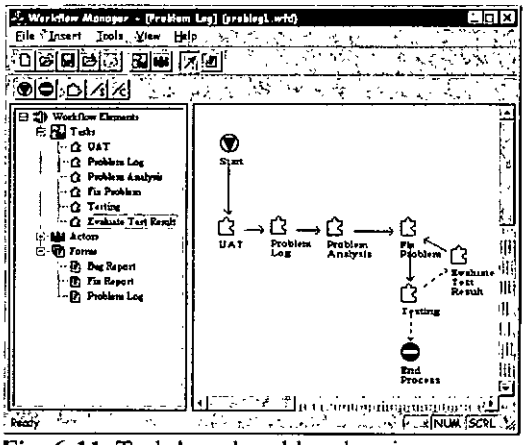


Fig. 6-11 Task-based problem logging process.

Task Name	Actors	Form Referenced
User Acceptance Test (UAT)	User	-
Problem Log	Programmer	-
Problem Analysis	Software Engineer, Programmer	Bug Report, Problem Log
Fix Problem	Programmer	Bug Report, Problem Log
Testing	Tester	Fix Report
Evaluate Test Result	Software Engineer, Programmer	Fix Report

Table 6-11 Tasks in problem logging process with respective actors and form referenced.

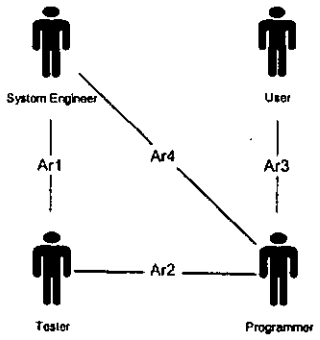


Fig. 6-12 Actor-based problem logging process.

Label	Connector Attributes
Ar1	**Information Sharing** Fix Report - Software Engineer (Evaluate Test Result) - Tester (Testing) **Conditional Task Dependency** Software Engineer (Evaluate Test Result) → Tester (Testing) Form Fix Report field RevStatus operator = field value Need to revise next task Evaluate Test Result
Ar2	**Information Sharing** Fix Report - Tester (Testing) - Programmer (Evaluate Test Result) **Unconditional Task Dependency** Tester (Testing) → Programmer (Fix Problem) **Conditional Task Dependency** Programmer (Evaluate Test Result) → Tester (Testing) Form Fix Report field RevStatus operator = field value Need to revise next task Evaluate Test Result
Ar3	**Unconditional Task Dependency** Programmer (Problem Log) → User (UAT)
Ar4	**Information Sharing** Bug Report - Programmer (Problem Analysis, Fix Problem) - Software Engineer (Problem Analysis) Problem Log - Programmer (Problem Analysis, Fix Problem) - Software Engineer (Problem Analysis) Fix Report - Software Engineer (Evaluate Test Result) - Programmer (Evaluate Test Result) **Task Cooperation** Problem Analysis, Evaluate Test Result **Unconditional Task Dependency** System Engineer (Problem Analysis) → Programmer (Problem Log) Programmer (Fix Problem) → System Engineer (Problem Analysis) Programmer (Fix Problem) → System Engineer (Evaluate Test Result)

Table 6-12 Summary of connector in actor-based workflow of problem logging process.

```
WORKFLOW_MODEL Problem Log (name, description, database) [Problem Log, , Workflow\ProbLog]
USES ACTOR_MODEL Prob_Actors;
USES INFORMATION_MODEL Prob_Forms;
START UAT;

TASK UAT
   GENERAL (description, type, precede, time, unit) [Perform user acceptance test, M, 1, 2, Day]
   ACTOR (actor list) [User/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Problem Log]
END TASK;
```

Table 6-13 The workflow specification of problem logging process.

```
TASK Problem Log
   GENERAL (description, type, precede, time, unit) [Create problem log or bug report, M, 1, 30, Minute]
   ACTOR (actor list) [Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Bug Report, Problem Log]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Problem Analysis]
END TASK;

TASK Problem Analysis
   GENERAL (description, type, precede, time, unit) [Problem Analysis, M, 1, 2, Day]
   ACTOR (actor list) [Software Engineer/Development,Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Bug Report, Problem Log]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Fix Problem]
END TASK;

TASK Fix Problem
   GENERAL (description, type, precede, time, unit) [Fix Problem, M, 2, 10, Day]
   ACTOR (actor list) [Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Fix Report]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Bug Report, Problem Log]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Testing]
END TASK;

TASK Testing
   GENERAL (description, type, precede, time, unit) [Testing, M, 1, 2, Day]
   ACTOR (actor list) [Tester/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Fix Report]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Fix Report, RevStatus, =,
            Completed, End Process), (By Form, Fix Report, RevStatus, =, Need to revise, Evaluate Test Result)]
END TASK;

TASK Evaluate Test Result
   GENERAL (description, type, precede, time, unit) [Evaluate Test Result, M, 1, 1, Day]
   ACTOR (actor list) [Software Engineer/Development, Programmer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Fix Report]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Fix Problem]
END TASK;
```

Table 6-13 (Cont'd) The workflow specification of problem logging process.

CHANGE CONTROL

The task-based workflow of the change control process is shown in Fig. 6-13. Table 6-14 summarizes the actors and forms referenced in all the tasks. Fig. 6-14 shows the respective actor-based workflow and Table 6-15 lists all the actor relationships with appropriate attributes. And, Table 6-16 is the workflow specification of this change control process.

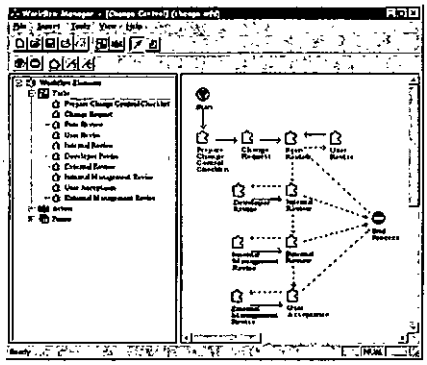


Fig. 6-13 Task-based change control process.

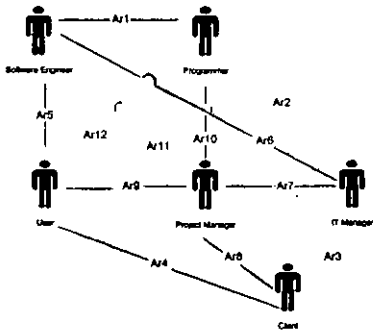


Fig. 6-14 Actor-based change control process.

Task Name	Actors	Form Referenced
Prepare Change Control Checklist	Project Manager	-
Change Request	User	Change control migration checklist
Peer Review	Programmer, Software Engineer	Change request form
Internal Review	Project Manager, IT Manager	Change request form
External Review	Client	Change request form
User Acceptance	User	Change request form
User Revise	User	Change request form
Developer Revise	Programmer, Software Engineer	Change request form
Internal Management Revise	Project Manager, IT Manager	Change request form
External Management Revise	Client	Change request form

Table 6-14 Tasks in change control process with respective actors and form referenced.

Label	Connector Attributes
Ar1	**Information Sharing** Change request form - Programmer (Peer Review, Developer Revise) - Software Engineer (Peer Review, Developer Revise) **Task Cooperation** Peer Review, Internal Review
Ar2	**Information Sharing** Change request form - Programmer (Peer Review, Developer Revise) - IT Manager (Internal Review, Internal Management Revise) **Unconditional Task Dependency** IT Manager (Internal Review) → Programmer (Developer Revise) **Conditional Task Dependency** IT Manager (Internal Review) → Programmer (Peer Review) **Form** Change request form **field** DevApproval **operator** = **field value** Approved **next task** Internal Review

Table 6-15 Summary of connector in actor-based workflow of change control process.

Label	Connector Attributes
Ar3	**Information Sharing** Change request form - IT Manager (Internal Review, Internal Management Revise) - Client (External Review, External Management Revise) **Unconditional Task Dependency** Client (External Review) → IT Manager (Internal Management Revise) **Conditional Task Dependency** Client (External Review) → IT Manager (Internal Review) **Form** Change request form **field** InternalManApproval **operator** = **field value** Approved **next task** External Review IT Manager (Internal Management Revise) → Client (External Review) **Form** Change request form **field** ExternalManApproval **operator** = **field value** Send back to Internal Management for review **next task** Internal Management Revise
Ar4	**Information Sharing** Change request form - User (User Acceptance, User Revise) - Client (External Review, External Management Revise) **Unconditional Task Dependency** User (User Acceptance) → Client (External Revise) **Conditional Task Dependency** User (User Acceptance) → Client (External Review) **Form** Change request form **field** ExternalManApproval **operator** = **field value** Approved **next task** User Acceptance Client (External Management Revise) → User (User Acceptance) **Form** Change request form **field** UserApproval **operator** = **field value** Send back to External Management for review **next task** External Management Revise
Ar5	**Information Sharing** Change request form - User (User Acceptance, User Revise) - Software Engineer (Peer Review, Developer Revise) **Unconditional Task Dependency** Software Engineer (Peer Review) → User (Change Request) Software Engineer (Peer Review) → User (User Revise) **Conditional Task Dependency** User (User Revise) → Software Engineer (Peer Review) **Form** Change request form **field** DevApproval **operator** = **field value** Send back to User for review **next task** User Revise

Table 6-15 (Cont'd) Summary of connector in actor-based workflow of change control process.

