# WORKFLOW-DRIVEN DESIGN CHAIN MANAGEMENT FOR COLLABORATIVE ENGINEERING OF TECHNOLOGY-INTENSIVE PRODUCT

XIONG XIAOHUA
(B.Eng., M.Eng., Huazhong Univ. of Science & Technology, P.R.China)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

# Acknowledgement

I would like to gratefully acknowledge the enthusiastic supervision of my supervisors, Dr. Liu Zhejie from the Data Storage Institute (Singapore) and Professor Wong Yoke San from the National University of Singapore. Their enthusiasm, inspiration and great effort lead me through the whole research process.

Many thanks are given to the members of Magnetic Recording Channel division, especially Li Jiangtao, Long Haohui, for their kind help during the period of my stay in Data Storage Institute.

Finally, I would like to thank my family. The encouragements and supports from my parents and brothers give me the strength to head forward. My wife, Wang Li, her understanding and accompanying get me through the difficult times. To them I dedicate this thesis.

# Table of Contents

# Summary

Due to increasing global competition, collaborative product design is envisaged as the strategy to reduce cost, ensure quality, and shorten time-to-market, especially for technology-intensive products, such as hard disk drives. It can enable teams of designers across enterprise boundaries to address total lifecycle issues at early design phases. In order to support such type of design processes, a distributed collaborative environment is necessary, in which the effectiveness of teams will translate directly to improved cost savings and competitiveness.

Workflow management system is a technology that has received much attention in recent years. It can streamline, coordinate and monitor an organizational process involving human and automated tasks spread across multiple enterprises with heterogeneous (existing and new) computing environments.

Adopting workflow management technology can greatly improve distributed collaborative design efficiency. In this thesis, a distributed collaborative design framework that is based upon and driven by workflow is presented. Under this framework, the design process can be effectively organized, executed, controlled and monitored at workflow management level, leading to short time-to-market which is a key factor for the success of a new product. The allocation of resources, such as material, human, software, etc., can also be optimized so that the development cost will be lowered.

In this framework, process modeling, process dynamics and information exchange are of major concern because these are the basic parts for constructing such a design environment. The process modeling is discussed with respect to process meta-model and organization meta-model. As to the process communication, the asynchronous and synchronous communications involved within intra- and inter-processes are examined. To adapt to the frequent changes occurring in design activities, dynamic process management is proposed to minimize the cost of changing processes. Combined with workflow technology, a workflow-driven communication method supporting process modeling and efficient information exchange between tasks is studied. Furthermore, dynamic workflow is investigated as one major difficulty in implementing this framework.

The research work in this thesis has special reference to the design chain activities for development of hard disk drive products at component, sub-system and system levels.

# List of Figures

# List of Tables

# Chapter 1. Introduction

Confronted with global competition and rapidly changing customer requirements, manufacturers face an increasingly arduous task in developing new products. Product development is becoming more reliant on geographically dispersed, multi-disciplinary designers during design, manufacturing, and delivery processes. Collaborative product design has been the perceived strategy for the companies. Because the activities of the design process determine both product competitiveness and cost in collaborative commerce, collaborative product development is receiving much attention. By making the entire collaborative product design process work more effective, manufacturers are taking a vantage position to manage product quality, cost and time-to-market.

The research effort presented in this thesis involves the development of a workflow-driven distributed collaborative engineering design and analysis framework. Workflow technology is introduced to organize, execute, control and monitor the product development process, leading to short time-to-market and lower cost.

## 1.1 Background and Research Motivation

Under the environment that is increasingly globalized, the design of complex artifacts and systems requires the collaboration of multidisciplinary design teams using various commercial and non-commercial engineering tools, such as CAD/CAE tools, modeling, simulation and optimization software, engineering database, and knowledge-based systems. Individuals or individual groups of multidisciplinary design teams usually work in parallel and separately with various engineering tools, which are located at different sites, often for quite a long time. At any moment, individual members may be working on different versions of a design or viewing the design from various perspectives, at different levels of details. These mean that design work may be distributed either physically (e.g. it may be carried out at different places) or temporally (e.g. it may be carried out at different times).

In order to facilitate the aforementioned design process, it is necessary to have an efficient distributed collaborative design environment. This environment should not only automate individual tasks, in the manner of traditional computer-aided engineering tools, but also enable individual members to communicate with one another, share information and knowledge, collaborate and coordinate their activities within the context of related design projects.

A large number of papers on distributed collaborative design have been published. Numerous commercial products are also available on the market, which implement partial features of distributed collaboration concepts. In [3], the research on distributed collaborative design falls into two categories. The first one tries to offer the theoretical models. For example, in [4] a distributed object-based modeling and evaluation (DOME) framework for product design was proposed. Also Case and Lu proposed a discourse model used in software environments that provides automation support for collaborative engineering design [5]. The second category tries to implement the idea of distributed collaborative design in a practical way. In [6] a prototype was proposed to implement web-enabled feature-based modeling in a distributed environment. *Cybercut* is one of the first web-based design systems for fabrication, which was developed at the University of California at Berkeley [7]. An agent-based prototype implementation was proposed in the design of a portable CD player [8]. To seamlessly share CAD information across an enterprise, major CAD suppliers have introduced a software system called product development management [9, 10, and 11].

While much of previous research has focused on the development of technology and tools to support distributed design, there is a lack of awareness and understanding of the entire design process. This thesis concerns the development of design support at the process level.

Since design is a complex process, process control should be an indispensable part during the construction of distributed collaborative design framework. Workflow management system is a technology that has received much attention in recent years. It can streamline, coordinate and monitor an organizational process involving human and automated tasks spread across multiple enterprises with heterogeneous (existing and new) computing environments. The basic idea in workflow management is to capture formal descriptions of the processes and to support the automatic enactment of the processes based on these formal descriptions. It can meet requirements proposed by distributed collaborative design very well.

The motivation for conducting research into distributed collaborative design using workflow is summarized in two ways:

- The benefits of distributed collaborative design are potentially great;
- Workflow technology can effectively support process management that cannot be provided by traditional technology.

## 1.2 Research Objectives

In contrast to the ongoing research efforts that emphasize more on technology and tools, the proposed research focuses on the process level. The overall aim of the research is to develop a workflow-driven infrastructure supporting process management for distributed collaborative design teams. This comprises the following objectives:

- Identification of requirements in distributed collaborative design activities with respect to process control;
- Construct an infrastructure supporting distributed collaborative design by using workflow technology;
- Application of such infrastructure to collaborative product design process in practice to demonstrate the feasibility of the proposed method.

In order to construct such a workflow-driven infrastructure, process modeling, intra- and inter-process communications are discussed and analyzed in detail.

## 1.3 Organization of Thesis

The thesis is organized in following order:

In Chapter 2, firstly we discuss the feasibility of adopting workflow technology in distributed collaborative design with respect to design process management, including process modeling, process communication and monitoring. Then a detailed literature review of distributed collaborative design and workflow technology will be given. Also the source, benefits and existing research of each are presented.

In Chapter 3, the requirement of distributed collaborative design is analyzed in detail. And then based on the results of analysis, a workflow-driven infrastructure is proposed. The two main parts constructing the infrastructure, namely workflow modeling and process communication mechanism, are studied. The related issues, such as service management and session management, are also discussed in this chapter.

The dynamic workflow is discussed in chapter 4. A dynamic workflow change management method based on workflow instance is proposed to resolve the dynamic changes that happen in the design process frequently.

In Chapter 5, a design scenario is presented as a case study using the proposed infrastructure and technologies combined with CoCADE (a distributed CAD/CAE tool).

Chapter 6 summarizes the results of the presented work. The future research direction is also given.

The establishment of such a workflow management framework is shown in Figure 1.1.

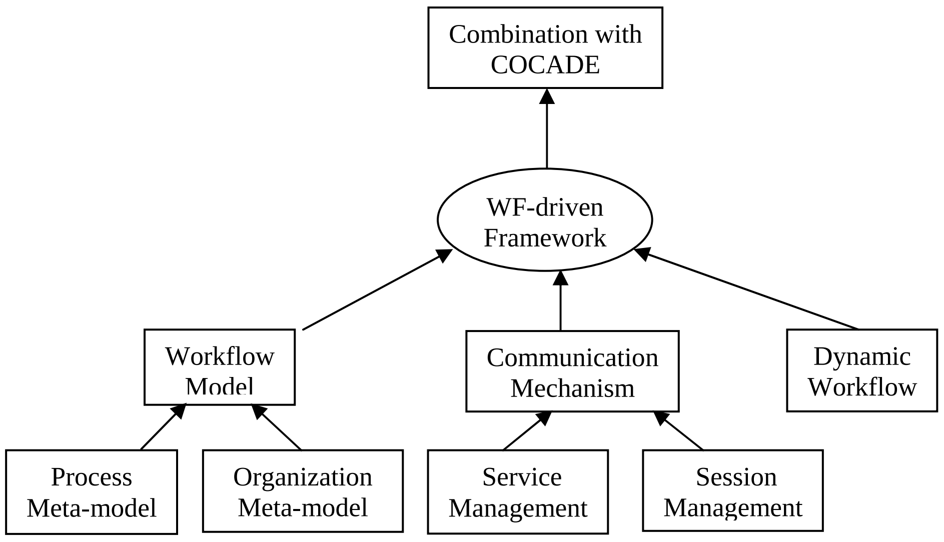


Figure 1. 1 Establishment of Workflow Management Framework

# Chapter 2. Literature Review

The practice of distributed collaborative design is ever increasing. Advances in the computing world and particularly the creation and growth of the Internet in the past few years have facilitated the growth of distributed design teams and distributed work. However, the practice is often inefficient [25, 26, and 27]. At the same time, workflow technology has been more and more widely used as an effective process management tool for rapid development. In this chapter, we will discuss the feasibility of adopting workflow technology in distributed collaborative design environment in detail. Related work is also presented.

## 2.1 Integrate Distributed Collaborative Design with Workflow Technology

Many of the current distributed collaborative design processes, whilst well defined, are either implicit in the operation of the organization or only informally recorded. This lack of explicit representation makes it difficult to carry out collaboration among different participants who may be dispersed geographically, to track the progress of active processes, to modify ongoing processes in response to situational changes, or to adopt newly defined processes. Assurance that processes are executed successfully and efficiently is often difficult or impossible to obtain.

The workflow community advocates the use of explicit models and representations of processes, along with automated tools to support the activation and ongoing management of workflow processes. Structured management of processes can provide benefits in several ways. Articulation of explicit processes can help to standardize operation, thus reducing the likelihood of introduced errors. Explicitly represented processes can be tuned to improve efficiency and effectiveness. Automated tools for process management hold promise for deeper insight into current and planned operations, through the provision of timely updates on progress and status. Such enriched understanding of operation processes will lead to better-informed decision-making by users. Automation can also provide adaptive capabilities that enable agility of operation and more effective use of resources. Such characteristics are critical for operation in dynamic environments where requirements and conditions can change rapidly and unpredictably.

Recently, in order to make the product design processes more efficient, a Design Chain Operations Reference (DCOR) Model has been launched by The Supply Chain Council, which is one of the industry’s leading professional organizations responsible for the Supply Chain Operations Reference (SCOR) Model (http://www.supply-chain.org/). DCOR can be used to analyze a company’s design chain processes to determine where weaknesses exist, identify principal process elements found throughout the design chain and link them to performance attributes and metrics. It will serve users the need to evaluate their processes and their current practices, and to assess gaps and requirements. It can be seen that as an effective process management method, workflow technology can play an important role in the establishment of DCOR. This is also one of the contents of the further research work.

In the following section, the literature review of these two domains is presented.

## 2.2 Distributed Collaborative Design Review

The need for distributed collaborative design can be readily justified, as the complexity of design continues to increase, demanding larger teams and higher productivity. Furthermore, the current shortage of qualified design engineers requires companies to search for manpower from different parts of the world, and in many cases it is more convenient not to relocate them. In addition, there is the need for experience sharing among members of the design team. Figure 2.1 shows a scenario of distributed collaborative design.

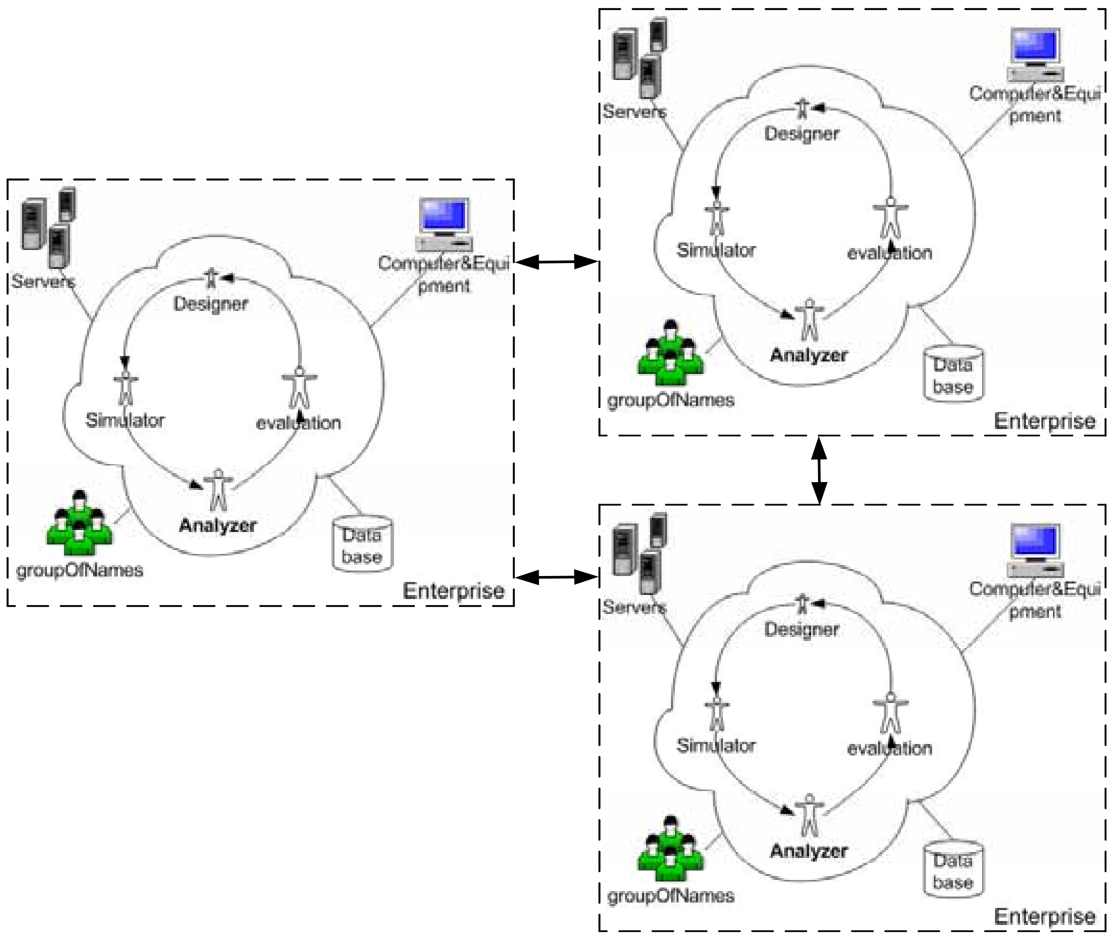


Figure 2. 1 A Distributed Collaborative Design Scenario

By adopting distributed design, great benefits can be obtained, e.g., shorter product development cycle, better product design and manufacturability, the lower cost of product development. For technology-intensive product, the benefits can be more obvious. As shown in Figure 2.2, a hard disk drive contains many parts. During the design process, many engineers from multiple disciplines are needed. The participants involved in the design are closely interdependent and interactional. Many parts are also highly technology-intensive, such as head, media, PCB and spindle motor.