Label	Connector Attributes
Ar6	**Information Sharing** Change request form - IT Manager (Internal Review, Internal Management Revise) - Software Engineer (Peer Review, Developer Revise) **Unconditional Task Dependency** IT Manager (Internal Review) → Software Engineer (Developer Revise) **Conditional Task Dependency** IT Manager (Internal Review) → Software Engineer (Peer Review) **Form** Change request form **field** DevApproval **operator** = **field value** Approved **next task** Internal Review
Ar7	**Information Sharing** Change request form - IT Manager (Internal Review, Internal Management Revise) - Project Manager (Internal Review, Internal Management Revise) **Task Cooperation** Internal Review , Internal Management Revise
Ar8	**Information Sharing** Change request form - Project Manager (Internal Review, Internal Management Revise) - Client (External Review, External Management Revise) **Unconditional Task Dependency** Client (External Review) → Project Manager (Internal Management Revise) **Conditional Task Dependency** Client (External Review) → Project Manager (Internal Review) **Form** Change request form **field** InternalManApproval **operator** = **field value** Approved **next task** External Review Project Manager (Internal Management Revise) → Client (External Review) **Form** Change request form **field** ExternalManApproval **operator** = **field value** Send back to Internal Management for review **next task** Internal Management Revise
Ar9	**Information Sharing** Change request form - Project Manager (Internal Review, Internal Management Revise) - User (User Acceptance, User Revise) **Unconditional Task Dependency** User (Change Request) → Project Manager (Prepare Change Control Checklist)
Ar10	**Information Sharing** Change request form - Project Manager (Internal Review, Internal Management Revise) - Programmer (Peer Review, Developer Revise) **Unconditional Task Dependency** Project Manager (Internal Review) → Programmer (Developer Revise) **Conditional Task Dependency** Project Manager (Internal Review) → Programmer (Peer Review) **Form** Change request form **field** DevApproval **operator** = **field value** Approved **next task** Internal Review

Table 6-15 (Cont'd) Summary of connector in actor-based workflow of change control process.

Label	Connector Attributes
Ar11	**Information Sharing** Change request form - Project Manager (Internal Review, Internal Management Revise) - Software Engineer (Peer Review, Developer Revise) **Unconditional Task Dependency** Project Manager (Internal Review) → Software Engineer (Developer Revise) **Conditional Task Dependency** Project Manager (Internal Review) → Software Engineer (Peer Review) **Form** Change request form **field** DevApproval **operator** = **field value** Approved **next task** Internal Review
Ar12	**Information Sharing** Change request form - Programmer (Peer Review, Developer Revise) - User (User Acceptance, User Revise) **Unconditional Task Dependency** Programmer (Peer Review) → User (Change Request) Programmer (Peer Review) → User (User Revise) **Conditional Task Dependency** User (User Revise) → Programmer (Peer Review) **Form** Change request form **field** DevApproval **operator** = **field value** Send back to User for review **next task** User Revise

Table 6-15 (Cont'd) Summary of connector in actor-based workflow of change control process.

```
WORKFLOW_MODEL Change Control (name, description, database)
[Change Control, Standard process for software change, workflow\chgctrl]

USES ACTOR_MODEL Change_Actors;
USES INFORMATION_MODEL Change_Forms;
START Prepare Change Control Checklist;

TASK Prepare Change Control Checklist
   GENERAL (description, type, precede, time, unit) [New Checklist, M, 1, 1, Day]
   ACTOR (actor list) [Project Manager/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Change Control Migration Checklist]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Change Request]
END TASK;

TASK Change Request
   GENERAL (description, type, precede, time, unit) [Submit request, M, 1, 30, Minute]
   ACTOR (actor list) [User/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Control Migration Checklist]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Peer Review]
END TASK;
```

Table 6-16 The workflow specification of the change control process.

```
TASK Peer Review
   GENERAL (description, type, precede, time, unit) [Review request, M, 2, 3, Day]
   ACTOR (actor list) [Programmer/Development, Software Engineer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 3, (By Form, Change Request Form,
            RevStatus, =, Need to Revise, User Revise), (By Form, Change Request Form, RevStatus, =,
            Completed, Internal Review), (By Form, Change Request Form, RevStatus, =, Rejected, End Process)]
END TASK;

TASK User Revise
   GENERAL (description, type, precede, time, unit) [User review and revise, M, 1, 5, Day]
   ACTOR (actor list) [User/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Peer Review]
END TASK;

TASK Internal Review
   GENERAL (description, type, precede, time, unit) [Internal management approval, M, 2, 3, Day]
   ACTOR (actor list) [Project Manager/Development, IT Manager/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 3, (By Form, Change Request Form,
            RevStatus, =, Need to Revise, Developer Revise), (By Form, Change Request Form, RevStatus, =,
            Completed, External Review), (By Form, Change Request Form, RevStatus, =, Rejected, End Process)]
END TASK;

TASK Developer Revise
   GENERAL (description, type, precede, time, unit) [Developers approval, M, 1, 2, Day]
   ACTOR (actor list) [Programmer/Development, Software Engineer/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Internal Review]
END TASK;
```

Table 6-16 (Cont'd) The workflow specification of the change control process.

```
TASK External Review
   GENERAL (description, type, precede, time, unit) [External management approval, M, 2, 3, Day]
   ACTOR (actor list) [Client/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 3, (By Form, Change Request Form,
            RevStatus, =, Need to Revise, Internal Management Revise), (By Form, Change Request Form,
            RevStatus, =, Rejected, End Process), (By Form, Change Request Form, RevStatus, =, Completed,
            User Acceptance)]
END TASK;

TASK Internal Management Revise
   GENERAL (description, type, precede, time, unit) [Internal management revise request, M, 1, 3, Day]
   ACTOR (actor list) [Project Manager/Development, IT Manager/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, External Review]
END TASK;

TASK User Acceptance
   GENERAL (description, type, precede, time, unit) [Perform UAT, M, 2, 5, Day]
   ACTOR (actor list) [User/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Change Request Form,
            RevStatus, =, Need to Revise, External Management Revise), (By Form, Change Request Form,
            RevStatus, =, Rejected, End Process)]
END TASK;

TASK External Management Revise
   GENERAL (description, type, precede, time, unit) [External management revise request, M, 1, 3, Day]
   ACTOR (actor list) [Client/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, User Acceptance]
END TASK;
```

Table 6-16 (Cont'd) The workflow specification of the change control process.

PROJECT MANAGEMENT

Fig. 6-15 and Fig. 6-16 show the task-based and actor-based workflow of project management respectively. Whereas, Table 6-17 summarizes the actors and forms referenced in all the tasks. And, Table 6-18 lists all the actor relationships in the actor-based workflow with appropriate attributes. The workflow specification of this process is shown in Table 6-19.

Task Name	Actors	Form Referenced
Raise Exception	User	-
Review Exception	System Analyst	Exception Report
Task Escalation	Project Manager	Exception Report
Problem Solving	Programmer, System Analyst	Exception Report, Escalation Report
Testing	Tester	Exception Report, Escalation Report
Amendment	Programmer, System Analyst	Exception Report, Escalation Report

Table 6-17 Tasks in project management with respective actors and form referenced.

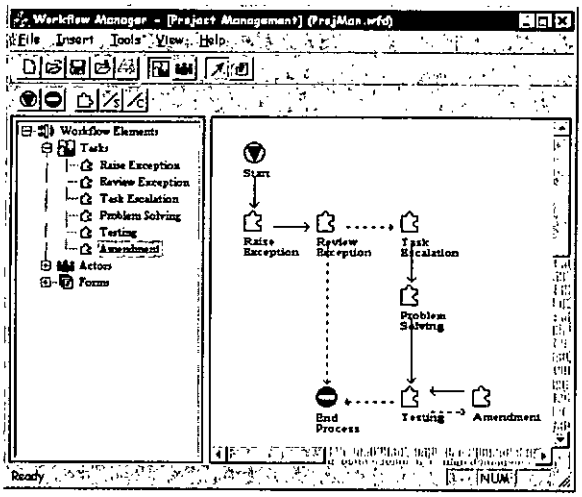


Fig. 6-15 Task-based of project management.