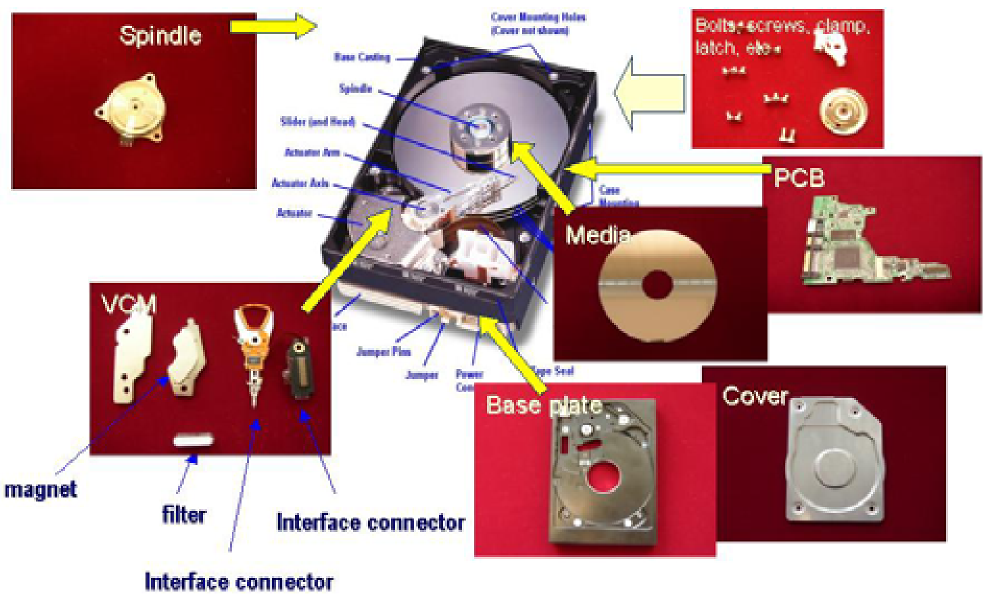


Figure 2. 2 Structure of Hard Disk Drives

Arising from the need for distributed and collaborative design, many research works have already been done. Most of these researches have focused on the role of tools and technology. Within the commercial domain, collaborative design tools are common within the AEC (Architecture, Engineering, Construction) sector as redlining and mark-up tools. More sophisticated tools include Magics Communicator from Materialise (www.materialise.be/communicator/) and Co-Create's OneSpace (www.cocreate.com/) which allow dispersed designers to share design models and modify design work synchronously. Dedicated collaboration tools used to bring distant people together include NetMeeting from Microsoft (www.mircosoft.com/windows/netmeeting/default.asp) and Groove Networks'

Groove (www.groove.net/) which provide different channels for communication and support for storing information. Shared workspaces such as Portfolio Wall (www.aliaswavefront.com/) allow designers to share ideas, primarily within the product design environment. Much of the research within the collaboration tools domain comes from CSCW (Computer Supported Cooperative Work), yet lacks the sophistication of commericial tools. The Grouplab research group at Calgary have developed Groupware tools such as the Notification Collage (www.cpsc.ucalgray.ca/grouplab/) that lets people post multimedia items to a communal, distributed bulletin board. Within the design field, [33] proposes a data visualization tool to improve communication in distributed product development.

Technology has progressed dramatically in recent years. Cheng and Kvan in [34] state the importance of collaboration structure in “finding the best fit between technology and group design for design collaboration”, adding factors that include collaborators’ profiles, mutual value of produced information and logistical opportunities. Further, [35] describes 13 communication roles that support design collaboration, stating that they could form the basis for prescriptive design methods which support communication. These roles focus on interaction and knowledge exploration. They exist within several sub-groups or boundary types including organization, task and discipline. Such distinctions highlight important contexts within the communication domain. Other research has aimed to develop support for collaborative design. [36] proposes a framework for a collaborative design environment which has three major components: a shared workspace, application domain and data management facility. [37] studies designers’ information access and storage to implement IT support. Such studies highlight the importance of building a sufficient knowledge and information infrastructure for design work.

However, certain applications have attempted to cater for a wide range of activities without taking into account of the specific requirements of the design process. As to the design process control level, established design processes from [38], [39] and French [40] are still pervasive in distributed work and although powerful, were not designed with distributed design specifically in mind. The Process Handbook at MIT (http://ccs.mit/edu/ph/) includes both design and non-design processes to resolve collaborative conflicts. Concurrent engineering processes also exist which may be used to rationalize distributed team effort. In [41], Prasad details processes for complexity, performance and planning that may be used to support distributed and collaborative design.

There are few cases of collaboration or communication processes specifically for design work. Lu and Cai in [42] propose a model specifically to manage different perspectives and design conflicts. Other research has focused on different areas of collaborative design such as negotiation [43] while Vadhavkar and Pena-Mora in [44] look at organizational processes as part of their team interaction space. McGrath in [45] focuses on the interaction and performance of groups and Olson et al. [46] focus on the effects of technology on group work. Lipnack and Stamps in [47] write on the concept of virtual teams while DeSanctis and Monge in [48] review communication processes for virtual organizations within the computer-mediated communication field.

With the emergence of workflow technology, the above-mentioned state can be changed. By using workflow technology, the distributed collaborative design process can be effectively controlled and managed.

## 2.3 Workflow Management Systems

*Workflow management* has received much attention in recent years. It aims to streamline, coordinate and monitor an organizational process involving human and automated tasks spread across multiple enterprises with heterogeneous (existing and new) computing environments.

*Workflow management systems (WFMS)* are software systems where processes are modeled and enacted and are called *workflows*. *Workflow modeling* consists of creating formal descriptions of workflows, called *workflow models,* which capture various aspects of workflows, including the steps to be carried out, assignment of the steps to processing entities, and relationships that exist among the steps, such as control and data flow. *Workflow enactment* involves the coordination and control of the execution of the different workflow steps according to the corresponding workflow models. Apart from the modeling and enactment of workflows, WFMS often provide additional functionality, such as monitoring tools that allow users to keep track of currently executing workflows and facilities that allow users to analyze workflow models as well as workflow executions.

By providing the aforementioned functionalities, WFMS can greatly benefit the product development process occurring in a distributed collaborative environment. The benefits can be divided into business-related and technology-related. One of the obvious key business-related benefits is the reduction of lag time in routing work tasks among people. This may result in increased productivity and reduced costs. Concerning the technology-related benefits, WFMS can be regarded as meta-programming tools that can be used to develop flexible distributed application, where the basic instructions used are autonomous, heterogeneous applications.

### 2.3.1 Basic Concepts

Workflow management involves the modeling and enactment of workflows. A workflow can be either basic or complex with the latter consisting of further sub-workflows. Thus we can use activities with arbitrary granularity to express the whole workflow, which means an activity can be a trivial task such as a simple file saving operation or it can be a complex task, such as a part design or subassembly process. Activities are executed by processing entities, which may be people or software tools. Figure 2.3 shows the details of some workflow concepts.

In this figure:

- Process is what is intended to happen
- Process Definition is a representation of what is intended to happen
- Process Instance is a representation of what is actually happening
- Work Item is a task allocated to a workflow participant
- Invoked Application is a computer tool/application used to support an activity

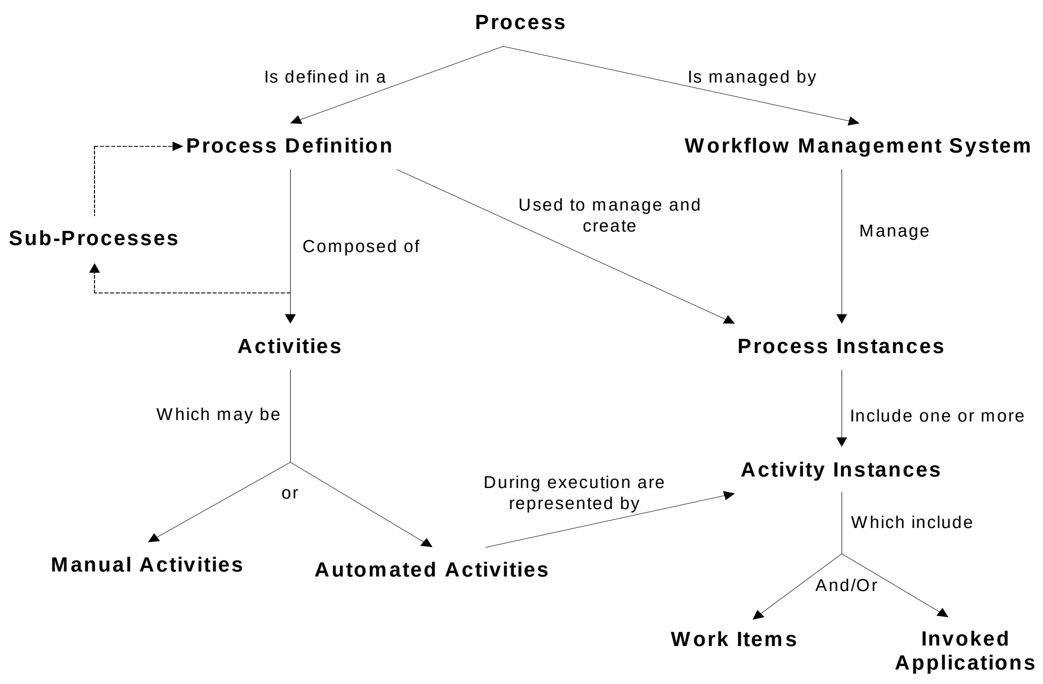


Figure 2. 3 Workflow Concepts

### *Workflow Modeling*

Workflow modeling denotes the creation of a workflow type, which is a formal description of various aspects of a workflow, such as the activities to be carried out, identified processing entities performing the activities, and dependencies/relationships among the activities (e.g., data flow among the activities). An arbitrary number of instances can be created from a workflow type.

### *Workflow Enactment*

Workflow enactment involves the coordination of the execution of the activities that workflows consist of according to the workflow model. More precisely - since processing entities carry out activities, workflow enactment requires coordination among the processing entities in executing the activities.

### *Workflow Management System*

The Workflow Management Coalition (WfMC), a standardization organization, defines a WFMS as “a system that defines, creates, and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants, and where required, invoke the use of IT tools and applications”.

A WFMS consists of two main functional components: a build-time component and a run-time component (as shown in Figure 2.4). The build-time component provides support for the development and persistent storage of workflow types. It offers to the workflow modeler a workflow modeling language in which workflow types can be expressed through appropriate tools, such as editors, browsers, and parsers/compilers. Besides workflow modeling, the WFMS should also support organizational modeling, which includes the specification of information about processing entities. Furthermore, organizational relationships among actors may have to be defined in order to enable the specification of activity assignment to actors based on organizational relationships, which can also be very useful in automatic workflow modeling. Besides the aforementioned functionality, the build-time component may provide additional facilities to simulate workflow executions and analyze workflow types.

The run-time component focuses on the execution of the workflow. It supports the creation and enactment of workflow instances according to the workflow types created with the build-time component. During workflow enactment, the run-time component interacts with the actors in order to ensure that the workflows are executed as prescribed by the corresponding workflow types. The WFMS usually provides monitoring tools that allow the workflow administrator to keep track of the execution progress of workflows. Also, logs about workflow executions are recorded by the WFMS for the purpose of identifying bottlenecks, improving workflow types, etc.

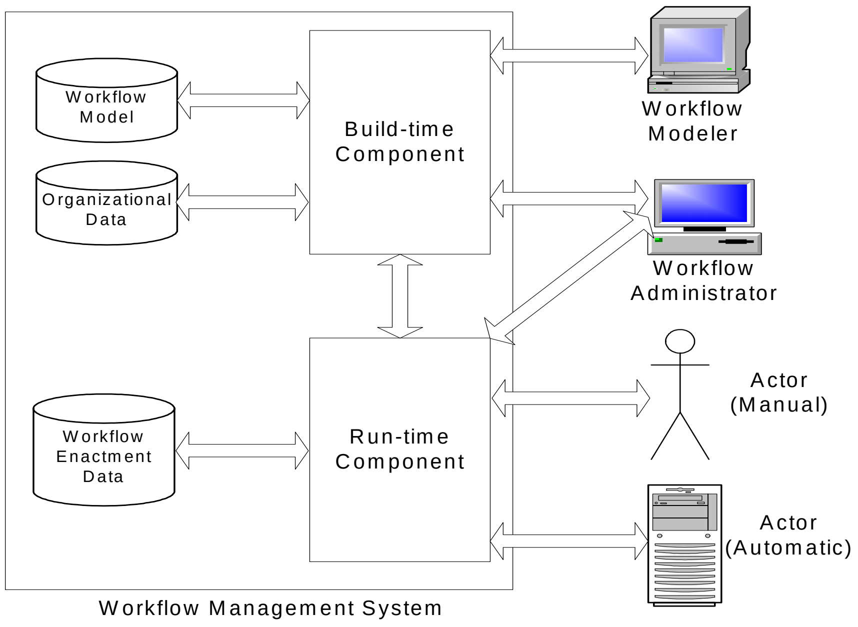


Figure 2. 4 Functional Components of WFMS

### 2.3.2 Related Work on Workflow Management System (WFMS)

Workflow management systems have received widespread attention since the advent of this technology in the late 1980s. The association for Information and Image Management (AIIM) (http://www.aiim.org/) estimated the worldwide revenue for workflow technologies to grow from $4.3bn in 2000 to $8.3bn in 2003 at a compounded annual growth rate of 31%. Recently, WFMSs have spread beyond the administrative environment and can also be found as embedded software components, which enhance existing application packages (e.g., ERP systems) as well as infrastructure components (such as application servers) with process management functionality.

In this section, we discuss two particular aspects of WFMSs: Coordination and integration.

***WFMS as Coordinating Systems***

From a conceptual perspective, the purpose of a WFMS is the coordination of all entities involved in the execution of a defined process. Coordination can be defined as the management of dependencies between activities. In [60], the authors classified dependencies and related coordination processes in a framework shown in Table 2.1.

Table 2. 1 Dependencies and Coordination Process

| Dependency | Description | Coordination Process |
|---|---|---|
| Prerequisite | An activity depends on the output of another activity | Activity ordering |
| Shared Resource | Multiple activities require the same resource | Resource allocation |
| Simultaneity | Two activities must be performed at the same time | Activity synchronization |
| Task/Subtask | Top-level goal is dependent on the achievement of other goals | Goal decomposition |

WFMSs address these dependencies through their coordination functions. Prerequisite dependencies between activities are managed through the supervision of control and data flows. Shared resources are managed through scheduling and staff resolution mechanisms. Task/Subtask dependencies are addressed through the hierarchical composition and decomposition of workflow models. Simultaneity constraints are observed through event-based synchronization of processes and activities. Through the automation of these coordination functions, WFMSs support several efficiency goals of the enterprise, such as process efficiency, resource efficiency, motivation efficiency, etc.

It is apparent that the benefits of WFMSs increase with the number of coordination tasks that can be automated through a workflow system. The number of coordination tasks varies with the granularity of the components controlled through the workflow system as well as with the type of the process controlled by the workflow system.

**WFMS as Integration Systems**

Integration is regarded as one of the primary goals during information system design. Literally, integration means to form, coordinate, or blend something into a functioning or unified whole with existing segregation. Two distinct types of integration can be distinguished:

1. Integration through connection. This occurs if a new system is created through the creation of links between disparate, but logically connected entities or sub-systems. Typically this is an ex-post integration of existing systems, such as the integration of enterprise applications through a WFMS.

2. Integration through combination. This occurs if similar system elements are combined, thus leading to a decreased number of elements and relationships within the system (in the sense of abstraction). Typically this form of integration happens during the conceptual design phase of an information system, for example the development of a complex application with an integrated workflow layer for the transport of application data.

In [61], the author names reduction of redundancy, increased system consistency and integrity, and better decision support through timely information supply as the main goals of integration efforts. Integration can be characterized by the information type, object, direction, scope, and realization of integration. In terms of the dimensions of integration, data integration, function integration, process integration, and object integration can be distinguished. Integration can extend across an organization horizontally (such as cross-organizational processes), or vertically (such as reporting data flow up the organizational hierarchy).

The design of a WFMS creates integration requirements that can be internal and external. Internal integration requirements concern those systems that a WFMS needs to be connected to, in order to ensure the functionality of the core workflow system. External integration requirements exist with regard to systems that either invoke the workflow system from the outside (embedded usage), or systems that are invoked by the WFMS.

Figure 2.5 summarizes the internal and external integration requirements of WFMS.

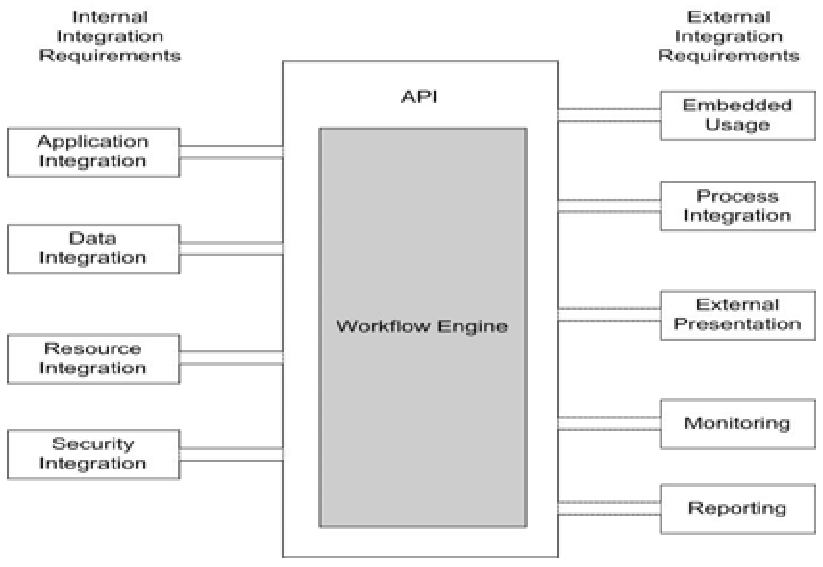


Figure 2. 5 Workflow Integration Requirements

**Existing Systems**

More than a concept, there are already some existing systems which can support the management of a process, such as *NELSIS* [13], *Ulysses* [14] & *Odyssey* [15], *WELD*

[16], *OmniFlow* [17] and *ASTAI(R)* [18]. But these systems have shortcoming individually. For example, in *NELSIS*, each activity abstracts the (partial) functionality of an automation tool, as well as controls its execution parameter. Also hierarchical description of activities is also supported. But the human actions are not included in the framework. The resources are also not effectively arranged around the related activities. In *Odyssey*, the plan created by the *Minerva* module needs to be defined carefully and all the related data need to be included at the design phase. It does not support dynamic change; thus, it is not flexible enough for the design process that is full of changes. The absence of the capability supporting dynamic changes fully can also be found in the PDM systems that are available commercially, e.g., SmarTeam. In this thesis, these problems are to be addressed.

## 2.3 Summary

In this chapter, we describe and review distributed collaborative design and workflow technology. Reasons are also given for combining workflow technology with distributed collaborative design. It is obvious that to accommodate engineering design containing complex CAD/CAE activities, the WFMS should employ workflow technology as a coordination tool as well as an integration tool. It should be able to not only coordinate design tasks and processes, but also integrate the available CAD/CAE tools into the design environment.

In the following chapter, an infrastructure driven by workflow technology is proposed to support distributed collaborative design well.

# Chapter 3. Workflow-driven Framework

In this chapter, a framework that supports distributed collaborative design is proposed, which is built upon and driven by workflow. Then the workflow modeling and communication are detailed as two main parts of the framework. The dynamic characteristic of this framework is also studied.

## 3.1 Requirement Analysis

According to [19], the requirements for the distributed collaborative design enterprise architecture include the following:

1. Have an Internet-base architecture
2. Manage/track concurrent change in a non-obtrusive manner
3. Mange product configuration across multiple organizations
4. Find and retrieve managed objects across multiple organizations
5. Rapidly define, create, and manage design objects and processes
6. Rapidly integrate legacy tools
7. Trade off conceptual designs at the system level
8. Support multiple design representation and views
9. Capture a product’s behavior and its properties
10. Share objects over the Internet
11. Put constraints and rules on the objects
12. Capture design rationale
13. Capture requirement of a design
14. Have interoperability standards, such as PDM enablers and CORBA

Among these requirements, only the 5th point is about process management. But we can see that all the other points can be well organized around the process management. It can be process-guided as Figure 3.1 shows.

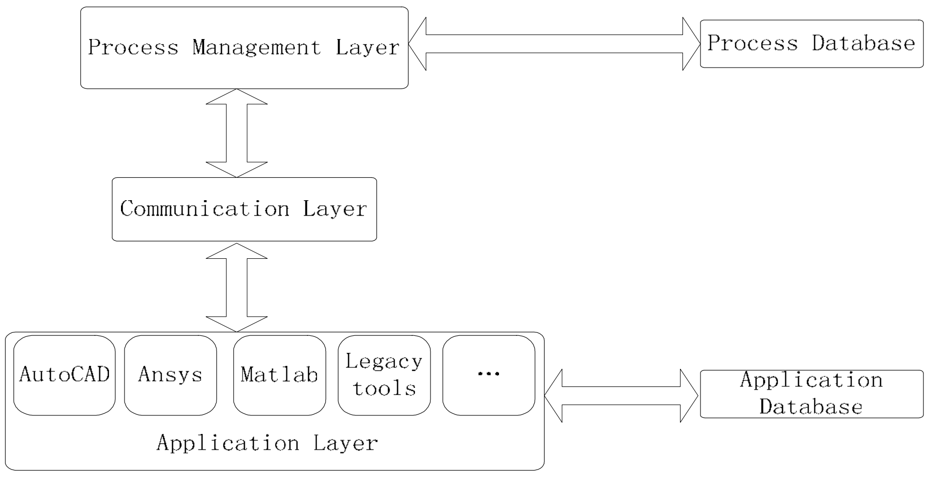


Figure 3. 1 Process-guided Distributed Collaborative Design

- Process management layer. This layer is at the top level. It will manage the whole design process, including the definition, control, simulation and monitor of the process. The process information will be stored in the process database.

- Communication layer. This layer is under the control of the process management layer. It includes the communication between process layer and application layer, as well as the communication happening at the application layer.

- Application layer. The layer will carry out the design tasks. It integrates different application software tools and human activities. These design tasks are to be organized by the process management layer. The information exchange is implemented through the communication layer. The generated design results will be stored in the application database.

Through effective support of process management, enterprise resources can be allocated to each process activity optimally.

## 3.2 System Architecture

The workflow-driven distributed collaborative design system framework can be constructed which uses the workflow as its backbone (see Figure 3.2). In such a framework, the workflow is: Event-driven, Constraint-triggered, and Decision-pushed, automated design process. The focused issues of this framework are: data sharing, work coordination, design quality control, accelerated design process and better understanding between participants.

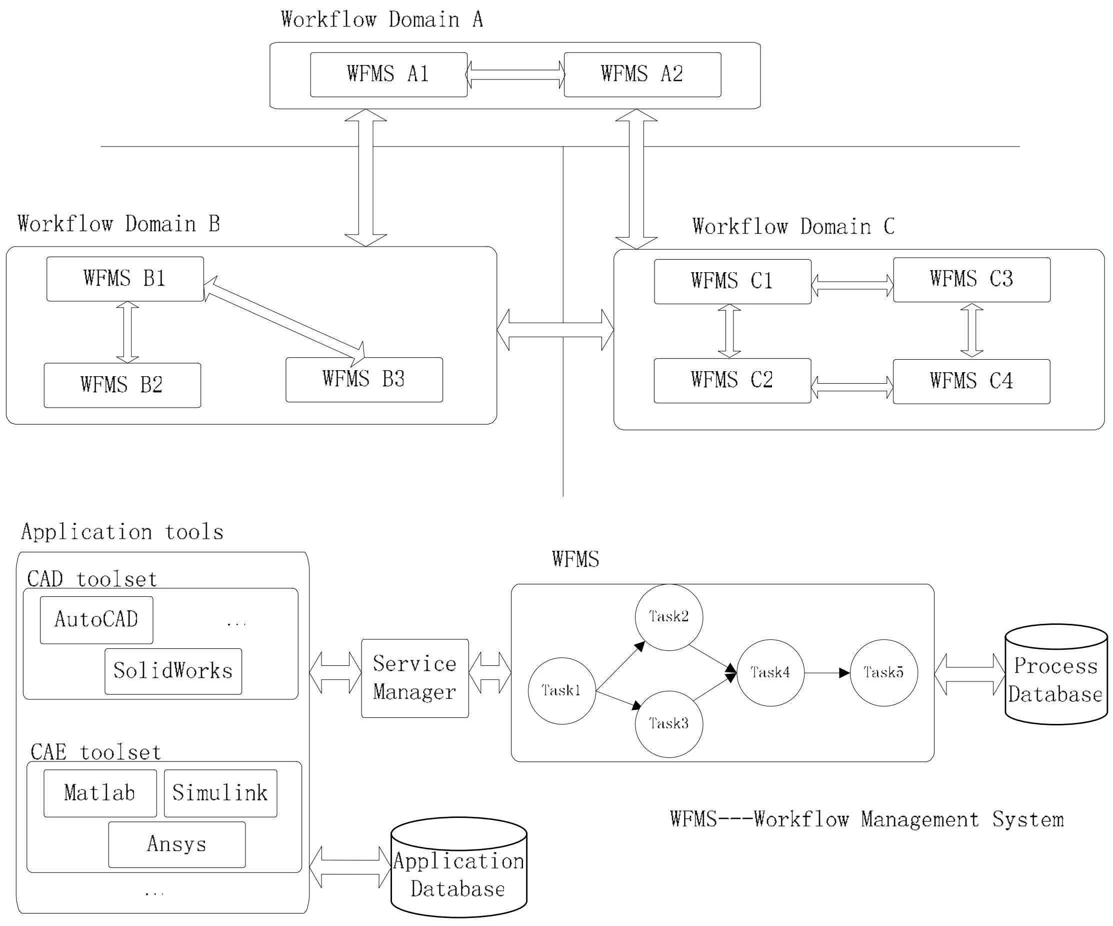


Figure 3. 2 Workflow-based Distributed Collaborative Design Framework

In the framework shown in Figure 3.2, each WFMS group residing in different domains has a representative workflow engine. Such a representative workflow engine has an interface to communicate with other representative workflow engine. Through it, the information exchange between different domains can be implemented.

In a workflow domain, the interactions happening between different WFMSs are realized by their workflow engines. Compared with the communication between different domains, the communication happening here is at a low level with the same meaning.

For a separate workflow system, the interactions between different tasks are carried out by their task engines, which will be discussed next. Such a workflow system is connected with a process database and a service manager. The process database is used to store the process information of the workflow, such as process states, process users, process resources and other log information. These data will be used to monitor and optimize the process performance. The service manager stores the information about services provided by the different engineers. It can be regarded as a service broker.  When a task requests a CAD/CAE service, it inquires the service manager. If the service exists, the request will be sent to the provider and the results will be returned to the task. The application database is used to store the data generated by the application tools.

A workflow engine is shown in Figure 3.3.

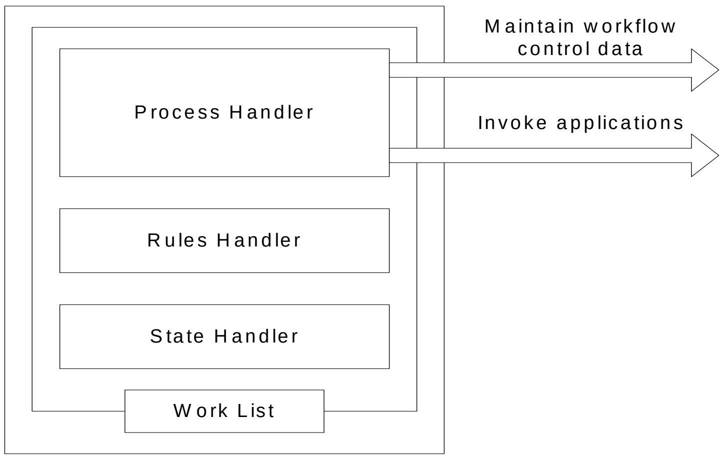


Figure 3. 3 Structure of Workflow Engine

In Figure 3.3, the process handler, rules handler and state handler are the three main parts of the workflow engine.

1. Process handler

   It provides flow control over business processes and practices of mission-critical applications.

2. Rule handler

   It enables users to enter, manage and easily change the business policies. It consists of 3 parts:

   - Rule Manager

     This is responsible for collecting the business rules that should be executed by the workflow engine on behalf of the consumer prior to transitioning to a new state in a given workflow activity.

   - Rule

     This one stores all information about a rule that is to be executed by the workflow engine prior to state transition.

- Rule Args

  This is used to hold additional data required by a business rule in order to perform rule validation

These three classes will be what a developer interacts with in order to create and manage custom business rules

3. State handler

When an event *e* is invoked on a workflow instance *I*, the following algorithm is executed:

- The current state $S_{current}$ is determined
- The transition *t* from $S_{current}$ to $S_{next}$ which has the event *e* is determined
- If *t* is not exactly defined, an exception is thrown
- All conditions of *t* are validated
- If all conditions are complied, the transition *t* fires
- All assignments of *t* are executed
- The workflow instance *I* is advanced to the state $S_{next}$

## 3.3 Workflow Meta-model

A meta-model is a precise definition of the constructs and rules needed to build specific models within a domain of interest [63].

Workflow meta-model is fundamental to the study of a workflow system. It is a representational language by which to express workflow models (to be described in detail in the following contents). From Figure 3.4, we can see that all the other models are derived from their meta-model. The meta-model determines how information is organized, stored and accessed in the system, how well data integrity can be maintained, and how easily the existing model can be extended to accommodate new needs.

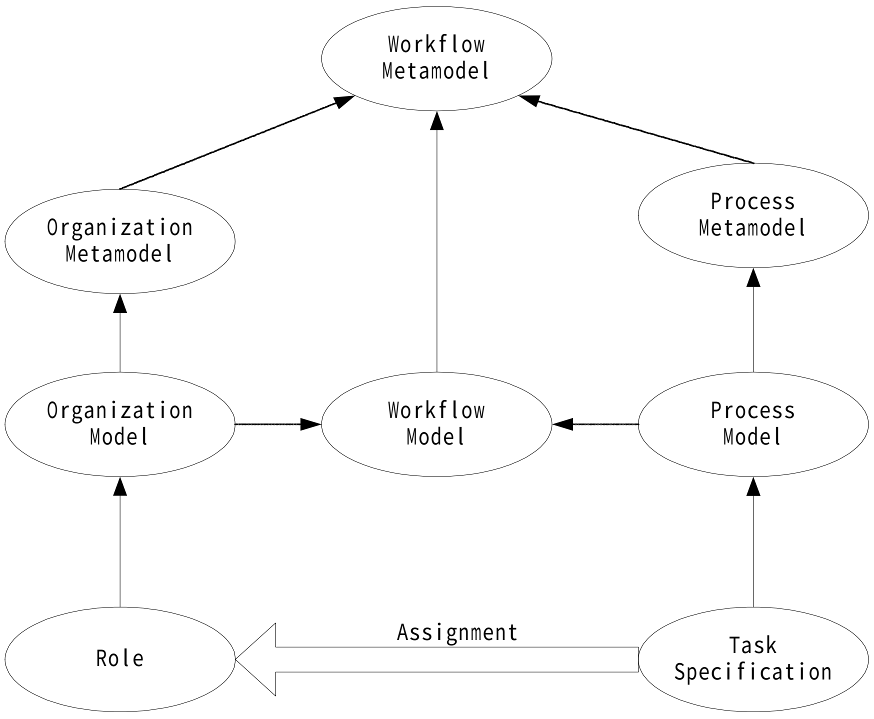


Figure 3. 4 Meta-model-centered Workflow Model Definition

In a workflow meta-model,

- A *task* is a definite piece of work.
- A *role* is a logical abstraction of one or more physical actors, usually in terms of functionality.
- A *process model* is an abstraction of processes. It emphasizes the coordination of tasks by highlighting their interdependence.
- An *organization model* is an abstraction of organizations.
- A *workflow model* combines a process model and an organizational model.

In this section, requirements for workflow meta-model are first discussed from two aspects: process meta-model and organization meta-model. And then the meta-model that can fulfill these requirements is proposed by using UML technology.

### 3.3.1 Process Meta-model

#### *Requirements*

According to the analysis of the development process, the use case diagram is shown in Figure 3.5.

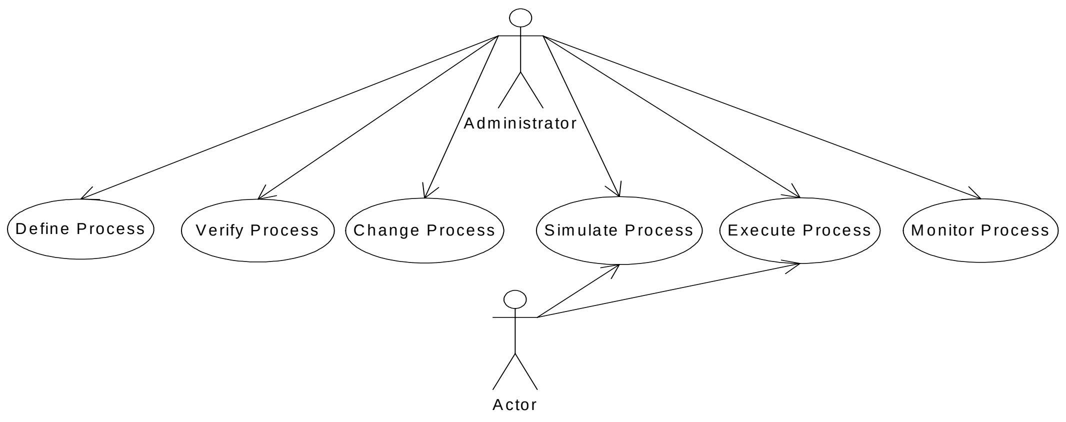


Figure 3. 5 Use Case Diagram of Operation

The basic scenarios that can be extracted from the use case diagram include:

- Administrator can define, verify, change, simulate, execute and monitor all processes

- Actor can only simulate and execute processes.