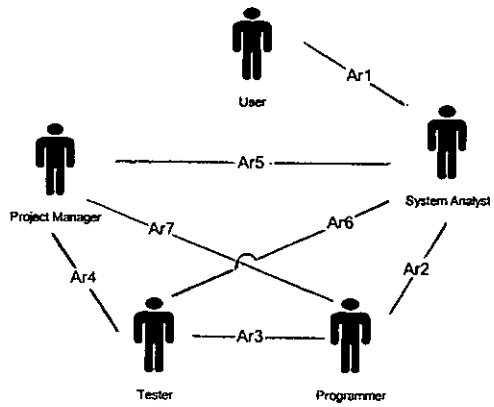


Fig. 6-16 Actor-based of project management.

Label	Connector Attributes
Ar1	**Unconditional Task Dependency** System Analyst (Review Exception) → User (Raise Exception)
Ar2	**Information Sharing** Exception Report - Programmer (Problem Solving, Amendment) - System Analyst (Problem Solving, Amendment) Escalation Report - Programmer (Problem Solving, Amendment) - System Analyst (Problem Solving, Amendment) **Task Cooperation** Problem Solving, Amendment

Table 6-18 Summary of connector in actor-based workflow of project management.

Label	Connector Attributes
Ar3	**Information Sharing** Exception Report - Programmer (Problem Solving, Amendment) - Tester (Testing) Escalation Report - Programmer (Problem Solving, Amendment) - Tester (Testing) **Unconditional Task Dependency** Tester (Testing) → Programmer (Problem Solving) Tester (Testing) → Programmer (Amendment) **Conditional Task Dependency** Programmer (Amendment) → Tester (Testing) **Form** Escalation Report **field** Completed **operator** = **field value** no **next task** Amendment
Ar4	**Information Sharing** Exception Report - Project Manager (Task Escalation) - Tester (Testing)
Ar5	**Information Sharing** Exception Report - Project Manager (Task Escalation) - System Analyst (Review Exception, Amendment) **Unconditional Task Dependency** System Analyst (Problem Solving) → Project Manager (Task Escalation) **Conditional Task Dependency** Project Manager (Task Escalation) → System Analyst (Review Exception) **Form** Exception Report **field** Continue **operator** = **field value** yes **next task** Task Escalation
Ar6	**Information Sharing** Exception Report - System Analyst (Review Exception, Problem Solving, Amendment) - Tester (Testing) Escalation Report - System Analyst (Review Exception, Problem Solving, Amendment) - Tester (Testing) **Unconditional Task Dependency** Tester (Testing) → System Analyst (Problem Solving, Amendment) **Conditional Task Dependency** System Analyst (Amendment) → Tester (Testing) **Form** Escalation Report **field** Completed **operator** = **field value** no **next task** Amendment
Ar7	**Information Sharing** Exception Report - Project Manager (Task Escalation) - Programmer (Review Exception, Amendment) **Unconditional Task Dependency** Programmer (Problem Solving) → Project Manager (Task Escalation) **Conditional Task Dependency** Project Manager (Task Escalation) → Programmer (Review Exception) **Form** Exception Report **field** Continue **operator** = **field value** yes **next task** Task Escalation

Table 6-18 (Cont'd) Summary of connector in actor-based workflow of project management.

```
WORKFLOW_MODEL Project Management (name, description, database)
[Project Management, , Workflow\PrjMgt]

USES ACTOR_MODEL ProjMan_Actors;
USES INFORMATION_MODEL ProjMan_Forms;

START Raise Exception;

TASK Raise Exception
   GENERAL (description, type, precede, time, unit) [User raise exception, M, 1, 3, Day]
   ACTOR (actor list) [User/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Exception Report]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Review Exception]
END TASK;

TASK Review Exception
   GENERAL (description, type, precede, time, unit) [Review exception report, M, 1, 1, Day]
   ACTOR (actor list) [System Analyst/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Exception Report]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Exception Report,
            Continue, =, no, End Process), (By Form, Exception Report, Continue, =, yes, Task Escalation)]
END TASK;

TASK Task Escalation
   GENERAL (description, type, precede, time, unit) [Assign task to responsible staff, M, 1, 3, Day]
   ACTOR (actor list) [Project Manager/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Escalation Report]
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Exception Report]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Problem Solving]
END TASK;

TASK Problem Solving
   GENERAL (description, type, precede, time, unit) [Solve problem based on exception report, M, 1, 10, Day]
   ACTOR (actor list) [Programmer/Development, System Analyst/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Escalation Report, Exception Report]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Testing]
END TASK;
```

Table 6-19 The workflow specification of project management process.

```
TASK Testing
   GENERAL (description, type, precede, time, unit) [Test programs after modification, M, 2, 5, Day]
   ACTOR (actor list) [Tester/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Escalation Report, Exception Report]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Escalation Report,
            Completed, =, yes, End Process), (By Form, Escalation Report, Completed, =, no, Amendment)]
END TASK;

TASK Amendment
   GENERAL (description, type, precede, time, unit) [Carry out Amendment, M, 1, 3, Day]
   ACTOR (actor list) [Programmer/Development, System Analyst/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Escalation Report, Exception Report]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Testing]
END TASK;
```

Table 6-19 (Cont'd) The workflow specification of project management process.

6.2. Simulated Cases

6.2.1. Design Change Procedure

In this section, a design change procedure (DCP) is used to demonstrate how WAT works. The details of the written procedure is first given and the activities (tasks) in the procedures will then be described and explained in the form of task-based workflow. Table 6-20 gives a detail written description. While Fig. 6-17 is the traditional workflow of the DCP and Table 6-21 is all the task descriptions.

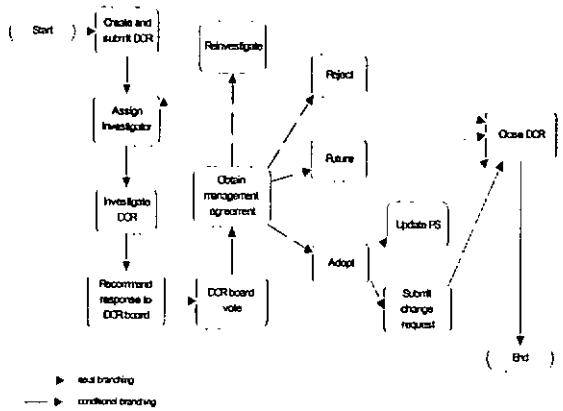


Fig. 6-17 Task-based workflow of DCP.

Purpose	The design change procedure (DCP) is to ensure that changes to the programming specifications (PS) are managed and controlled to maintain product quality and development productivity. This procedure covers the following : • Submission of a request for change to the PS. The request contains all the information to enable an investigator to make recommendation about the proposed change. • Actions required to make a decision on the proposed change. • Ensuring that all items impacted by the proposed change are identified and changed. The DCP is activated for a project as soon as the PS is ready for approval, at which time change requests can be submitted.
Owner	The DCP is owned by the manager of the product design department. He/she is the final arbiter relative to all changes to the design for which there is not unanimity.
Input	A design change request (DCR) is submitted to propose a change to the PS during the development process after the PS is ready for approval. DCRs can only be submitted to request changes to the design; they will not be accepted for proposed additional function. DCRs are submitted to the product design department and may be submitted by anyone.
Output	The output from the DCP is a closed DCR. DCRs are closed with one of three dispositions. 1. Reject. Proposed change is rejected and is not recommended for future consideration. 2. Future. Proposed change is rejected for this release but should be considered for inclusion in some future release of the product. 3. Adopt. Proposed change is accepted for inclusion in this release of the product. Updates to the PS plus change requests for other impacted items must be made before the DCR is closed.

Table 6-20 Detail written description of the DCP.

Create and submit a DCR	Fill out a DCR form requesting a change to the PS. Submit the DCR form to the project design department. Anyone can create and submit a DCR.
Assign an investigator	The change request coordinator reviews the DCR for completeness, assigns a number for tracking purposes, and assigns the DCR to an investigator for investigation.
Investigate DCR	The investigator reviews the DCR and proposed solution and recommends adopt, reject, or future. The investigator provides a written statement substantiating his or her recommendation. The DCR, along with the investigator recommendation and rationale, is forwarded to members of the DCR review board for its review and vote.
DCR Review Board Vote	DCR review board members are responsible for reviewing each DCR and recommendation and returning their vote within five days of assignment of the DCR to the investigator. Votes must be unanimous. When they are not unanimous, a meeting may be required to resolve disagreements. The manager of the product design department is the final arbiter on all DCRs.
Obtain Management Agreement	The adopted DCR must receive management agreement from all affected and impacted item owners prior to being officially made part of the plan. Management agreement not only indicates conceptual agreement with the change, but indicates commitment to implement the changes. Representatives from all impacted areas review the adopted DCR, assess the impact to their area, determine if they can accommodate the required changes, and make their recommendation to their manager. The manager approves or disapproves the change. DCRs with agreement from all affected parties are incorporated into the plan.
Update PS	When a DCR receives management agreement, the PS is updated by its owner. If sections of the PS have different owners, each owner is responsible for making the changes to his or her respective section.
Submit Change Request	The project design department must submit change requests to each area impacted by the agreed design change.
Close DCR	When the PS updates are complete and change requests to impacted areas are submitted, the DCR coordinator closes the adopted DCR. For rejected or future DCRs, the DCR coordinator closes the DCR as soon as the DCR review board vote is received.