In Figure 3.5, the process is the center of all operations. The requirements for the workflow meta-model concerning the modeling of the process aspect are identified as follows:

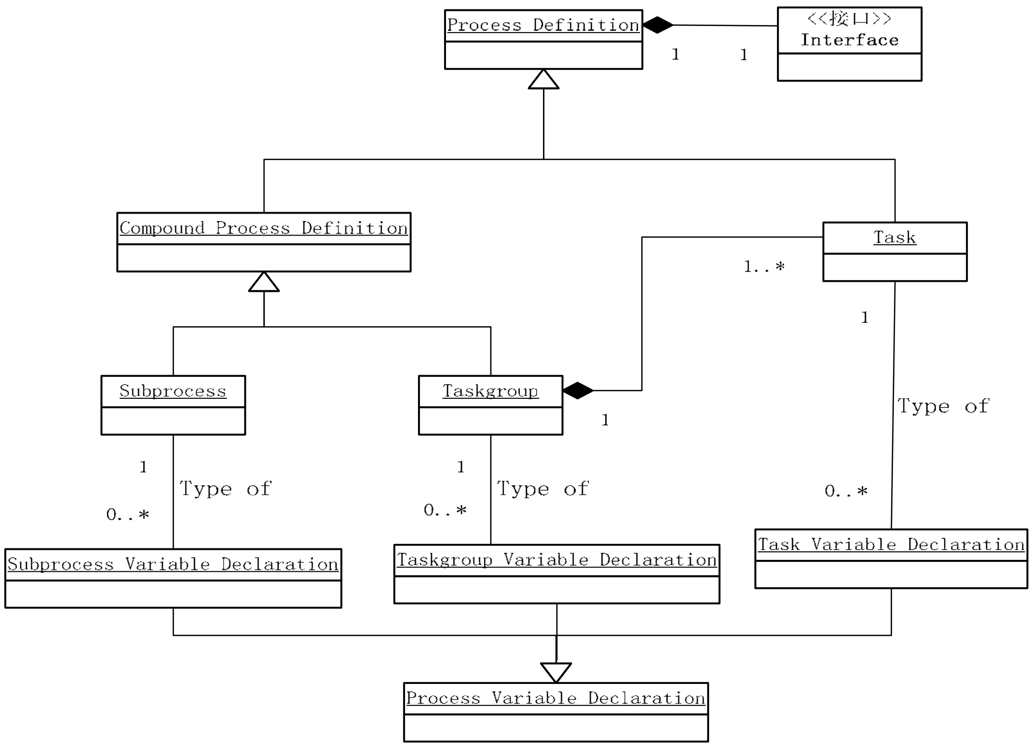
- Specification of atomic and composite processes. As a basic requirement, a process meta-model should support the specification of the basic tasks or services that the processing entities provide. Furthermore, a process meta-model should also support the specification of composite processes, which are aggregates of basic tasks.

- Composition of processes. Concerning the specification of composite processes, it is essential that a process meta-model supports the composition of new processes out of existing ones.

- Local structuring of process. A process can be very complex and may consist of a large number of tasks. Therefore, a process model should provide mechanisms that can bring tasks together into a group.

***Modeling Concepts***

Figure 3.6 illustrates the constructs of the process meta-model. The central part of the process meta-model is the process definition, which is abstracted as a textual and an interface. The interface defines the aspects that are visible to other workflow definitions and can be referred to by them. A process definition is either a task, representing a basic activity, or a compound process definition. And the compound process definition in turn can be either a sub-process or a task-group. A task-group structures parts of a compound process definition, without in turn being a full-fledged process.



Process Definition
<<接口>>
Interface
1
1
Compound Process Definition
Task
1..*
1
Subprocess
Taskgroup
1
Type of
1
Type of
1
Type of
0..*
1
0..*
0..*
Subprocess Variable Declaration
Taskgroup Variable Declaration
Task Variable Declaration
Process Variable Declaration

Figure 3. 6 Constructs of Process Meta-model

The detailed definition of concepts composing process is shown in Figure 3.7.

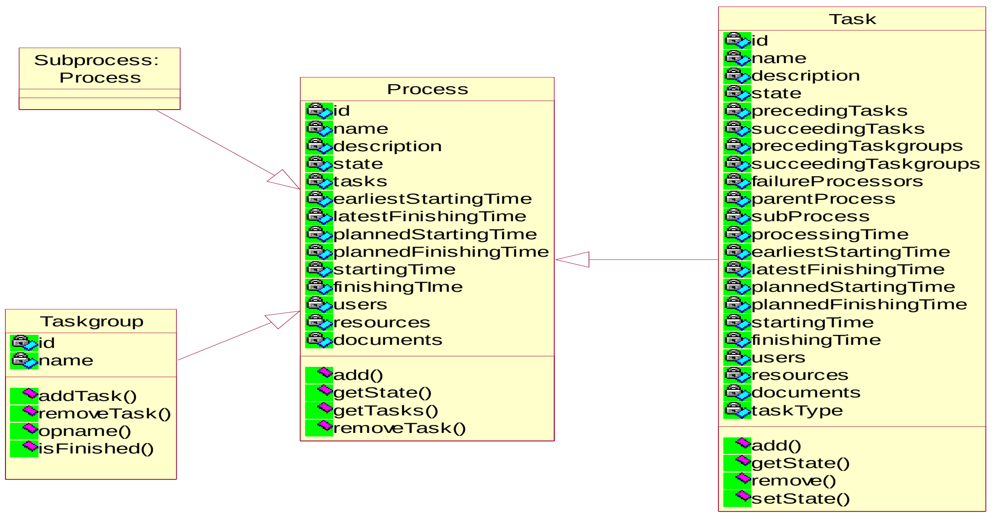


Figure 3. 7 Detailed Definition of Process Concepts

Among these classes, the task class is the one that is used to construct the other classes.

In the definition of process concepts, some attributes and operations are important for the implementation of dynamic changes of processes:

- Time attributes are very important. It can serve as a time stamp that can help to form a dynamic workflow, and is very useful when failure happens during the execution of workflow instance. Among these time attributes, finishing time can be used as a trigger to implement dynamic changes of workflow model.

- State. This attribute is used to indicate the state of the task or the process. The state of a task-group should be determined by the combination of tasks included in the task-group. The state can be one of these: not available, waiting, ready, working, suspended, aborted, obsolete and finished. The change of state can also trigger the dynamic change of tasks or processes.

- Add and remove operations. These operations can be called during build time or run time. When they are called by a workflow engine during the execution of processes, processes configurations are altered, which means that the dynamic changes of processes have happened.

### 3.3.2 Organization Meta-model

***Requirements***

Since certain processing entities may have overlapping or identical capabilities, there might be more than one processing entities that are able to execute a given activity.

Thus, for each activity that can potentially be executed by more than one processing entities, the question arises at runtime, as to which processing entity that the activity should actually be executed. A simple approach is to assign each activity to a specific processing entity prior to runtime. Such an approach, however, is very inflexible and has several drawbacks. For instance, in case the processing entity that an activity has been assigned to is unavailable at runtime, the activity will not be executed although another processing entity that could execute it might be available. In general, the assignment of activities to processing entities should be possible on the basis of:

- Individual properties of processing entities
- Relationships between processing entities and organization entities (such as other processing entities, groups or roles)

In order to support the specification of activity assignment based on the above aspects, the organization meta-model has to support the modeling of processing entities, the properties of processing entities, and relationships that exist among processing entities.

Most existing approaches either do not support organization modeling at all or use a fixed organization model. Many WFMSs provide a role concept for organization modeling, where a role contains a set of characteristics and capabilities of a set of processing entities such that each processing entity can play one or more roles. Here a new definition for role is proposed to avoid the case that activities are not directly associated with processing entities but with roles. It has the following features:

- It is generic, in the sense that it is not tightly bound to a specific process, but expresses general properties that can be used in different processes.

- Then, it is related to contexts. For example, any person in an organization that is authorized to do the same thing will play the same role, but in different phases, the same role may use different resources. So this means that the role may be imposed specific rules or some local capabilities by each phrase or each process.

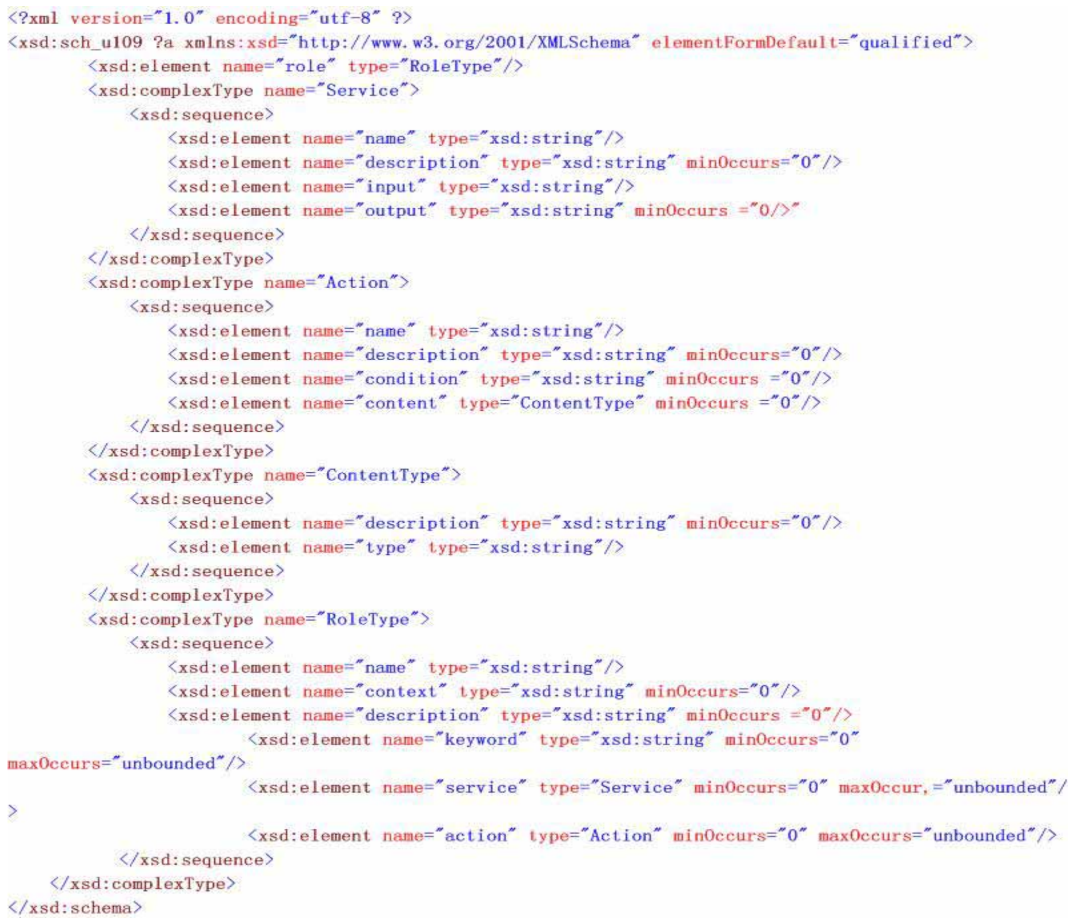
- Finally, the role should have a standard list of Input/Output parameters to provide services. In this way, the roles can be understood by one another. Otherwise, the interaction between them will be difficult.

The XML expression of role definition is shown in Figure 3.8. There are three main parts that have to be specified in the XML Schema of a role, following the model described above:

- The basic information. This part includes the information used to identify the role, and to specify a context for such role.

- The provided services. Each role providers are expected to provide some services. Each service is characterized by a name, an input list and an output list.

- The allowed actions. These actions are used to carry out a task related to a role. An action is characterized by a name and a description. Moreover, two elements are used to specify the execution condition of an action and the content of such an action.



```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:sch_u109 ?a xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xsd:element name="role" type="RoleType"/>
        <xsd:complexType name="Service">
            <xsd:sequence>
                <xsd:element name="name" type="xsd:string"/>
                <xsd:element name="description" type="xsd:string" minOccurs="0"/>
                <xsd:element name="input" type="xsd:string"/>
                <xsd:element name="output" type="xsd:string" minOccurs ="0/>"
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Action">
            <xsd:sequence>
                <xsd:element name="name" type="xsd:string"/>
                <xsd:element name="description" type="xsd:string" minOccurs="0"/>
                <xsd:element name="condition" type="xsd:string" minOccurs ="0"/>
                <xsd:element name="content" type="ContentType" minOccurs ="0"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="ContentType">
            <xsd:sequence>
                <xsd:element name="description" type="xsd:string" minOccurs="0"/>
                <xsd:element name="type" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="RoleType">
            <xsd:sequence>
                <xsd:element name="name" type="xsd:string"/>
                <xsd:element name="context" type="xsd:string" minOccurs="0"/>
                <xsd:element name="description" type="xsd:string" minOccurs ="0"/>
                        <xsd:element name="keyword" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element name="service" type="Service" minOccurs="0" maxOccur,="unbounded"/
>
                        <xsd:element name="action" type="Action" minOccurs="0" maxOccurs="unbounded"/>
           </xsd:sequence>
     </xsd:complexType>
</xsd:schema>
```

Figure 3. 8 The XML Schema for Roles Defined

***Modeling Concepts***

We view an organization as a set of organization entities (such as processing entities, groups and roles), and a set of organization relationships that exist among the roles to which organization entities belong.

As illustrated in Figure 3.9, an organization entity defines a name, a set of attributes and a role, where an attribute definition is a pair (attribute name, attribute type). An organization relationship specifies a name and associates two or more roles. Since the kinds of entities and relationships may differ considerably from organization to organization, the organization model is not fixed and the specification of user-defined organization entity and relationship is supported.

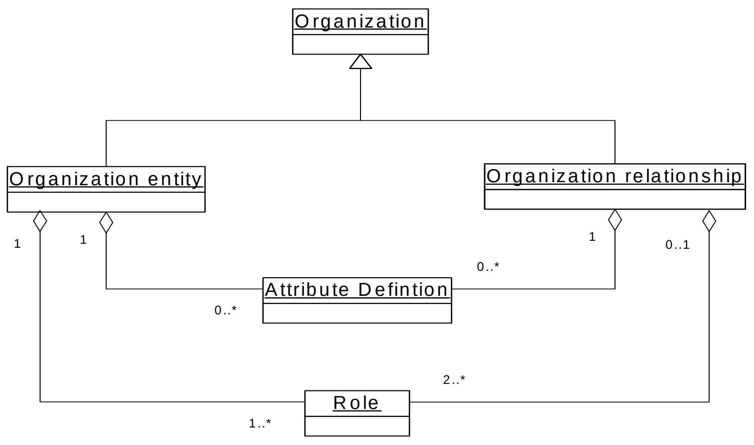


Figure 3. 9 Constructs of Organization Meta-model

The detailed definition of concepts for composing an organization is shown in Figure 3.10. Among the operations of each class definition, there are add and remove operations, which allow the instances of these classes to be changed dynamically.

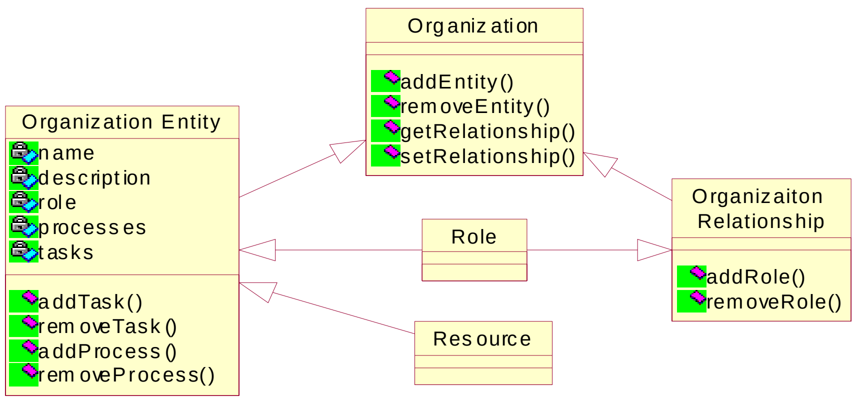


Figure 3. 10 Detailed Definition of Organization Concepts

## 3.4 Workflow-driven Communication

Design is a highly diverse social activity, especially for distributed collaborative design. Its success depends in part, on the effectiveness of communication channels between members of the design team. The diversity of activity together with ever challenging customer requirements leads to teams greater in size and complexity with higher levels of knowledge requirements. So it is necessary for the team members to understand each other very well through effective communication. Furthermore, the reliance on communication increases in distributed scenarios – problems including the inability to make decisions, misunderstanding, and poor quality of work can be attributed to poor or inefficient communication.