Table 6-21 Task descriptions of DCP.

After importing the actors (such as users, manager, DCR coordinator, etc.) and forms (such as DCR and PS) from the databases of workflow enactment component, the workflow, actors and forms created, modified and referenced can be specified. Fig. 6-18 is the task-based workflow of DCP in the workflow editor.

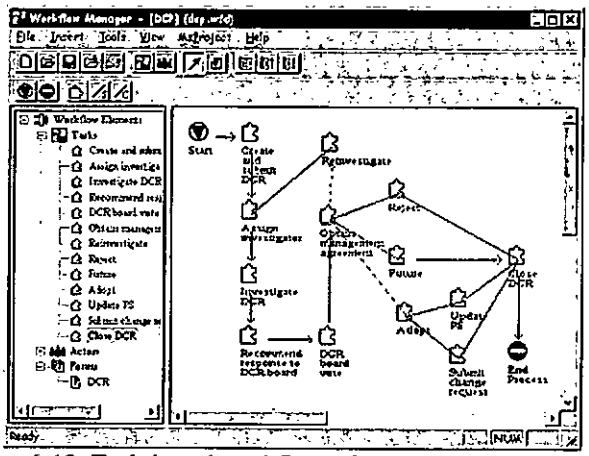


Fig. 6-18 Task-based workflow of DCP in workflow editor.

With all the above information, actors and forms referenced in DCP are summarized in Table 6-22 as follows. Table 6-23 illustrates all the actor relationships in Fig. 6-19 with appropriate attributes.

Task Name	Actors	Form Referenced
Create and submit a DCR	User	-
Assign an investigator	Change request coordinator	DCR
Investigate DCR	Investigator	DCR
Recommend response to DCR board	Change request coordinator Manager	DCR
DCR board Vote	DCR review board member	DCR
Obtain Management Agreement	DCR review board member	DCR
Reinvestigate	Investigator	DCR
Reject	DCR review board member	DCR
Future	Change request coordinator Manager	DCR
Adopt	Change request coordinator Manager	DCR
Update PS	System analyst	DCR, PS
Submit Change Request	Change request coordinator	-
Close DCR	Change request coordinator	-

Table 6-22 Actors and forms referenced in DCP.

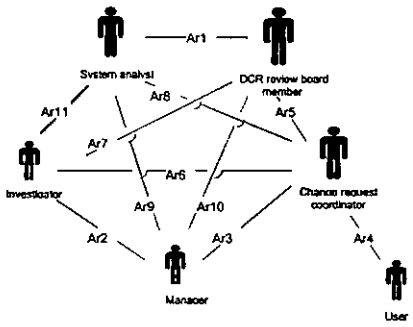


Fig. 6-19 Illustrative actor diagram of DCP.

Label	Connector Attributes
Ar1	**Information Sharing** DCR - DCR review board (Adopt, DCR board vote, Future, Obtain management agreement, Reject) - System Analyst (Update PS) **Unconditional Task Dependency** System analyst (Update PS) → DCR review board member (Adopt)
Ar2	**Information Sharing** DCR - Manager (Adopt, Future, Recommend response to DCR board) - Investigator (Investigate DCR, Reinvestigate) **Unconditional Task Dependency** Manager (Recommend response to DCR board) → Investigator (Investigate DCR)
Ar3	**Information Sharing** DCR - Manager (Adopt, Future, Recommend response to DCR board) - Change request coordinator (Assign investigator, Recommend response to DCR board) **Task Cooperation** Recommend response to DCR board **Unconditional Task Dependency** Change request coordinator (Submit change request) → Manager (Adopt) Change request coordinator (Close DCR) → Manager (Future)
Ar4	**Unconditional Task Dependency** Change request coordinator (Assign investigator) → User (Create and submit DCR)
Ar5	**Information Sharing** DCR - DCR review board (Adopt, DCR board vote, Future, Obtain management agreement, Reject) - Change request coordinator (Assign investigator, Recommend response to DCR board) **Unconditional Task Dependency** DCR review board member (DCR board vote) → Change request coordinator (Recommend response to DCR board) Change request coordinator (Submit change request) → DCR review board member (Adopt) Change request coordinator (Close DCR) → DCR review board member (Reject, Future)
Ar6	**Information Sharing** DCR - Investigator (Investigate DCR, Reinvestigate) - Change request coordinator (Assign investigator, Recommend response to DCR board) **Unconditional Task Dependency** Investigator (Investigate DCR) → Change request coordinator (Assign investigator) Change request coordinator (Assign investigator) → Investigator (Reinvestigate) Change request coordinator (Recommend response to DCR board) → Investigator (Investigate DCR)

Table 6-23 Summary of connector in actor view of DCP.

Label	Connector Attributes
Ar7	**Information Sharing** DCR - Investigator (Investigate DCR, Reinvestigate) - DCR review board (Adopt, DCR board vote, Future, Obtain management agreement, Reject) **Conditional Task Dependency** Investigator (Reinvestigate) → DCR review board (Obtain management agreement) **Form** DCR **field** continue **operator** = **field value** reinvestigate **next task** Reinvestigate
Ar8	**Information Sharing** DCR - Change request coordinator (Assign investigator, Recommend response to DCR board) - System Analyst (Update PS) **Unconditional Task Dependency** Change request coordinator (Close DCR) → System analyst (Update PS)
Ar9	**Information Sharing** DCR - Manager (Adopt, Future, Recommend response to DCR board) - System Analyst (Update PS) **Unconditional Task Dependency** System analyst (Update PS) → Manager (Adopt)
Ar10	**Information Sharing** DCR - Manager (Adopt, Future, Recommend response to DCR board) - DCR review board (Adopt, DCR board vote, Future, Obtain management agreement, Reject) **Task Cooperation** Adopt, Future **Unconditional Task Dependency** DCR review board member (DCR board vote) → Manager (Recommend response to DCR board) **Conditional Task Dependency** Manager (Future) → DCR review board (Obtain management agreement) **Form** DCR **field** continue **operator** = **field value** future **next task** Future Manager (Adopt) → DCR review board (Obtain management agreement) **Form** DCR **field** continue **operator** = **field value** adopt **next task** Adopt
Ar11	**Information Sharing** DCR - Investigator (Investigate DCR, Reinvestigate)

Table 6-23 (Cont'd) Summary of connector in actor view of DCP.

After workflow is captured, it is translated into workflow specification and imported into workflow specification database. Table 6-24 is the DCP workflow specification. And, Fig. 6-20 is an example of DCR in the application database.

```
WORKFLOW_MODEL DCP (name, description, database) [DCP, Design Change Procedure, workflow\dcp]

USES ACTOR_MODEL DEVELOPMENT;
USES INFORMATION_MODEL DCP;
START Create and submit DCR;

TASK Create and submit DCR
   GENERAL (description, type, precede, time, unit) [new DCR, Manual, 1, 30, Minute]
   ACTOR (actor list) [User/General]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [DCR]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Assign investigator]
END TASK;

TASK Assign investigator
   GENERAL (description, type, precede, time, unit) [find suitable investigator, Manual, 2, 2, Day]
   ACTOR (actor list) [Change request coordinator/Design]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [DCR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Investigate DCR]
END TASK;

TASK Investigate DCR
   GENERAL (description, type, precede, time, unit) [study DCR, Manual, 1, 5, Day]
   ACTOR (actor list) [Investigator/Design]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [DCR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Recommend response to DCR board]
END TASK;

TASK Recommend response to DCR board
   GENERAL (description, type, precede, time, unit) [Response to DCR board, Manual, 1, 2, Day]
   ACTOR (actor list) [Manager/Design, Change request coordinator/Design]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [DCR]
   FORM_REF (form list) [DCR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, DCR board vote]
END TASK;

TASK DCR board vote
   GENERAL (description, type, precede, time, unit) [DCR meeting, Manual, 1, 2, Hour]
   ACTOR (actor list) [DCR review board member/Board]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [DCR]
   FORM_REF (form list) [DCR]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Obtain management agreement]
END TASK;
```