### 3.4.1 Communication in Design Process

Much of current communication happening during communication process is either synchronous or asynchronous. When synchronous communication is used, the maker of a request waits for the recipient to finish. This is the model of telephone calls and procedure calls. Synchronous communication has the advantage that the requestor has an easy time knowing what a response is about, but the disadvantage that the requestor has to wait for the answer (when it could be doing other things). In an asynchronous communication, a requestor sends off a request and then continues about its business. This is like a letter or event-based messaging system. Asynchronous communication has the advantage that the requestor can continue processing while the request is being handled, but the disadvantage that there is no simple way to match responses to the requestor's code.

In [24], the communication flow within a distributed collaborative design process in an industrial company was investigated. One major finding is that asynchronous communication tools are found to be significantly more prevalent than synchronous tools.

This phenomenon can be explained by the slight time difference and the difficulty inherent in the availability of dispersed colleagues. More significantly, it may be that the bulk of the subject of interaction is only supplementary to the ongoing design activity with only periods of core designing done distributively, especially in a synchronous mode. Such rules can also be found in other kinds of distributed collaborative design environments.

In addition, many of today's collaborative tools, such as video conferencing tools, are highly interactive and only support formal meetings. Although video conferencing is an important part of a collaborative environment, much of the collaboration requires more informal and asynchronous mechanisms. Especially when collaborations are worldwide and often extend beyond normal working hours, asynchronous communication becomes critical.

In the distributed collaborative design environment that is driven by workflow technology, in order to facilitate the asynchronous communication between participants, the following issues should be carefully considered in the WFMS:

- The communication model
- The communication protocol
- The communication framework
- The communication path

***Communication Model***

There are several types of communication models.

Direct Method Call

In remote procedure calls (RPC), distributed objects access one another through direct method calls. This was very common in early distributed systems: a central lookup service provides a client with the location, methods, and method parameters with which to execute an operation. An RPC schema is typically effective for small synchronous systems of objects [60].

Task-based Message

A second model uses task-based messages, whereby the code to complete a task is given from the client to a remote object. The remote object then executes the code inside the task, presumably more effectively due to its specialized resources. An example of such a system can be found in the Java-RMI tutorial.

Event-based Message

Event-based message is a communication model in which distributed objects access one another through events that are passed from suppliers to consumers. This is necessary for systems in which asynchronous communications are needed. Events may be used in a push model, in which they are broadcast by the supplier; or they may be used in a pull model, in which the consumer requests the events.

In general, there are three classes of events in such systems:

1. Internal events (e.g. events generated by activities like activity finished)
2. Clock events (generated by an external clock)
3. Administrative events (generated by a user, e.g. start process)

In a design environment where most of the communications are asynchronous, the event-based messaging model is adopted in this thesis.

***Communication Protocol***

As we know, design is a complex process with multiple activities. Different engineers perform different tasks in the design process, and data must flow among these engineers to ensure that the overall design is free from discrepancies. If the required software is not available locally, engineers must be able to find the software and operate it remotely. Moreover, in a dynamic product development environment, new engineers or those with different expertise may join the product design team, and they must be able to access the design process instantly, anytime and anywhere. Therefore the software framework must be web-based. HTTP is a most common communication protocol on the web. Because it is ubiquitous and widely supported, it is a reasonable option for building distributed systems.

***Communication framework***

To support multi-level communication, a hierarchical communication framework can be established, as shown in Figure 3.11.

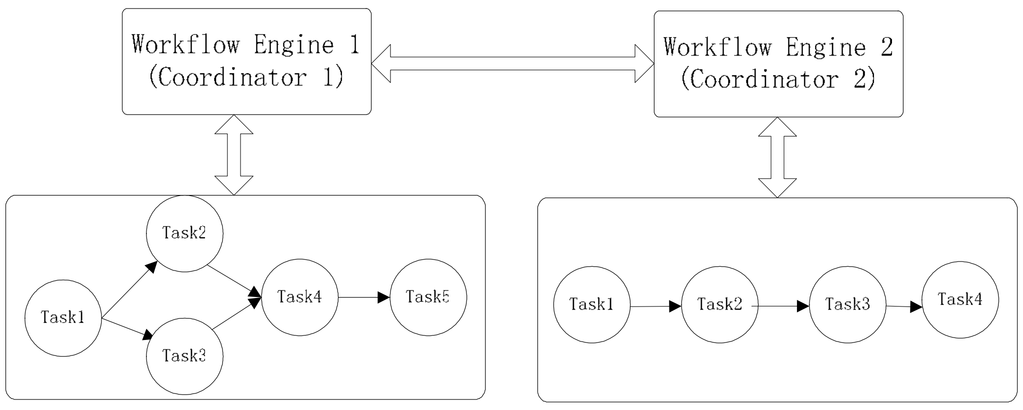


Figure 3. 11 Communication Framework

In this framework, interactions exist among tasks residing in each process, and also among workflow engines (representing different processes). The communication between workflow processes will be executed by workflow engine, while the communication between tasks is carried out by task engine by using the request/response mode.

***Communication Path***

Regarding the paths routed in the communication process, the backup and recovery of communication should be carefully considered. This is not the focus in this thesis and will not be discussed.

### 3.4.2 State-oriented Communication

Although there is much development in workflow technology, there are not many research efforts focusing on the aforementioned asynchronous communications. Furthermore, most of the existing communication mechanisms are at the process level rather than the task level. These mainly concerned the communication and connection between different WFMSs that reside in different domains while the means to implement asynchronous communication between tasks are not studied deeply. Here a state-oriented communication mechanism is proposed to solve this problem. The basic concept of this mechanism is that the communication between tasks is determined and triggered by the states of the tasks. The workflow process is executed through the change of each task’s state. These changes happen as a result of the communication between tasks.

But firstly, we introduce the state changes that can happen to workflow instances and workflow tasks.

### *Workflow Instance State Change*

During each phase of its enactment, a workflow instance maintains a well-defined state. This state may be modified through the workflow engine or through an external entity interacting with the workflow engine. Figure 3.12 shows the state model of a workflow instance in the form of a UML state diagram. The model is composed of the two super-states “doing” and “done”. A workflow instance in the state “doing” can be manipulated through the workflow engine or an external entity, while a workflow instance in the state “done” is finished and cannot be reactivated.

A “doing” workflow instance can be finished either by completion, or by forced termination. In the latter case, the resulting state can be aborted, if running activities at times of the cancellation command were allowed to complete. If the cancellation of the workflow instance leads to the immediate cancellation of running activity instances, the resulting state is terminated (also known as forced abort).

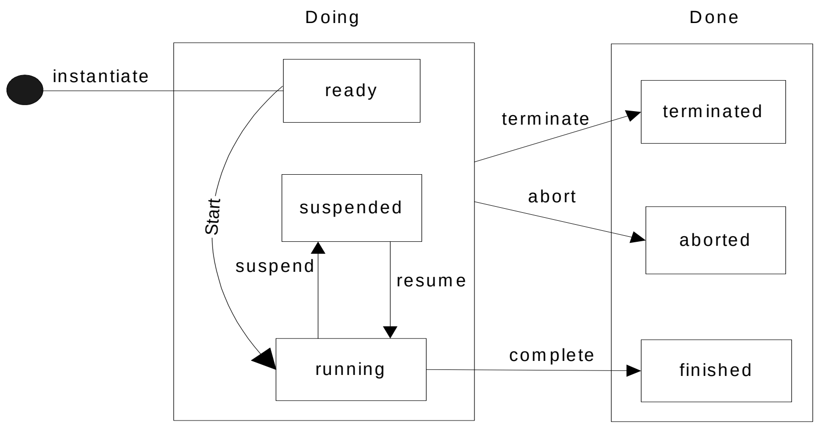


Figure 3. 12 Workflow Instance State Change Diagram

***Workflow Task State Change***

Similar to workflow instances, each task instance follows a lifecycle that can be described using a state diagram. Figure 3.13 shows the state-change-diagram for a typical task instance. Like the state-change-diagram of the workflow instance, this model consists of several nested states. A task instance is either in the state "doing", and can be started, suspended etc., or it is in the state "done", if it has been completed or aborted. Upon activation by the workflow engine, a task instance is in the state ready, indicating that it is ready for execution.

If the task instance is assigned to an arbitrary number of workflow participants (i.e., via a role or an organizational entity such as a department), the task instance will be visible to all authorized participants through a work item on their shared work list. Once a participant selects the work item representing the task instance for further processing, the task instance changes into the running state. A running task may be suspended and resumed an arbitrary number of times.

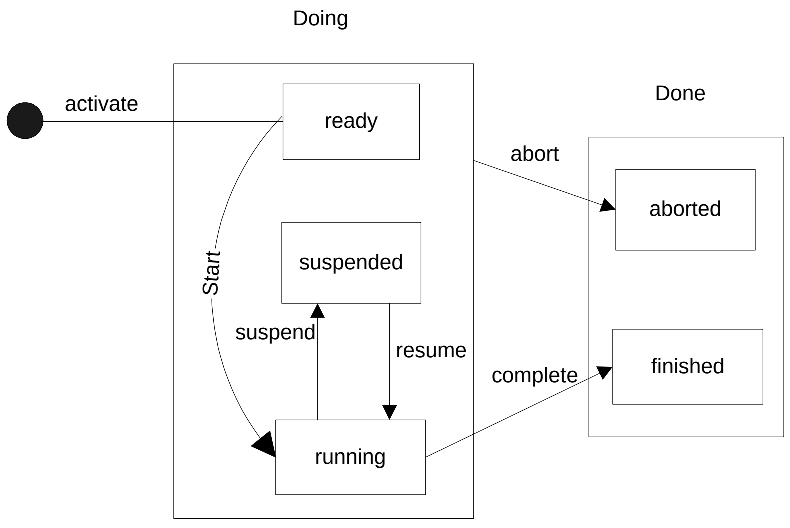


Figure 3. 13 Workflow Task State Change Diagram

Different communications take place according to the state of tasks. Therefore a state-event table (see Table 3.1) that includes pairs of state and event is needed. Compared with the workflow engine that is in charge of the whole workflow, here we use a task engine which is embedded in a task to maintain such a table and manage the behavior of each task.

Table 3. 1 State_Event Table

| State | Event |
|---|---|
| Not Available | Send_NAInfo() |
| | On_NA() |
| Ready | Send_ReadyInfo() |
| | On_Ready() |
| Suspended | Send_SuspendedInfo() |
| | On_Suspended() |
| Running | Send_RunningInfo() |
| | On_Running() |
| Aborted | Send_AbortedInfo() |
| | On_Aborted() |
| Finished | Send_FinishedInfo() |
| | On_Finished() |

Here Send_XXInfo() means that the task engine sends its state information to the workflow engine after every defined period for the purpose of external control. On_XX() event happens when the task reaches a certain state, and it will be explained and executed by the task itself.

In Table 3.1, according to the available states of each task, sub-process and process, more events can be defined. These events will be used to trigger the target workflow tasks. Users can decide on the contents of these events and sequentially control the execution of the workflow process. By these user-defined events, continuous and asynchronous communications can be implemented.

In this manner, the workflow engine just needs to manage the states of each task. Each task is an autonomous processing cell. So the workload of the workflow engine can be greatly lightened, and the network traffic will also not be heavy. The relationship between the workflow engine and the task engine is shown in Figure 3.14.

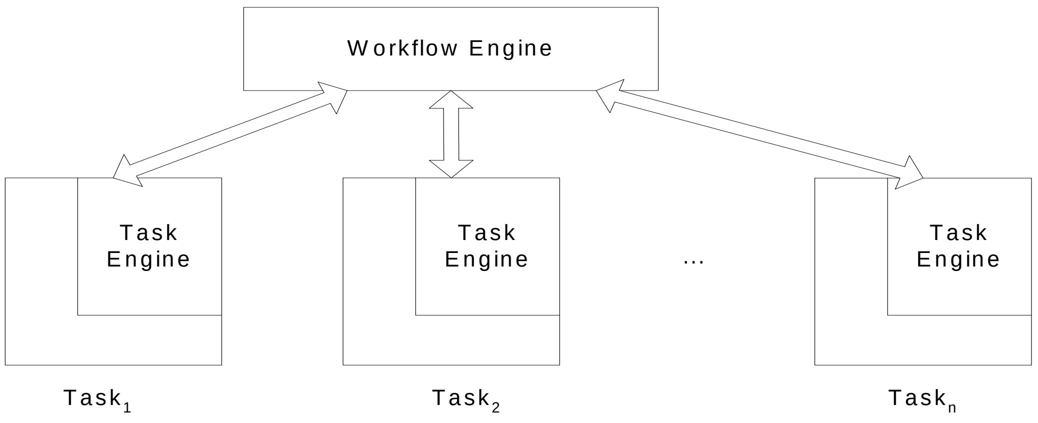


Figure 3. 14 Relationship between Workflow Engine and Task Engine

The workflow administrator can control the whole workflow process by managing the workflow engine. When a task's behavior needs to be modified manually, the administrator can change its state stored in the workflow engine. Then the state-change message will be transferred to the task and change the task's action. In this way, the administrator can control the process at a higher level. It will be more convenient and easier.

## 3.5 Related Issues

### 3.5.1 Service Registration

To integrate the CAD/CAE tools provided by the dispersed users, the framework supports the registration of these services.

Engineers who wish to provide services over the Internet should fill the registration form which will be stored by the service manager. The engineer must define the service name, and program script to be executed by remote request, and input operator name, service type, and description of the service. The registration form is shown in Table 3.2.

Table 3. 2 Service Registration Form

| Service Element | Value |
|---|---|
| Service Name | CAX |
| Server Information | IP address |
| ClassPath | Path of exexcution file |
| Script | Macro instruction |
| Operator | User |
| Service Type | Automatic/Manual |
| Description | |

In automatic service type, the application tool executes the user-defined script with a remote method invocation call. Otherwise, it prompts a notifying dialog box to inform the engineer that a request has arrived from another engineer.

The services provided can be changed, e.g., engineer can just drop the service from service table, or register a new service. So both external and internal services can be integrated into the system dynamically, contributing to the scalability of this framework. By using the service registration mechanism, this framework can be more extensible and more powerful.

### 3.5.2 Design Session Management

In order to carry out distributed collaborative design effectively, network-based sessions are established in distributed product design environment to support reliable collaboration between geographically dispersed engineering teams. In a collaborative session, different engineers can share the common data and communicate with each other through conferencing tools, such as email, instant messaging tools, etc.

Traditionally, a session refers to the process that a group of users connect from various locations to work together on shared data or use conferencing tools to communicate ideas. However, collaborative product design activities include asynchronous activities as well as synchronous activities. Dependencies exist between sequential activities, that is, even if one may work as an individual to perform an asynchronous collaboration activity, he still works on shared resources which may affect other collaboration activities. In addition, the functions of traditional network-based sessions have to be extended in order to facilitate both design and analysis activities, such as providing co-modeling, visualization of meshing result and engineering data. Thus, the definition of session for collaborative product design may be extended as: The process in which multi-disciplinary designers, who may be from geographically dispersed locations, work together to design a product or analyze engineering results, synchronously or asynchronously, with the help of collaboration tools.

Here the session can be classified into two types: short session and long session. A short session means the session including just one activity, such as defining product functions. While a long session means the session that includes two or more activities, such as design of product geometry model which needs CAD design, CAE analysis and simulation.

In this way, a session can be represented by an activity or a sub-process (or process, if this process includes just one session), and the workflow of design is then composed by sessions (see Figure 3.15). The session management can also be the management of an activity or a process. When a design workflow model is instantiated, a process, including corresponding activities, is created. Then as a workflow node is activated, the session responsible for executing this node is invoked automatically. Thus dispersed designers can join this session to complete the activity collaboratively. If it is a short session, the task engine can handle the events and communication happening in the session. If this activity includes a long session, the communication between task engines is needed and the workflow engine will control these interactions.

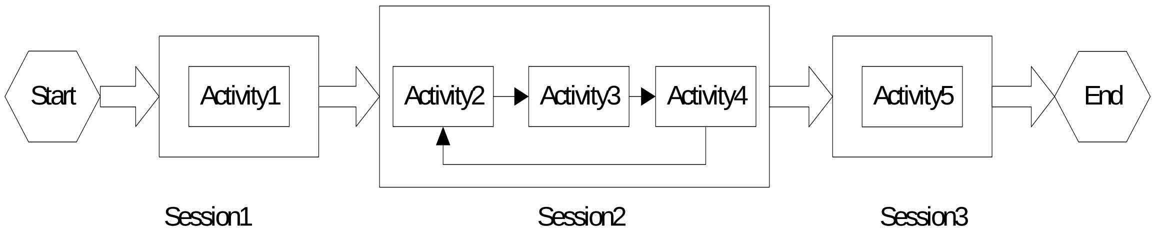


Figure 3. 15 Process Composed by Sessions

During the execution of a session, two kinds of concurrency happen which should be considered carefully. One kind is the concurrency taking place in the session. This issue has been discussed in [59] which incorporates a centralized coordination mechanism (CCM) (see Figure 3.16) and a synchronization scheme to solve this problem.

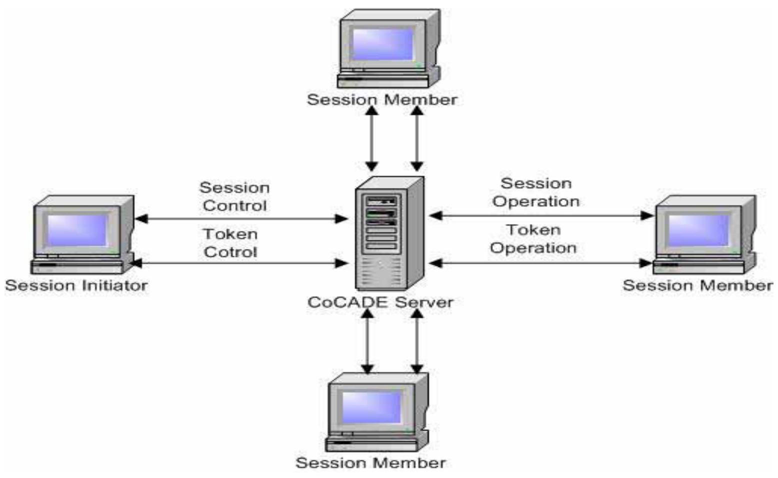


Figure 3. 16 Centralized Coordination Mechanism (CCM)

Another kind of concurrency is taking place between several sessions. As a workflow model can be used by multiple workflow instances, which leads to call for several sessions of the same type. It is possible that these sessions invoke an application tool and other resources, or write/read same data at the same time. Thus a mechanism should be used to coordinate these synchronous operations. The process handler embedded in the workflow engine can handle it based on a first-come-first-serve protocol. For example, when session A calls a CAD tool, the process handler registers this session firstly, and then invokes the CAD tool for session A. As session B needs to use this CAD tool, the process handler checks its register list and let session B wait until session A ends the usage of this CAD tool. By this means, all the sessions can be served.

## 3.6 Summary

In this chapter, requirements are analyzed. Then a workflow-based design framework is proposed. This framework takes full advantage of workflow technology and is extensible and flexible. To implement this framework, workflow meta-model and state-oriented communication are discussed in detail. Then the related issues: service registration and session management are also presented. This framework can effectively support distributed collaborative design and deal with the problems that possibly happen in engineering design process. In the next chapter, the dynamic feature of the design process is to be studied in detail.

# Chapter 4. Dynamic Workflow Change Management

Typically, the design process is constantly changing, such as new customer requirements have to be met, business processes are reengineered, or new legal requirements to change the way the design is carried out. All these factors can greatly affect the execution of the design process. These dynamic changes can be more frequent in a distributed and collaborative environment, especially for the technology-intensive product development. Current workflow management systems are capable of handling static business processes. However, dynamic workflow change has not been addressed by most workflow management systems. The inability to support dynamic workflow change compromises the application of workflow system in certain scenarios.

Therefore, to fully facilitate a design process, the proposed framework must be able to fulfill dynamic change requirements. In this chapter, an approach is proposed to facilitate efficient management of dynamic workflow change by minimizing unnecessary or repeat execution of tasks after the change takes effect. In addition, limitation of the approach is discussed.

## 4.1 Introduction

As a tool to model, execute and control business processes, a workflow management system can computerize business processes as workflows, store them in a database, and carry them out by a workflow engine.

From the perspective of organizations adopting workflow solutions to manage business processes, these business processes should be as static as possible since:

(1) A business process is typically executed in many cases where its static nature makes these related cases foreseeable;

(2) Dynamic behaviors of business processes make involved elements, including human, equipment, and information, difficult to handle since these elements are initially designed to cope with the specified process.

Therefore, organizations take very cautious measures to define their business processes and seldom change them. In accordance with the abovementioned fact, most workflow management solutions are designed to cope with business processes in static manner, e.g., a workflow template for a business process is defined and executed till its completion. There is not much functionality provided by these solutions to handle dynamic behaviors, such as changes of the running workflow on-the-fly.

Unfortunately, process change does exist in real business environment due to two reasons:

(1) At design time, the specification of the workflow is not complete due to lack of knowledge and at run time errors may happen;

(2) While during the execution of the workflow instances, changes occur causing various problems, such as breakdowns, reduced quality of services, and inconsistencies.

Therefore, the system should be ready to handle these undesirable events related to the dynamic aspects of workflows.

Once process changes occur, new workflow templates are defined for all new process cases. However, it is critical to handle existing cases based on old workflow templates. Basically, there are 4 possible policies to follow:

(1) *Forward recovery*. These old cases are aborted and handled outside of the workflow management system;

(2) *Backward recovery*. These old cases are stopped and restarted according to the new workflow template;

(3) *Proceed*. These old cases proceed as if the change has not occurred. New cases are executed based on the new template;

(4) *Transfer*. These old cases are transferred to the new workflow template and executed.

Most workflow management systems are able to implement the first three policies in various degrees. The example illustrated in Table 4.1 shows how forward recovery, backward recovery and proceed policies are realized in a PDM system, SmarTeam.

However, the forth policy, transfer, is not effectively supported in SmarTeam. Before applying the new workflow template, the old instance has to be stopped, followed by a restart. Restarting a workflow may lose some key runtime information since the state of the workflow instance is refreshed to the original value. More importantly, some completed tasks have to be carried out unnecessarily after restarting the workflow instance. If the affected workflow instance is complex and involves a number of external collaborators, substantial business cost will be incurred.

Table 4. 1 Change Management in SmarTeam

| | |
|---|---|
| Forward recovery | The administrator stops the workflow and handles the change manually. |
| Backward recovery | The administrator stops/deletes the associated process, defines the new workflow template, associates the process with the new template, and initiate the process. If the stopped process on the old template causes anything that needs to be cleared, the administrator does it manually. |
| Proceed | The administrator does not do anything to the associated process. |

Therefore, dynamic workflow change management comes in as a potential solution. A dynamic change can happen on a single workflow instance or a set of instances under their common workflow template. A workflow management system, if it supports dynamic workflow change, can either modify the affected instance without restarting it, or re-initiate it from the new workflow template while minimizing the work to be performed by affected users. The first method is instance-based while the second is template-based (schema evolution).

In this thesis, an approach is proposed to address template-based dynamic workflow change instead of instance-based change. This is based on an industrial practice that a new workflow instance is initiated from its template, instead of another workflow instance. The approach is implemented in a popular PDM system SmarTeam to demonstrate its feasibility.

This chapter is organized as below: firstly related works on dynamic workflow change are introduced; then the proposed approach is presented; the following part gives implementation details and examples; and finally the summary of the work.

## 4.2 Related works

Existing works on workflow can be classified into the following categories: (1) modeling technologies, (2) analysis and verification, (3) design and implementation, and (4) workflow change.

In these four topics, dynamic workflow change still remains an unsolved problem. Casati et al. [50] proposed a method to facilitate change of workflow schemas by applying a complete, minimal and consistent set of modification primitives. His analysis was performed at theoretical level and did not provide implementation details. Van der Aalst adopted the concept of workflow inheritance to handle dynamic workflow change [51, 52]. A limitation of his approach is that changes only happen to workflows with inheritance relationship. Shingo defined the cost of dynamic workflow change as change times to evaluate the performance of basic workflow changes [53]. Ellis presented Petri-Net based approach to handle dynamic changes in workflow systems [54]. He also reported ML-DEWS as a modeling language to support dynamic workflow changes [55]. Reichert et al. developed a method, ADEPT-flex, to facilitate dynamic changes of workflow [56, 57]. In his work, three kinds of dynamic changes were discussed, namely, insertion of tasks, deletion of tasks, and change of task sequences.

Most of existing researches on dynamic workflow change focus on theoretical aspects. Due to lack of research addressing implementation issues, present commercial solutions are not capable to manage dynamic workflow change. Therefore, it is significant to investigate how to make the dynamic workflow change management happen in actual environments.

## 4.3 Problem Statement

As a computerized business process, a workflow can be modeled by a number of formal ways. In this thesis, however, a simple representation is adopted for simplified algorithm and straightforward implementation. With respect to simple representation, a workflow $W$ can be denoted as a set of nodes:

$$W = N = \{n\} \tag{1}$$

Here a node $n$ stands for a tuple $(\{t\},\{u\},\{c\})$, with $t$ denoting a task, $u$ denoting a user, and $c$ denoting a connector of the specified node.

Each connector $c$ is also denoted as a tuple $c = (r, n_s, n_e)$ where $r$ is the response of $c$, $n_s$ is the node from which the connector starts, and $n_e$ is the node to which $c$ ends.

During the execution of a workflow, the users of an active workflow node perform tasks specified in the node and issue responses to all following nodes based on the completion of these tasks. If all tasks are completed successfully, the response is positive, normally indicated by ‘accept’. If some problems are encountered, the users issue negative response, normally indicated by ‘reject’, to notify other users, pre-defined by the workflow, to re-perform their tasks.

Note that the above description is to a large degree simplified to focus on control logic of workflow. In a complete workflow model, however, there are other essential properties mentioned in chapter 3, including (1) document, (2) resource, and (3) event logic. Document is the information attached to specific nodes or passed between two nodes. In most cases, the output information from a preceding node is the input information to a succeeding node. Resource refers to equipments or facilities occupied by a node when the node is being executed by its users. Event logic is a set of pre-defined actions triggered by events associated with a node. Normally, event logic is implemented as scripts that are associated with specific nodes and managed by workflow management system.

Here the primary object is to develop an approach to manage dynamic workflow change from the perspective of workflow control logic; so document, resource and event logic are not considered.

In actual manufacturing environment, once a workflow instance gets initiated, it cannot be changed without being stopped first. At the moment of stopping, the workflow instance can be represented as below:

$$W = N_f \cup N_a \cup N_u \tag{2}$$

Here, $N_f$ denotes all finished nodes, $N_a$ denotes all active nodes, or nodes being executed, and $N_u$ denotes all unreached nodes. $N_a$ actually defines a `front' of the current workflow.

For each node $n \in N_f$ , a specific amount of business cost is rendered for the execution of all tasks of $n$ by its users. The cost may include components like time spent and resources consumed. Therefore, the total cost incurred at the specific moment, namely the sum of cost for each finished node, can be represented as following:

$$S = \sum_{n \in N_f} s(n) \tag{3}$$

Once a workflow instance is subject to a dynamic change, it has to be shut down, attached with a new workflow template, and re-started. Under the new template, the workflow instance can be denoted as below:

$$\overset{\circ}{W} = \overset{\circ}{N}_f \bigcup \overset{\circ}{N}_a \bigcup \overset{\circ}{N}_u \tag{4}$$

Here $\overset{\circ}{N}_f = \varnothing$. However, in case that the new workflow instance $\overset{\circ}{W}$ has some nodes which have been executed in $W$, e.g., $N_f \bigcap \overset{\circ}{W} \neq \varnothing$, there is the possibility that some tasks need not to be re-executed. Therefore, the problem can be stated as identification of nodes $\ddot{N}$ that satisfy three conditions:

(1) These nodes exist in both old and new workflow instances;

(2) They have been executed in the old workflow instance;

(3) The results associated with these nodes in the old workflow instance can be reused in the new workflow instance without having their users carry out their tasks again.

In this way, the cost of re-executing the new workflow can be reduced to

$$S = \sum_{n \in \overset{\circ}{W}} s(n) - \sum_{n \in \ddot{N}} s(n) \tag{5}$$

Obviously, to achieve maximum business benefit upon the dynamic change of workflow, as many nodes in $\ddot{N}$ should be identified as possible.

Another indicator for efficiency of dynamic workflow change management is defined as below:

$$e_f = \frac{\left|\overset{\circ}{N}_f\right|}{\left|\overset{\circ}{W}\right|} \tag{6}$$

## 4.4 Mechanism

Given workflow template $W$ and $\overset{\circ}{W}$, for a node $n \in N_f \bigcap \overset{\circ}{W}$, there are two possibilities:

(1) It can be bypassed;

(2) It has to be re-executed.

A bypass-able node $n$ can be denoted as $\overset{\rightarrow}{n}$. To determine if $n$ can be bypassed, the following fact is introduced.

**Fact 1** *A node $n$ is by-passable iff the following two conditions are satisfied:*

*(1) All preceding nodes of $n$ are by-passable;*

*(2) And $n$ has been executed in the old workflow instance $W$.*

Here, a preceding node of $n$ sits between $n$ and $n_0$, the start node of the workflow. The preceding node of $n$ can be denoted as $n^{'} : n^{'} \rightarrow n$. If $n^{'}$ and $n$ are directly connected, $n^{'}$ is an immediate preceding node of $n$ and denoted as $n^{'} : n^{'} \overset{\bullet}{\rightarrow} n$.

Therefore, the algorithm for identifying all by-passable nodes in $\overset{\circ}{W}$ is straightforward as below:

In the above algorithm, a set of by-passable node in the new workflow instance

**Algorithm 1**. Identify by-passable nodes in new workflow $\overset{\circ}{W}$

```
    Obtain $N_f \subset W$
    $N_f \leftarrow \varnothing$ {Initially there is no by-passable node}
    Set $\overset{\circ}{n_0}$ as by-passable and add it into $\overset{\circ}{N_f}$
    Set node pool $\overset{\circ}{N_p} \leftarrow \overset{\circ}{W} - \{\overset{\circ}{n_0}\}$
    Set flag $f \leftarrow true$
    **while** $\overset{\circ}{N_p} \neq \varnothing$ and $f = true$ **do**
      $f \leftarrow false$
      **for all** $n \in \overset{\circ}{N_p}$ **do**
        **if** $n \in N_f$ **then**
```

```
          Build preceding node set $\{n' : n' \to n\}$ for node $n$
          **if** $\forall n' \in \{n'\} : n' \in \overset{\circ}{N}_f$ **then**
            Remove $n$ from $\overset{\circ}{N}_p$ to $\overset{\circ}{N}_f$
            Set $n$ as by-passable
            $f \leftarrow true$
          **end if**
        **end if**
      **end for**
    **end while**
```

$\overset{\circ}{W}$ is returned. The workflow engine can set the state of these nodes to 'finished' and users associated with these nodes need not re-perform their tasks in the new workflow instance.

The algorithm sets up a node pool and scans all nodes inside the pool repeatedly to identify and remove out by-passable nodes. In the worst case, the algorithm has to scan the pool $N-1$ times while the size of the pool starts with $N-1$ and ends with $0$, where $N$ refers to the number of nodes in the workflow instance. Therefore, the algorithm has a $O(n^2)$ complexity, which is not computationally optimal.

In general, the number of nodes in a workflow is less than 100; so algorithm complexity is not a major issue. However, in certain cases, more efficient searching approach is preferred and the connecting information kept in each node can be exploited to expedite identification of all by-passable nodes. An alternate algorithm is proposed as below.

The complexity of the second algorithm depends on the topology of the processed workflow. In the best case where the workflow has a linear structure, the algorithm can finish with $O(n)$ complexity. In the worst case where the workflow has a balance-tree-like topology, the complexity goes up to $O(n \lg n)$.