Table 6-24 Workflow specification of the DCP.

```
TASK Obtain management agreement
  GENERAL (description, type, precede, time, unit) [agree on decision on DCR, Manual, 1, 1, Day]
  ACTOR (actor list) [DCR review board member/Board]
  NOTIFY (actor list) []
  FORM_CREATE (form list) []
  FORM_MODIFY (form list) [DCR]
  FORM_REF (form list) [DCR]
  EXCEPTION (method, argument) [Ignore, ]
  COMPLETE (method, argument) [Start Single Task Based on Condition, 4, (By Form, DCR, continue, =,
      reinvestigate, Reinvestigate), (By Form, DCR, continue, =, reject, Reject), (By Form, DCR, continue, =,
      future, Future), (By Form, DCR, continue, =, adopt, Adopt)]
END TASK;

TASK Reinvestigate
  GENERAL (description, type, precede, time, unit) [investigate DCR again, Manual, 1, 5, Day]
  ACTOR (actor list) [Investigator/Design]
  NOTIFY (actor list) []
  FORM_CREATE (form list) []
  FORM_MODIFY (form list) []
  FORM_REF (form list) [DCR]
  EXCEPTION (method, argument) [Ignore, ]
  COMPLETE (method, argument) [Start Single Task, Assign investigator]
END TASK;

TASK Reject
  GENERAL (description, type, precede, time, unit) [Reject DCR, Manual, 1, 1, Hour]
  ACTOR (actor list) [DCR review board member/Board]
  NOTIFY (actor list) []
  FORM_CREATE (form list) []
  FORM_MODIFY (form list) [DCR]
  FORM_REF (form list) [DCR]
  EXCEPTION (method, argument) [Ignore, ]
  COMPLETE (method, argument) [Start Single Task, Close DCR]
END TASK;

TASK Future
  GENERAL (description, type, precede, time, unit) [Design change for future use, Manual, 1, 1, Hour]
  ACTOR (actor list) [DCR review board member/Board, Manager/Design]
  NOTIFY (actor list) []
  FORM_CREATE (form list) []
  FORM_MODIFY (form list) []
  FORM_REF (form list) [DCR]
  EXCEPTION (method, argument) [Ignore, ]
  COMPLETE (method, argument) [Start Single Task, Close DCR]
END TASK;

TASK Adopt
  GENERAL (description, type, precede, time, unit) [adopt DCR, Manual, 1, 1, Hour]
  ACTOR (actor list) [Manager/Design, DCR review board member/Board]
  NOTIFY (actor list) []
  FORM_CREATE (form list) []
  FORM_MODIFY (form list) []
  FORM_REF (form list) [DCR]
  EXCEPTION (method, argument) [Ignore, ]
  COMPLETE (method, argument) [Start Multiple Task, Update PS, Submit change request]
END TASK;
```

Table 6-24 (Cont'd) Workflow specification of the DCP.

```
TASK Update PS
   GENERAL (description, type, precede, time, unit) [update PS according to DCR, Manual, 1, 5, Day]
   ACTOR (actor list) [System analyst/Design]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [PS]
   FORM_REF (form list) [DCR, PS]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Close DCR]
END TASK;

TASK Submit change request
   GENERAL (description, type, precede, time, unit) [Submit DCR, Manual, 1, 30, Minute]
   ACTOR (actor list) [Change request coordinator/Design]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Close DCR]
END TASK;

TASK Close DCR
   GENERAL (description, type, precede, time, unit) [close, Manual, 4, 30, Minute]
   ACTOR (actor list) [Change request coordinator/Design]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) [DCR]
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [End Process, ]
END TASK;
```

Table 6-24 (Cont'd) Workflow specification of the DCP.

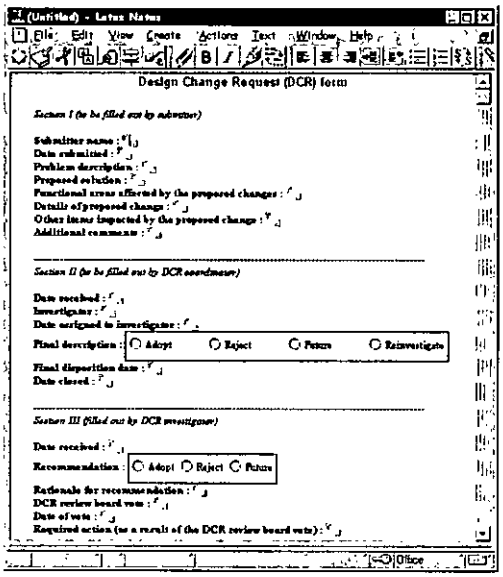


Fig. 6-20 DCR in the application database.

6.2.2. Software Change Control

We use a software change control process as another simulated test case. Table 6-25 summarizes the actors and forms referenced in all the tasks. Fig. 6-21 is the task-based workflow of this process in the workflow editor.

Task Name	Actors	Form Referenced
Initiate change request	User	-
Approve change request	Project Manager	Change request form
Archive change request	Support	-
Impact analysis	Support	Change request form
Schedule change request	Project Manager	Project schedule, Staff schedules
Schedule UAT	Project Manager	Staff schedules
Do the changes	Support and Programmer	Change request form, Source code
Test the changes	Tester and Programmer	Source code, user requirements
Perform UAT	User	Change request form

Table 6-25 Tasks in software change control process with respective actors and form referenced.

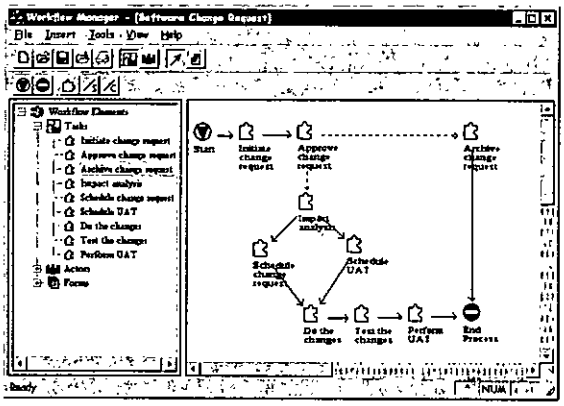


Fig. 6-21 Task-based software change control.

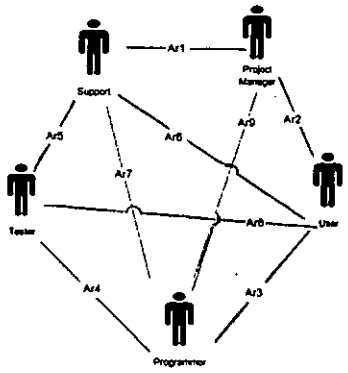


Fig. 6-22 Actor-based software change control.

Respective actor-based workflow is shown as in Fig. 6-22. Table 6-26 summarizes all the actor relationships in the actor-based workflow and Table 6-27 is the workflow specification.

Label	Connector Attributes
Ar1	**Information Sharing** Change request form - Support (do the changes, impact analysis) - Project Manager (approve change request) **Unconditional Task Dependency** Project Manager (Schedule change request, schedule UAT) → Support (impact analysis) Support (Do the change) → Project Manager (Schedule change request, schedule UAT) **Conditional Task Dependency** Support (Impact Analysis) → Project Manager (Approve change request) **Form** change request form **field** status **operator** = **field value** approved **next task** impact analysis Support (Archive change request) → Project Manager (Approve change request) **Form** change request form **field** status **operator** = **field value** disapprove **next task** archive change request
Ar2	**Information Sharing** Change request form - User (perform UAT) - Project Manager (approve change request) **Unconditional Task Dependency** Project Manager (approve change request) → User (initiate change request)
Ar3	**Information Sharing** Change request form - User (perform UAT) - Programmer (do the changes) **Unconditional Task Dependency** User (Perform UAT) → Programmer (Test the changes)
Ar4	**Information Sharing** Source code - Tester (test the changes) - Programmer (do the changes, test the changes) User requirements - Tester (test the changes) - Programmer (test the changes) **Task Cooperation** Test the changes **Unconditional Task Dependency** Tester (Test the change) → Programmer (Do the changes)
Ar5	**Information Sharing** Source code - Tester (test the changes) - Support (do the changes) **Unconditional Task Dependency** Tester (Test the change) → Support (Do the changes)
Ar6	**Information Sharing** Change request form - User (perform UAT) - Support (do the changes, impact analysis)
Ar7	**Information Sharing** Change request form - Programmer (do the changes) - Support (do the changes, impact analysis) Source code - Programmer (do the changes, test the changes) - Support (do the changes) **Task Cooperation** Do the changes **Unconditional Task Dependency** Tester (Test the change) → Programmer (Do the changes)

Table 6-26 Summary of connector in actor-based workflow of software change control process.

```
WORKFLOW_MODEL Software Change Request (name, description, database)
[Software Change Request, , wf\request]

USES ACTOR_MODEL Change_Actors;
USES INFORMATION_MODEL Change_Forms;
START Initiate change request;

TASK Initiate change request
   GENERAL (description, type, precede, time, unit) [Create change request form, M, 1, 30, Minute]
   ACTOR (actor list) [User/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) [Change Request Form]
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Approve change request]
END TASK;

TASK Approve change request
   GENERAL (description, type, precede, time, unit) [Change request approval, M, 1, 2, Day]
   ACTOR (actor list) [Project Manager/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task Based on Condition, 2, (By Form, Change Request Form,
            status, =, disapproved, Archive change request), (By Form, Change Request Form, status, =,
            approved, Impact analysis)]
END TASK;

TASK Archive change request
   GENERAL (description, type, precede, time, unit) [Archive disapproved request, M, 1, 2, Day]
   ACTOR (actor list) [Support/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) []
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [End Process, ]
END TASK;

TASK Impact analysis
   GENERAL (description, type, precede, time, unit) [Analyse impact after changes, M, 1, 4, Day]
   ACTOR (actor list) [Support/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Multiple Task, Schedule change request, Schedule UAT]
END TASK;
```

Table 6-27 The workflow specification of software change control process.

```
TASK Schedule change request
   GENERAL (description, type, precede, time, unit) [Schedule changes, M, 1, 1, Day]
   ACTOR (actor list) [Project Manager/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Staff schedule, Project schedule]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Do the changes]
END TASK;

TASK Schedule UAT
   GENERAL (description, type, precede, time, unit) [Schedule UAT, M, 1, 1, Day]
   ACTOR (actor list) [Project Manager/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Staff schedule]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Do the changes]
END TASK;

TASK Do the changes
   GENERAL (description, type, precede, time, unit) [Perform changes, M, 2, 10, Day]
   ACTOR (actor list) [Programmer/Development, Support/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form, Source code]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Test the changes]
END TASK;

TASK Test the changes
   GENERAL (description, type, precede, time, unit) [Test after changes, M, 1, 8, Day]
   ACTOR (actor list) [Programmer/Development, Tester/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Source code, User requirements]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [Start Single Task, Perform UAT]
END TASK;

TASK Perform UAT
   GENERAL (description, type, precede, time, unit) [Carry out user acceptance test, M, 1, 5, Day]
   ACTOR (actor list) [User/Development]
   NOTIFY (actor list) []
   FORM_CREATE (form list) []
   FORM_MODIFY (form list) []
   FORM_REF (form list) [Change Request Form]
   EXCEPTION (method, argument) [Ignore, ]
   COMPLETE (method, argument) [End Process, ]
END TASK;
```

Table 6-27 (Cont'd) The workflow specification of software change control process.

6.3. Evaluation and Discussion

In the previous sections, we have shown how WAT can be used for applications in software development by presenting different application scenarios. With the exceptions of two of them, the rest are real cases obtained from several software development teams in Hong Kong and Singapore.

To evaluate the effectiveness of WAT, it was installed for use by two development teams with members playing different roles. Members of these teams were given some initial training and they were asked to use the tool and to note any problem they had with it. They were also asked to evaluate how effective the various features of the tool had helped them in the management of quality. WAT has been installed for six months and throughout the period, some help-desk supports were provided to the users in the sense that we made ourselves available to answer questions they had.

At the end of the period, we selectively interviewed some of the users of the tool. These users include project managers, developers and testers. Their opinion, therefore, represent management and technical developers involved in design to coding to testing in the development life cycle. Our original intention was to also interview the quality manager, however, none of any member of these teams was assigned the responsibility for quality assurance. Such a role is played by the project manager and to a less extent, the test team leaders.

Before the interview, we have developed a questionnaire to evaluate the effectiveness of WAT. The analysis was intended to be qualitative and it was formulated from the Goal/Question/Metric (GQM) paradigm [*Shepperd, 1995*]. Such a goal driven method is used because it aims at identifying what measures could be usefully employed in order to evaluate goal achievement. Therefore, it can be employed for a variety of software projects, the management of change, and also the assessment of services delivered by an IT organization [*Shepperd, 1995*].

To begin with the GQM paradigm, we consider the following as the goal of our tool :

> To help organizations to build a quality management system so as to facilitates ISO 9000 certification.

For applications in software industry, three primary goals were defined :

G1 : To build a quality management system in software organizations (short-term)
G2 : To improve software development processes (mid-term)
G3 : To facilitate the ISO 9001 certification (long-term)

These goals were then decomposed into sub-goals and relevant questions for the questionnaire were designed based on them. The decomposition was shown as in Table 6-28 below :

G1 :	To build a quality management system in software organizations
G1.1 :	To document the quality management system using the tool
Q1 :	Is it convenient and efficient to prepare quality manual and procedures ?
Q2 :	Is it easy to change the documentation ?
Q3 :	Is extra human effort needed for version control and edit history for the documentation ?
Q4 :	Can all the documentation be referenced quickly ?
G1.2 :	To clearly define responsibilities of each staff
Q1 :	Can organizational structure clearly defined ?
Q2 :	Can the line of command and duties easily defined and modified ?
G1.3 :	To implement the system based on documentation
Q1 :	Can documented procedures be implemented easily ?
Q2 :	During implementation, is it easy to detect problems or inadequacies of the documented procedures ?
G2 :	To improve software development processes
G2.1 :	To understand the existing software development process
Q1 :	Can the number of sub-process in the development life-cycle be easily obtained ?
Q2 :	Can the number of task in each sub-process be easily obtained ?
Q3 :	Can the information about each task be easily obtained ?
Q4 :	What are the actors, form created, referenced and modified in each task ?
Q5 :	Exception handling mechanism for each task ?
G2.2 :	To detect shortfalls of the process
Q1 :	Missing information in tasks can be easily detected ?
Q2 :	Deadlock or loop back can be easily detected ?
Q3 :	Improper end of process can be detected ?
G2.3 :	To capture in the process
Q1 :	Is it easy to capture process ?
Q2 :	Was there be a better understanding of process after the process is captured ?
Q3 :	Is it possible to detect problems or inadequacies of the process during capturing ?
G2.4 :	How well to automate and control the process
Q1 :	Do the responsible staff aware of the process ?
Q2 :	Can relevant forms be created, modified and referenced easily ?
Q3 :	Does process execution minimize human effort ?
Q4 :	Does process execution save time ?
Q5 :	Does the process performs as expected ?
Q6 :	Is the execution of process under control ?

Table 6-28 The decomposition of the three primary goals.

G2.5 :	To evaluate process performance
Q1 :	Can progress and status of different process instances easily be checked ?
Q2 :	Can number of completed and processing process instances easily be checked ?
Q3 :	Can expired process instances be monitored ?
Q4 :	Can current workload of each staff be monitored ?
G3 :	To facilitate the ISO 9001 certification
G3.1 :	To understand the ISO 9001 requirements
Q1 :	Is ISO 9001 available ?
Q2 :	Is there any assistant in understanding the ISO requirements ?
G3.2 :	To evaluate current readiness
Q1 :	Is there any method to assess the current readiness ?

Table 6-28 (Cont'd) The decomposition of the three primary goals.

With regard to document control, about all the team members agreed that the WE component provides a central repository to store useful templates and information (quality manual and procedures) in the organization. Management found it much easier and it required less time to prepare, maintain and organize the quality and other documents than before. They found that, using WAT, all the documentation in the quality management system could be modified and referenced easily. Also, they found that the features to support version controls and audit trails were particularly useful in building a documented QMS.

Regarding organizational structure and staff responsibilities, the managers found that the organizational structure database in the WE component was very useful in the definition of actors and their relationships in the organization. It was found that it facilitated task assignments and supported clear specification of lines of commands. Also, having well-defined workflows produced by WAT, most of the team members found that the responsibilities of each team member were better understood and this made better collaboration possible.

System analysts commented that WAT forced them to think and plan thoroughly and logically before the actual implementation. It is because the users we interviewed defined a workflow in the WC component only after they have identified a set of well-defined properties for each task. For example, they have to know what the responsible actors are and what forms need to be created, modified and referenced in the workflow. Because of this, system analysts and developers found that they were more disciplined as they had to

understand and found out what information were lacking and they were not allowed to describe a workflow in an early stage without sufficient information and planning. The possibility of rework was thus minimized. However, they found that it was difficult for them to integrate several related workflows to model a larger workflow scenario. And, they could not model different level of process abstraction as our model currently only supports a single layer of process.

From the interview, the developers found WAT was very helpful and user-friendly because they could view the workflow defined through the graphical workflow editor. And, they could easily locate loop back problems and aware of the incompleteness of the workflow if there was any. Such a visualization of the workflow helps them detect the complex part of it. Also, this facilitates the evaluation of the feasibility and difficulties in the implementation tasks. Therefore, according to the developers, WAT is useful in analysis and design in the software development cycle. In general, less human resources and effort are needed to model the workflow. However, the developers recommended that consistency checking and redundancy avoidance should be available in modeling the workflow. And, this part is not noticeable among the managers.

On the other hand, the managers found that human aspect such as communication, coordination and dependency could be collected and highlighted from the tool. Owing to the emphasis on the significance of actor relationships, the managers were aware of the relationships between actors and they could use this information to plan and manage the organization. And, respective adjustments and modification to the workflow could be done easily. For example, the managers could modify the respective workflow diagram without rebuilding the whole workflow if the management found problems in any existing workflow or wanted to improve it. According to the managers and system analysts, users could use the graphical editor to change any workflow without much effort. And, the revised workflow specification could be generated and imported to the workflow enactment component again easily. Then, the workflow could be implemented and automated based on the new specification. As a result, WAT is capable of minimizing the time for the management and developers to modify a workflow as well as to enforce the execution of the new workflow.