Based on the above approaches, the change of a workflow template can be reflected into all instances under the template and the cost for users to carry out their finished tasks in the new workflow has been reduced as suggested in equation 5.

**Algorithm 2**. Identify by-passable nodes in new workflow $\overset{\circ}{W}$

**Require**: a node $n$ as the parameter

**Require**: a global flag $f$ with initial value $0$

```
if $f = 0$ then
    Set $\overset{\circ}{N}_f \leftarrow \varnothing$
    Set current node $n \leftarrow \overset{\circ}{n}_0$
    $f \leftarrow 1$
end if
```

```
if $n \in N_f$ and $\forall n' \overset{\bullet}{\rightarrow} n : \overset{\rightarrow}{n'}$ then
  Add $n$ into $\overset{\circ}{N}_f$
  Build the following node set $\overset{\circ}{N}_n$ of $n$
  for all $n \in \overset{\circ}{N}_n$ do
    Set $n$ as the parameter and fork a thread of this algorithm
  end for
end if
```

Correctness of state of all nodes in the new workflow instance after the dynamic change takes effect is guaranteed since the algorithm will not bypass a node unless all of its preceding nodes get bypassed.

## 4.5 Exception handling

The general scenario of for a template-based workflow change, or schema evolution, is to apply the new workflow template on all running workflow instances under the original workflow template. In a workflow system, the transfer proceeds in the following steps:

(1) Create the new template $T'$ based on old one $T$

(2) Find all instances $\{W : W \lhd T\}$ initiated from $T$

(3) Apply $T'$ to all instances of $\{W\}$

In certain cases, it is necessary just to transfer part of $\{W\}$ into the new template. Therefore, the above process should be changed as below:

(1) Create the new template $T^{'}$ based on old one $T$

(2) Find all instances $\{W : W \lhd T\}$ initiated from $T$

(3) Identify all instances to undergo transfer $\{W^{'} : W^{'} \in \{W\}\}$

(4) Apply $T^{'}$ to all instances of $\{W\}$

In above approach, step (3) can be operated manually by an administrator.

However, a mechanism is preferred to enable automatic identification of instances for transfer.

## 4.6 Implementation

The algorithm is implemented as a program running at the server side of this workflow management system. When it runs, the following steps are executed:

(1) Start the workflow engine and create necessary data sets;

(2) Obtain a flow process instance by its ID;

(3) Obtain the current workflow template of flow process and the new workflow template;

(4) Save the current workflow in memory and attach the new workflow template to the flow process;

(5) Apply the algorithm to determine bypass-able nodes in the new workflow instance;

(6) Initiate the new workflow instance and execute by-passable nodes one by one at back end;

(7) Release the modified flow process for user access.

To execute a work node in a program without user intervention, the program needs to log on to the system as a regular user specified in the work node. To execute all by-passable nodes, the program should keep a list of user login information (user name and password) for all these nodes. However, this approach is not secure since sensitive information such as password should be stored only in the central user database of the system, instead of other places.

Further more, frequent login actions via different users compromise the performance of the program and the system. In addition, the system has no knowledge about the context of execution of a specific node: whether it is executed by a user normally, or by the managing program automatically.

To facilitate automatic execution of by-passable nodes, a new user, Dynamic Workflow Manager, is created and added into each node of all workflow templates as an executor. Hence, the dynamic workflow program can execute bypass-able nodes under the user ‘Dynamic Workflow Manager’ without keeping logging in and out.

When implementing the proposed algorithm, an essential step is to determine if a node $n$ of the new workflow template can also be found in the old workflow template. Virtually, this process compares the node $n$ with each node $n'$ of the old workflow template. The comparison is property-wise since it checks each key property of node

$n$ in node $n'$. If $n$ and $n'$ have same values for a set of common key properties, these two nodes can be deemed identical.

In the proposed workflow environment, only the name and tasks are checked as key properties since they directly indicate the nature of their work node. The comparison procedure is described as below:

```
**Procedure** NodeComparison($n$, $n'$)
  $r = 1$ {return value with 0 indicating different and 1 identical}
  **if** $n$ and $n'$ have different names **then**
    $r = 0$
  **else if** $n$ and $n'$ have different number of tasks then
    $r = 0$
  **else**
    **for all** task $t$ of $n$ do
      **if** $t$ cannot be found in $n'$ **then**
        $r = 0$
      **end if**
    **end for**
  **end if**
```

To demonstrate the application of the developed program, one example is given to simulate the scenarios of real dynamic workflow change.

Company ABC runs business in multiple regions. In each region, a branch is established to handle marketing and product support. To consistently manage operations of all its branches, the headquarter (HQ) standardizes its business processes and makes them mandatory to adopt in all these branches. To enable such process management, a workflow management system proposed is set up at HQ site. It manages all product- and process- related data and staff in branches can access the data via web.

Specifically, HQ defines a standard process for problem handling. It requires that all branches should apply this process to handle problems encountered by their customers. The process is implemented as a workflow template in the workflow system, as illustrated in Figure 4.1. When staff in one branch receives a problem from customer, they initiate a process based on the workflow template and follow it till its completion.

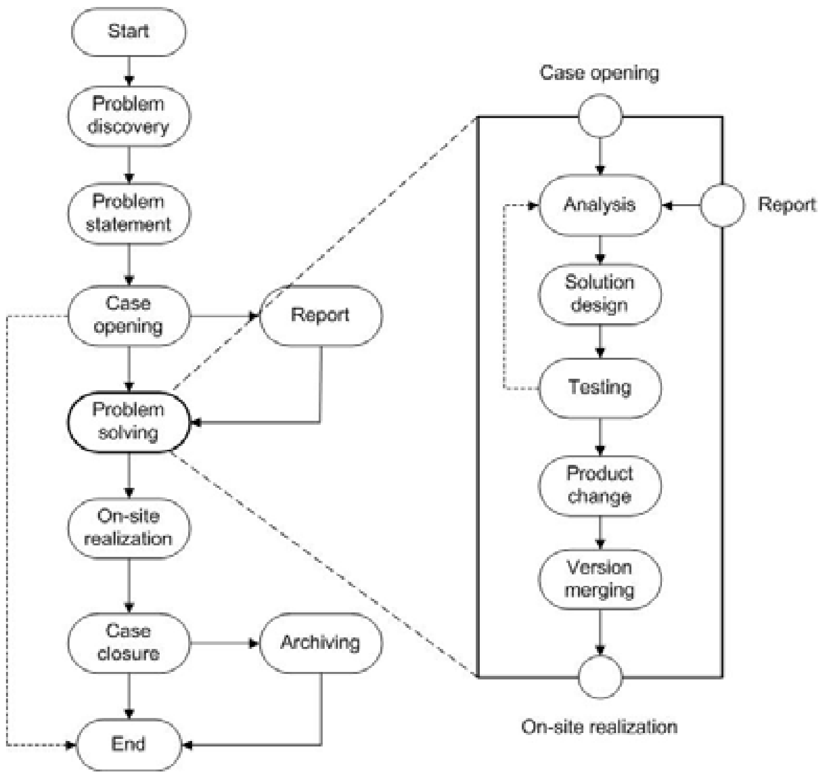


Figure 4. 1 Old Workflow Model

Following the template, a problem is handled in two major steps, problem solving and on-site realization, after it is found and formulated. In the workflow, problem solving involves several steps and thus is represented by a compound node which serves as a nesting node containing several tasks. At the moment of opening a case, subsidiaries need to report the case to HQ. When closing the case, subsidiaries should also archive related documents in HQ database.

The HQ manages all instances related to the problem handling process. To adapt to the new business environment, HQ decides to change the problem handling process in all its subsidiaries.

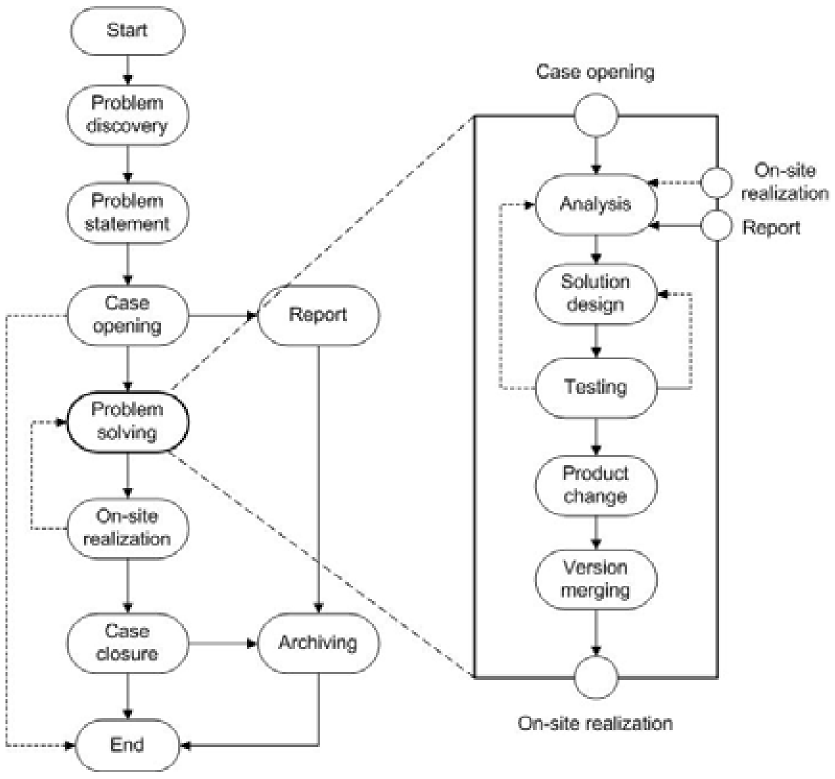


Figure 4. 2 New Workflow Model

There are 188 instances under the old template and they need to be transferred to the new template, as shown in Figure 4.2. The Table 4.2 below shows the states of all these instances before and after the transfer, in terms of:

(1) $n$ - instance number

(2) $N_f$ - finished nodes before transfer

(3) $P_f$ - finished percentage before transfer

(4) $\overset{\circ}{N}_f$ - finished nodes after transfer

(5) $\overset{\circ}{P}_f$ - finished percentage after transfer

Based on the data in the table, it is clear that tasks performed in early stages can be preserved after the transfer as the modifications happen in late stage of the workflow template.

Table 4. 2 Instance States

| $n$ | $N_f$ | $P_f$ | $\overset{\circ}{N}_f$ | $\overset{\circ}{P}_f$ |
|---|---|---|---|---|
| 11 | 0 | 0% | 0 | 0% |
| 4 | 1 | 7% | 1 | 7% |
| 14 | 2 | 14% | 2 | 14% |
| 30 | 3 | 21% | 3 | 21% |
| 18 | 4 | 29% | 4 | 29% |
| 5 | 5 | 36% | 5 | 36% |
| 2 | 6 | 43% | 6 | 43% |
| 1 | 7 | 50% | 7 | 50% |
| 5 | 8 | 56% | 8 | 56% |
| 2 | 9 | 42% | 8 | 56% |
| 11 | 10 | 71% | 8 | 56% |
| 7 | 11 | 79% | 8 | 56% |
| 8 | 12 | 86% | 8 | 56% |
| 23 | 13 | 93% | 8 | 56% |
| 47 | 14 | 100% | 8 | 56% |

## 4.7 Conclusion

In this chapter, an approach to facilitate dynamic workflow change is presented and implemented. The approach identifies work tasks, which have been finished before the change and can be preserved after the change, and holds their states without human intervention. Therefore, there is no need to re-execute these tasks and this feature is efficient in a large collaborative environment where a lot of parties are involved in the affected workflow.

The presented approach can be improved if the following further works can be done:

(1) Handling difficult issues such as script associated with affected work nodes;

(2) Capability to cope with other aspects such as document and resource;

(3) Mechanism to make this approach more robust and hassle-free in case of exception;

(4) Flexibility to incorporate various policies for dynamic workflow management.

# Chapter 5. Prototype Implementation

## 5.1 System Functions

The workflow-driven distributed collaborative design system provides the underlying process, session and data management that enable distributed collaborative design to take place within an integrated product development team. In this system, an integrated, hierarchical set of design automation tools have been developed and incorporated.

The concept of operation for the enterprise framework includes the ability to execute project plans, expressed as workflows, by teams of engineers. Execution of a workflow by a member of a design team initiates control commands to a CAD/CAE tools as relevant for the particular workflow step. This execution also initiates data transactions with the enterprise product data management system, local data management systems, and library systems, as relevant for the particular workflow step. In addition, the system couples project management tools with the design environment, which receives regular status updates as workflow steps are executed. This process facilitates effective, non-interfering project management.

Users execute the workflows using enterprise resource and knowledge management tools, which link to tools, data access mechanisms, and other services. This process allows designers to operate at a higher level of abstraction, allowing them to focus on the real design tasks instead of tool and data management, which significantly improves their productivity.

To support workflow usage, multiple workspace views for the environment should be provided. These views include:

1. Tool and application workspace;
2. A data workspace for product and reuse information;
3. Project/workflow workspaces.

The resources, data objects, and application available to particular engineers are determined by their identity and role in an authorization hierarchy implemented in the enterprise system.

Workflow management in the system comprises methods and tools to provide the project team with an environment that facilitates day-to-day work. A workflow-driven mechanism is provided. The workflow models are instantiated as workflows in the workflow management tool. The workflow captures:

1. Process steps and their precedence relationships;
2. Roles of personnel authorized/required to perform work;
3. Information objects involved (created, used, modified, destroyed, etc.) in the process step;
4. Tools to be launched or controlled at each step.

The WFMS graphically represents the workflows of a project (defined by workflow editor), enforces workflow use and tracks status of the workflows (carried out by workflow engine). Each activity in a workflow may be associated with multiple tools. An activity can be instantiated in two ways:

1. Clicking on the box representing the activity in a workflow by users;
2. Activating automatically by workflow engine, once specific conditions are satisfied.

When users exit the activity, the status of the activity is recorded. The workflow engine decides whether an activity is to be launched or not, based on the status of the activities that precede it in the workflow and the availability of the resources required. The workflow engine provides for pre-condition and post-condition scripts of the activities in a workflow. Examples of these activities include functions, such as checking for the existence of data objects, or translating data objects to the appropriate formats. As such, this removes the necessity of having to invoke those functions as required responsibilities for the design engineers, thereby enabling significant productivity improvement through increased focus on design tasks. Project engineers or supervisors (playing an administrator role) would normally be responsible for design and implementation of project plans based on workflows by using the system.

The workflows are hierarchical (including tasks, task groups, sub-process, etc.) and represent the various disciplines, particularly those associated with technology-intensive product design. They consist of reusable workflow segments, which can be combined in various configurations to address specific project needs. These segments consist of multiple process steps, each of which is also reusable. Option is made available to a user to make use of the workflow elements in current form to develop process plans based on a combination of reused workflow segments, individual process steps, and possible custom user steps.

The functionalities of the workflow system include: access controls, hierarchical workflow modeling capability, verification of workflow model, capability to track status of a project, project management interfaces and simulation of workflows. The workflow system also helps to capture useful metrics for projects. The metrics that should be collected include:

1. Time spent in a step;

2. Tool usage;

3. Person(s) performing the step.

## 5.2 System Architecture

By combining with the existing COCADE environment [59], the architecture of the prototype implementation is shown in Figure 5.1. In this architecture, a 3-tier structure is employed.