When enacting workflow, users found that they could initiate a process instance simply by the workflow definition stored in the workflow specification database in the WE component. As there were already many workflow facilities (such as automatic process initiation, task notification and assignment) that were built in WAT, developers found that about one weeks' programming effort could be saved. Also, the direct linkage between the workflow specification and application databases could further save their development effort so that they can concentrate on other application specific requirements. For the users, they found those built-in facilities very important and useful to support the dynamic nature of workflow. In particular, they found that automatic task assignment and notification were essential and practical to enforce participation and accountability.

In addition, the managers found that WAT could help them to trace the progress of different instances of processes that have been started. Although processes are implemented in different databases, the managers could monitor process status, as well as the number of completed, processing and expired process instances through the process tracking documents in the workflow specification database. And, it is useful for them to check the current workload and performance of each staff. For the users, they could reference and track the required process information directly and quickly.

After the interview, the users suggested ways of improving WAT. Based on their feedback and recommendations, we found that many problems and difficulties that they encountered were due to their unfamiliarity with the tool. For example, in the course of defining a workflow, they did not know task parallelism can be implemented by unconditional task relationship, automatic tasks can be triggered by agents and complicated rules to determine conditional task branching can be programmed and evaluated by agents. And, during workflow enactment, some users were not aware that they could access relevant documents and raise exception through buttons in the process tracking document. Apart from these, we found that some of the limitations are due to Lotus Notes and we expect that they can be corrected as Lotus Notes improves. For example, owing to the hidden design of Lotus Notes form, our tool cannot support dynamic form design interface in the graphical editor. Also, additional modification is required for external data access other than Notes databases. However, good recommendations have been made to improve the functionality of WAT, such as consistency checking and redundancy avoidance in the process model that requires artificial intelligence. It was also a good suggestion to enhance WAT to

support process integration and task abstraction resulting in a model having multiple layer of process. And, some users commented that exceptions should be made user-driven rather than system-driven. Nevertheless, WAT can still support a reasonable number of real cases obtained from several software development teams.

7. CONCLUSIONS

7.1. Summary

Quality management aims at ensuring products or services to meet the requirements and needs of its users. For a QMS to achieve this, it is important to focus on the design of the right processes and the actual carrying out of the tasks according to the design. The basis for a QMS will best be based on some international quality standards as this will give the confidence of its successful implementation, limit risk and shorten learning curve. Thus, a QMS based on the ISO 9001 can give a firm foundation for process improvement. To build a QMS, it is important to provide a better visualization of processes through explicit recognition of process elements and their inter-relationships. To achieve this, we consider a process to be a sequence of tasks that depend on the cooperation and communication within and between groups of collaborating individuals. This involves complex interaction between task and task, actor and actor and actor and task. In this thesis, a workflow approach is used to manage processes. We have developed a tool, called WAT, that is able to make process visible through well defined workflow elements, structure and representation to support group work. It helps team members to cooperate and communicate, and assists companies to install QMS in compliance with the ISO 9001.

WAT has two major components: workflow capturing (WC) and workflow enactment (WE). The WC component ensures consistent process implementation by specifying processes, task relationships and information flow. Based on the workflow reference model from the WfMC, WAT captures both dynamic (execution paths of a process and flow of documents) and static (task, actor, form properties) aspects of a process. Specifically, WAT supports three different models : (i) an actor model for capturing of organizational structure, (ii) an information model for recording of presentation details, and (iii) a process model for supporting workflow automation and management. Graphical user-interface allows users *say what they do* and makes workflow capturing more understandable and flexible. Therefore, processes can be more adaptive to changes, better controlled and monitored. Organizational structure is captured as well to model the relationships between actors and facilitates dynamic task assignment during process execution. WAT supports both task-based and actor-based workflows. And, it allows

these models to be transformed from one to the other so as to capture different views of organization and explores relationships between workflow elements.

After the workflow is captured, respective workflow specification, with reference to the WIDE workflow specification language can be generated from the WC component and import to the WE component. They are stored in database so that process instances will follow the same specification unless it has been changed. Based on the specification, the WE component, which is implemented with Lotus Notes Release 4.6, helps users to *do what they say* and implement workflow by automating task sequences and bringing responsible development team members into the process. The WE component supports group work by automating task sequences and bringing responsible development team members into process. Moreover, documents can be controlled and shared by the efficient and reliable document manipulation environment. The WE component controls and monitors process execution and provides current and accurate information. It also provides a standard environment to enact processes and quality templates for documentation. As a result, existing procedures can be improved and streamlined by removing redundant data gathering and processing. Through WAT, capabilities to automate and manage processes can be enhanced. Also, quality of product and services can be improved continuously. Therefore, WAT can help company to build a QMS in compliance to the ISO 9001.

In summary, WAT provides numerous benefits including :

- workflow capturing functions to retrieve process information;
- work-to-do list handling functions for notification of assignment of a task;
- process status functions to keep tracks of a specific process or task;
- process maintenance functions to modify process and/or task definitions;
- data handling functions to retrieve workflow relevant system or control data;
- application integration to invoke other external applications;
- process control functions to raise exceptions when necessary.

7.2. Future Work

7.2.1. Workflow improvement and optimization

Achieving workflow automation is the primary goal in workflow management, we can proceed to workflow improvement and optimization. For example, workflow consistency and redundancy checking can be performed with reference to line of command and actor responsibilities in organization structure. Also, we can base on graph theory as a symbolic representation of workflow. Then, by applying and taking advantage of graph theory, we would like to investigate possible improvement in the workflow. For instance, it is possible to detect bottlenecks or uneven workload. Also, it is preferable to seek for better workflow path so as to reduce number of task and minimize actor involvement. As a result, processes can be better monitored and their quality can be assured.

7.2.2. Knowledge-based workflow

Having processes in an organization being well defined automated, controlled and managed, it is desirable to apply intelligence in workflow model and is one of the most interesting significant developments for workflow. A knowledge-based workflow system would use statistical, heuristic, and artificial intelligence to infer correct routing, scheduling, and exception routing, beyond its original definition. That will require some degree of integration with AI systems. For instance, traditional message routing is either pre-defined or condition-based, in order to make routing more effective and efficient, message routing can be based on keywords or patterns learned from previous examples.

7.2.3. Process Measurement

Planning and controlling processes can be achieved by effective measurements. Proper measurement allows an organization to assess processes to ensure they adhere to original performance requirements continuously. This will provide assistance to assessing, monitoring, and identifying improvement actions for achieving a set of company quality and productivity goals. We can use quantitative data collected as an indicator of process problem areas. The measurement data provides information for investing wisely in tools for quality and productivity improvement. They can be used as a quality threshold mechanism, a check on conventional costing methods, a technique for monitoring project progress and checking on the functional growth of the system, and as a method for ensuring that quality does not degrade during maintenance. Ultimately, the inherent quality of a product and the efficiency of a development effort can then be identified, quantified, and improved.

7.2.4. Internet Workflow Application

As a growing business, companies are eager to use Internet to tap new markets, increase efficiency and speed up communication both inside and outside the company to cement relationships with customers and suppliers across different geographic areas. Therefore, Internet and collaborative technologies are driving key business trends. Companies can take advantage of the Internet quickly and cost-effectively. They can gain significant competitive advantage and bottom-line results by exploiting Internet technologies in high value applications. These business solutions will help companies track contacts, orders, and documents; keep customers informed; and resolve problems without picking up the phone. Also, project information, status and on-line discussions with the entire organization can be shared. Moreover, global enhanced, seamless communication is established throughout the organization and related parties using Internet electronic mail.

8. BIBLIOGRAPHY

Armenise P., Bandinelli S., Ghezzi C., Morzenti A., "Software Process Representation Languages : Survey and Assessment," Proceedings, 4th Conference of Software Engineering and Knowledge Engineering, Los Alamitos, Calif., pp. 455 - 462 (1992).

Bernard Moulin, Brahim Chaib-draa, Louis Cloutier, "A Multi-Agent System Supporting Cooperative Work Done by Persons and Machines," 1991 IEEE International Conference on System, Man, and Cybernetics, 'Decision Aiding for Complex System', NY, USA, pp. 1889 - 1893 (1991).

Berthold Reinwald, Mohan C., "Structured Workflow Management with Lotus Notes Release 4", Digest of Papers. COMPCON '96. Technologies for the Information Superhighway. Forty-First IEEE Computer Society International Conference, Santa Clara, California, 25-28 Feb, 1996, pp. 451 - 457 (1996).

Bussler C., Jablonski S., Schuster H., "A New Generation of Workflow-Management-Systems: beyond Taylorism with MOBILE," SIGOIS Bulletin, 17(1), pp. 17-20 (1996).

Casati F., Grefen P., Pernici B., Pozzi G., Sanchez G., "Conceptual Modeling of WorkFlows," OOER'95 : Object-Oriented and Entity-relationship Modeling, 14th International conference, Bond University, Gold Coast, Queensland, Australia, Springer-Verlag, pp. 341-353 (1995).

Casati F., Grefen P., Pernici B., Pozzi G., Sanchez G., "WIDE Workflow model and architecture," Workflow Management Coalition, April 1996 (1996).

Casati F., Grefen P., Pernici B., Pozzi G., Sanchez G., "Deriving Active Rules for Workflow Enactment," Database and Expert System Applications, 7th International Conference, DEXA'96, Zurich, Switzerland, pp. 94 - 115, (1996).

Chan P.W., "Installing an ISO 9001 accredited software quality management system," SQM'93, First International Conference on SQM, Southampton, UK, pp. 13 - 26 (1993).

Chidung Lac, Jean-Lue Raffy, "A Tool for Software Quality," Proceedings of the IEEE Second Symposium on assessment of quality software development tools, New Orleans, L.A., 27-29 May, 1992, pp.144 - 150 (1992).

Christoph Bubler, Stefan Jablonki, "Implementing Agent Coordination for Workflow Management Systems Using Active Database Systems," Forth International Workshop on Research Issues in Data Engineering, Active Database Systems, Los Alamitos, CA, USA, pp.53 - 59 (1994).

Christoph Bubler, Stefan Jablonski, "An Approach to integrate workflow modeling and organization modeling in an enterprise," Proceedings Third Workshop on Enabling Technologies : Infrastructure for collaborative enterprises, Los Alamites, CA, USA, pp. 81 - 95 (1994).

Ciancarini P., "Modeling the Software Process using Coordination Rules," The Fourth Workshop on Enabling Technologies : Infrastructure for Collaborative Enterprises, Los Alamites, CA, USA, 20-22 April, 1995, pp. 46 -52 (1995).

Coleman D., Khanna R., "Groupware : Technology and Applications," Prentice Hall (1995).

Daniel K.C. Chan, Jochem Vonk, Gabriel Sanchez, Paul W. P. J. Grefen, Peter M.G. Apers, "A Specification Language for the WIDE Workflow Model," Conference on Advanced Information Systems Engineering CAiSE 97, Barcelona, Catalonia, Spain, June 16-20, 1997 (1997).

Dennis R. McCarthy, Sunil K. Sarin, "Workflow and Transactions in InConcert," Data Engineering Bulletin, Vol. 16, No. 2, pp. 53-56 (1993).

D.K.C. Chan, K.P.H. Leung, C.Y. Yan and K.C.C. Chan, "Liaison : a Workflow Model for Novel Applications," In Proceedings of the 1998 Asia-Pacific Software Engineering Conference (APSEC'98), IEEE Computer Society Press, pp. 144-152 (1998).

Dr. Keith C.C. Chan and Yan Ching Ying, "A Workflow Automation Tool for ISO 9000 Compliant Quality Management," ISO 9000 and Total Quality Management, Proceedings of the Third International Conference, pp. 53-61 (1998).

D. Wodtke, J. Weissenfels, G. Weikum, A. Kotz Dittrich, "The Mentor Project: Steps Towards Enterprise-Wide Workflow Management," IEEE International Conference on Data Engineering, New Orleans, LA, USA, 26 Feb - 1 Mar, 1996, pp. 556-565 (1996).

Eric S.K. Yu, "From E-R to 'A-R' - Modeling Strategic Actor relationships for business process reengineering," International Journal of Cooperative Information Systems, Vol. 4, No.2 & 3, pp.125-144 (1995).

Friedemann Schwenkreis, "APRICOTS - Management of the Control Flow and the Communication System," Proceedings of the 12th IEEE Symposium on Reliable Distributed Systems, Princeton (1993).

Gary Born, Process Management to Quality Improvement, John Wiley & Sons (1994).

Georgakopoulos D., Hornick M., Sheth A., "An Overview of Workflow Management : From Process Modeling to Workflow Automation Infrastructure," Distributed and Parallel Databases, 3, pp. 119-153 (1995).

Gerhard Chroust, Stefan Hardt, "Executing Process Models Activity and Project Management," Proceedings IEEE Symposium and Workshop on Engineering of Computer-Based Systems, pp. 364 - 370 (1996).

Gianluigi Caldiera, "Impact of ISO 9000 on Software Maintenance," Proceedings of Conference on Software Maintenance, CSM-93, Montrical, 27-30 Mar 1993, pp. 228-230 (1993).

Gulla J. A. and Lindland O. I. , "Modeling Cooperative Work for Workflow Management," Advanced Information System Engineering 6th International Conference CAiSE '94, Manchester, UK, 12-15 May, pp.53 - 65 (1994).

Hartel P., "Modeling Business Processes over Object," International Journal of Cooperative Information Systems, Vol. 4, No. 2 &3, pp. 165 - 188 (1995).

Helmut Wächter, Andreas Reuter, "The ConTract model," A.K. Elmagarmid (ed.): Transaction Models for Advanced Applications, Morgan Kaufmann Publishers (1992).

Herb Krasner, "Groupware Research and Technology Issues with Application to Software Process Management," IEEE transaction on Systems, Man, and Cybernetics, Vol. 21, No 4, pp.704-712 (1991).

ISO 9000, "Quality management and quality assurance standards : Guidelines for selection and use, " International Organization for Standardization, 15 March 1987 (1987).

ISO 9000-3, "Guidelines for the application of ISO 9001 to the development, supply, and maintenance of software," International Organization for Standardization, 1 June 1991 (1991).

ISO 9001, "Quality systems : Model for quality assurance in design, development, production, installation, and servicing," International Organization for Standardization, 1 July 1994 (1994).

Jablonski, Stefan, "Workflow Management : modeling, concepts, architecture and implementation," International Thomas Computer Press (1996).

John McCarthy, "The state-of-the-art of CSCW : CSCW systems, cooperative work and organisation," Journal of Information Technology, Vol. 9, No 2, pp. 73 - 83 (1994).

Kappel G., Lang. P., Rausch-Schott S., Retchitzegger W., "Workflow Management Based on Objects, Rules and Roles," Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society, Vol. 18, No. 1, pp. 11 -18 (1995).

Kari Kuutti, Tuula Arvonen, "Identifying Potential CSCW Applications by Means of Activity Theory Concepts: A Case Example," Proceedings CSCW92, Toronto, Canada, 31 Oct - 4 Nov, pp.233-240 (1992).

Kevin Crowston, "Modeling Coordination in Organizations," Artificial Intelligence in Organization and Management Theory, Michael Masuch, Massino Warglien, pp. 215 - 234 (1992).

Lawrence P., "Workflow Handbook 1997 Ed.," John Wiley & Sons Ltd., (1997).

Meichun Hsu, Charly Kleissner, "ObjectFlow : Towards a Process Management Infrastructure," Distributed and Parallel Databases 4, pp. 169-194 (1996).

Michael G. Jenner, "Software Quality Management and ISO 9001", John Wiley & Sons, Inc. (1995).

Mohan C., Alonso G., Guenthoer R., Kamath M., "Exotica : A Research Perspective on Workflow Management Systems," Data Engineering Bulletin (Special Issue on Infrastructure for Business Process Management), Vol. 18, No. 1, pp.19-26 (1995).

Neal Whitten, "Managing Software Development Projects : Formula for Success", John Wiley & Sons, Inc. (1995).

Ranjit Bose, "CMS : A Knowledge-based Tool for Intelligent Information Systems Application Development," IEEE International Conference on Developing and Managing Intelligent System Projects, Washington, DC, USA, 29-31 Mar 1993, pp. 85-92 (1993).

Raul Medina-Mora, Terry Winograd, Rodrigo Flores, Fernando Flores, "The Action Workflow Approach to Workflow Management Technology," Proceedings CSCW 92, Toronto, Canada, 31 Oct - 4 Nov, pp. 281 - 288 (1992).

Schmauch, Charles H. ,"ISO 9000 for software developers," ASQC Quality Press (1995).

Stef Joosten and Sjaak Brinkkemper, "Fundamental Concepts for Workflow Automation in Practice," ICIS'95 conference, Amsterdam (1995).

Sunil K. Sarin, "Object-Oriented Workflow Technology in InConcert," Digest of Papers, COMPCON'96, Technologies for the Information Superhighway, Forty-First IEEE Computer Society International Conference, Santa Clara, CA, USA, pp. 446-450 (1996).

Thomas Kreifelts, Uta Pankoke-Babatz, Frank Victor, "A model for the coordination of cooperative activities," International Journal of Cooperative Information Systems, Vol. 4, No. 2 & 3, pp. 85 - 99 (1995).

Welsh J., Han J., "Software Documents : Concepts and Tools," Software - Concepts and Tools, Vol. 15, pp. 12-25 (1994).

Yan Ching Ying and Keith CC Chan, "A Lotus Notes Implementation of a Workflow Automation Tool for ISO 9001 Certification," Sixth International Conference on Software Quality Management (SQM' 98), pp. 161 – 172, April Amsterdam (1998).