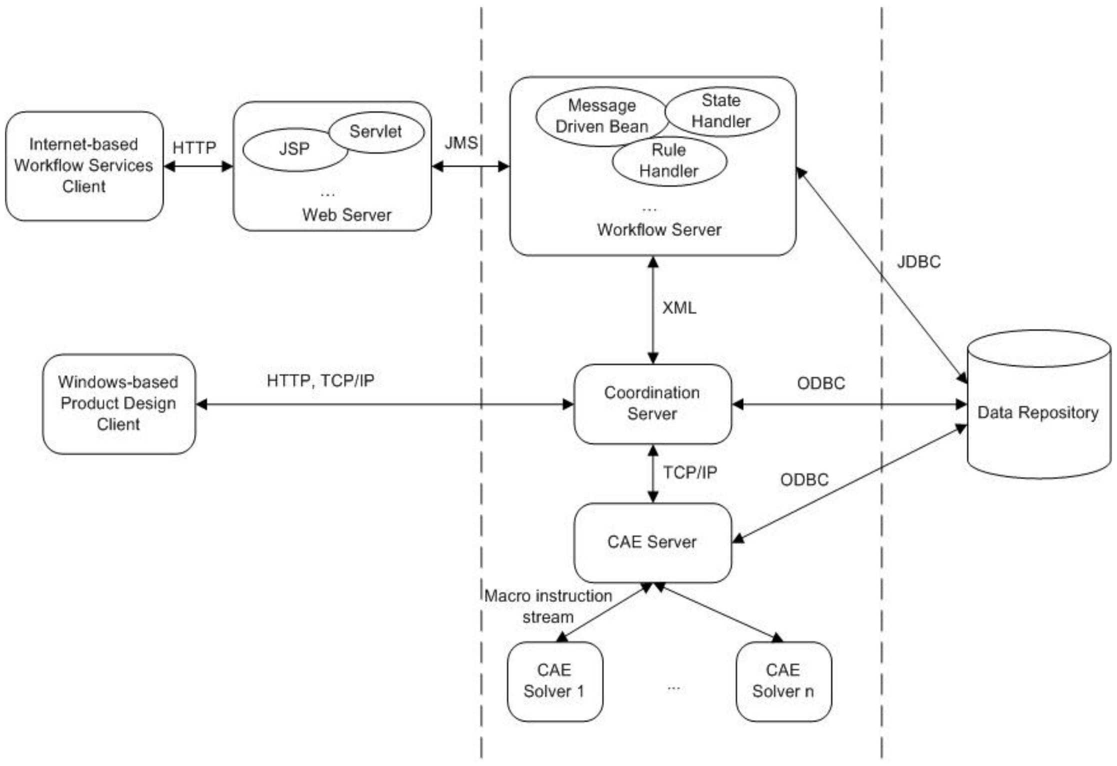


Figure 5. 1 Workflow-driven System Architecture

Such a 3-tier structure has the following advantages:

1. It is easier to modify or replace any tier without affecting the other tiers;
2. Separating the application and database functionality means better load balancing;
3. Adequate security policies can be enforced within the server tiers without hindering the clients.

In this framework, the workflow server and the coordination server are responsible for the implementation of business logic. They are the center of the whole system. XML files will carry out the communications between these two servers. As the focus of this thesis, the workflow sub-part adopts J2EE as its framework, which includes workflow clients, a web server, a workflow server and a data repository. The workflow clients are implemented by JAVA Applet which is embedded in Internet pages and explained by the Internet browser. JSP and servlet components residing in the web server are used to connect with the back-end server and generate the Internet pages that are to be shown to the clients. The workflow server is responsible for the modeling and execution of workflows, and returns results to the clients. As to the data repository, it stores all the specified information that is generated during design processes.

## 5.3 Case Study - Design of Head and Media

Hard disk drive is a technology-intensive product, and it is a very important industry in Singapore. Head and media are essential parts in a hard disk drive. These two parts are high-technology intensive and also closely coupled. Figure 5.2 illustrates the working mechanism of head and media in detail.

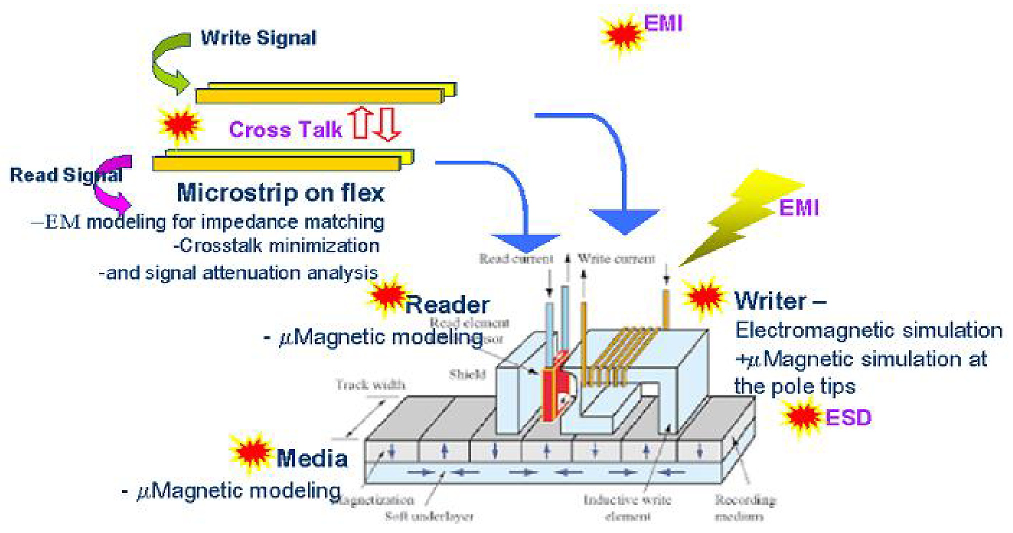


Figure 5. 2 Combination of Head and Media [64]

Following is a writing sequence:

1. The drive channel module receives data in binary form from the computer and converts them into a current in the head coil.

2. The interaction of the magnetic field generated by the current and the media results in magnetization of the media, whose direction depends on the direction of the current in the coil.

The reading process includes excitation of the current in the head coil when the head “senses” changes in the magnetic flux. The read voltage pulses at the flux transitions are then translated into sequences of bits of 0 and 1.

After checking the working process of the head and media, and understanding the interdependency between them, the design process for each part can be determined. Figure 5.3 shows the design process of the media. Firstly, the geometric model is created as the description of its specification. Then the seeds are generated on the geometric model. According to the distribution of seeds, the irregular grain structure can be obtained. After this step, the geometry is extruded from 2D to 3D. The microtrack model for the simulation of transition jitter noise and other effects can then be established.

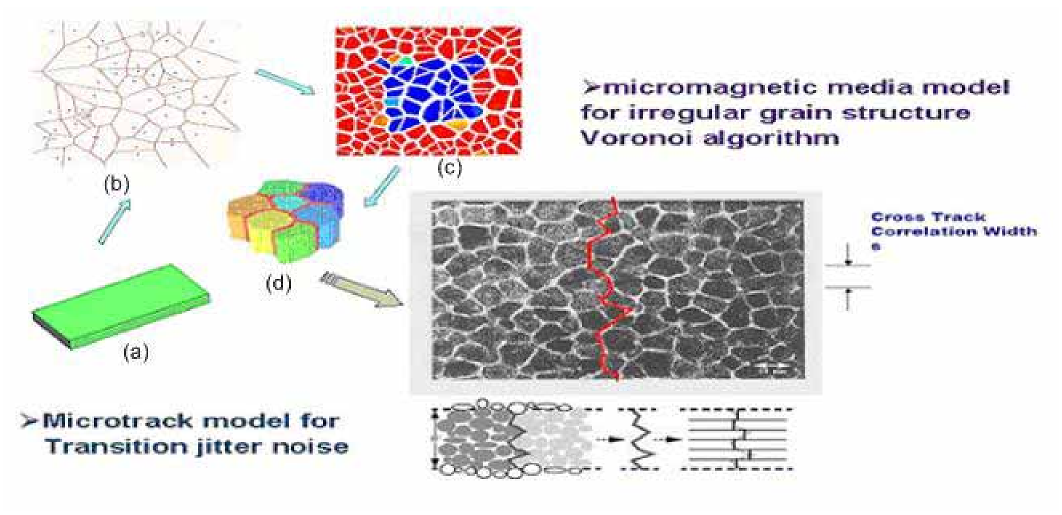

(b)
(c)
➢micromagnetic media model for irregular grain structure Voronoi algorithm
(d)
(a)
Cross Track Correlation Width
➢Microtrack model for Transition jitter noise

Figure 5. 3 Design Process of Media

The design process of the head is similar to the design of the media. These design processes will be executed by engineers from different disciplines, such as material, electronics, magnetics, and aerodynamics etc. These engineers are often at different locations. Their work needs to be organized effectively and efficiently so that the design can be finished on time and good product quality can be achieved.

By using the aforementioned workflow-driven framework, we can get the general process of a head and media design scenario (see Figure 5.4). The workflow administrator distributes the tasks to the engineers according to the pre-defined workflow template. During the design process, the workflow management system will serve as a back-end server and provide real-time process management, including instance initialization, instance execution and dynamic workflow management, etc.

The practical design activities will be carried out by the COCADE system that is installed on each computer. Part of the process information, e.g., task state and available resource, is shared by the two kinds of systems.

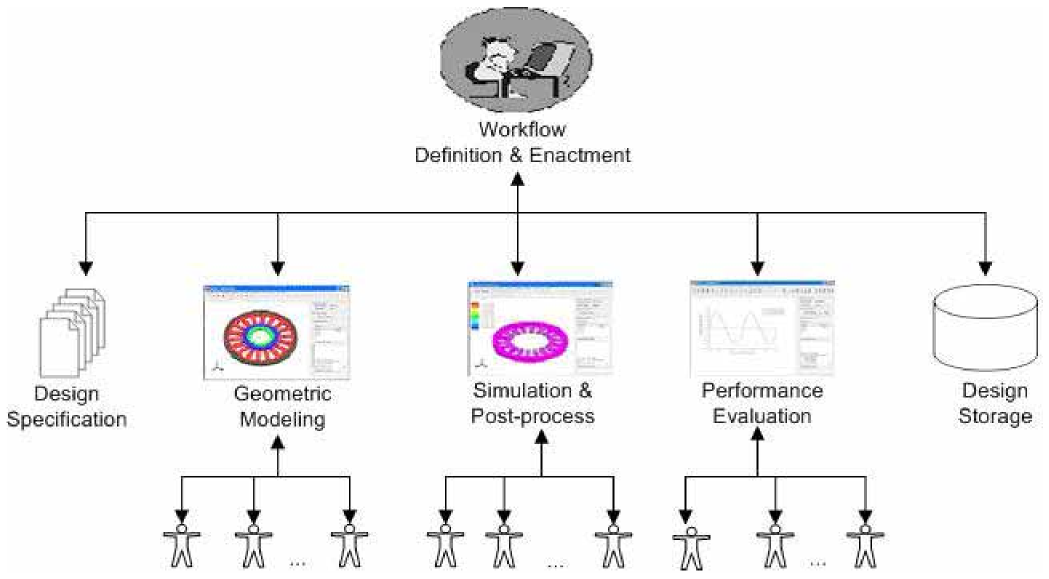


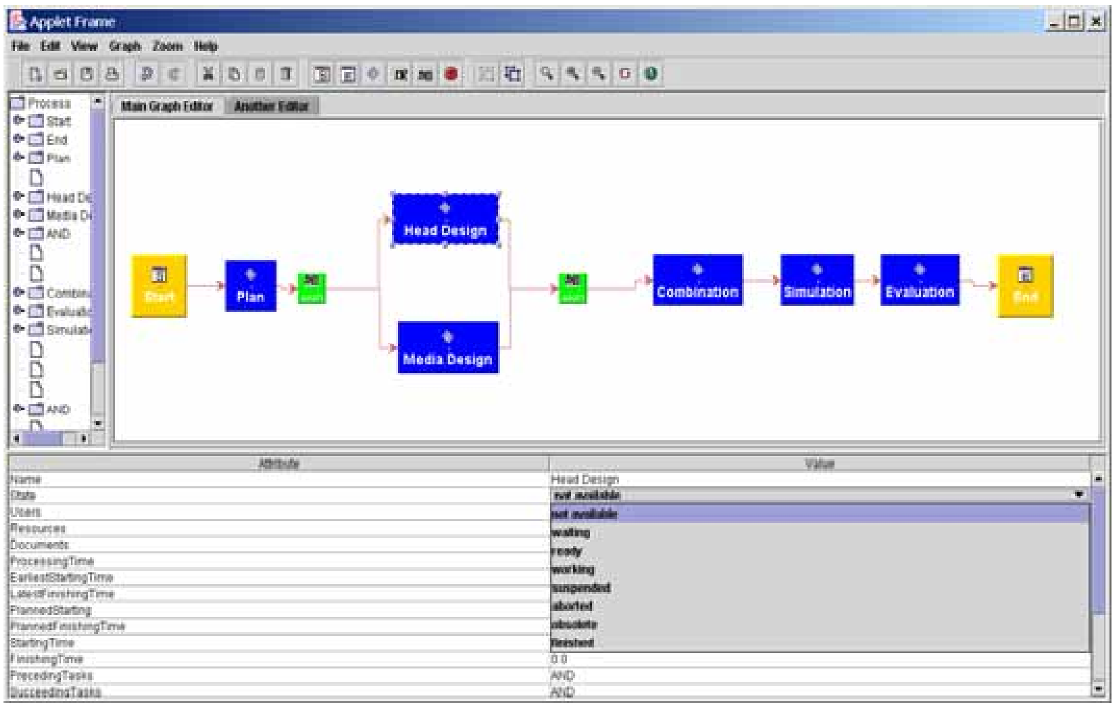
Figure 5. 4 A Distributed Collaborative Head/Media Design Scenario

The detailed design process is shown below:

**Workflow Model Definition**

In this case, the workflow model is firstly created with consideration of the following:

- The requirements of product design and product specification;
- Appropriate knowledge that can be used to enhance the workflow process;
- The most efficient and useful methods of integrating and transferring knowledge;
- The needs of distributed designers and the required modes of communication.

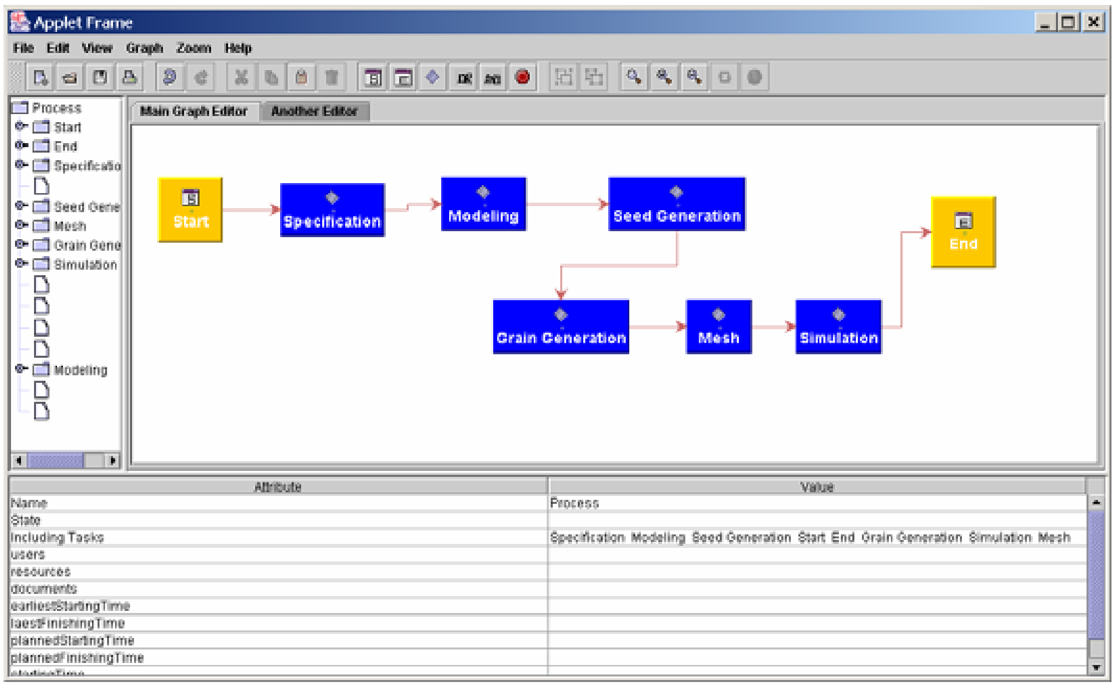


| Attribute | Value |
|---|---|
| Name | Head Design |
| State | not available |
| Users | |
| Resources | |
| Documents | |
| ProcessingTime | |
| EarliestStartingTime | |
| LatestFinishingTime | |
| PlannedStarting | |
| PlannedFinishingTime | |
| StartingTime | |
| FinishingTime | 0.0 |
| PrecedingTasks | AND |
| SucceedingTasks | AND |

Figure 5. 5 Workflow Model Definition

As shown in Figure 5.5, the task attributes, such as state, users, resources, documents, time requirements, etc., are described. The possible dependencies are evaluated in the workflow model and corresponding solutions are described as new tasks. Some tasks can also be a sub-process which includes several detailed sub-tasks. For example, the media design task can be shown in further details as shown in Figure 5.6.

A major task of the workflow model administrator is to predict as much as possible the potential interdependent tasks in the product modeling stage and computing stage. This can help to avoid conflicts and errors occurring in the product development process in the earlier stage, and save time and resources. After completion of the process definition, an XML file is created by the workflow engine to record all necessary information related to the design process. This file will be the template of instances of the head/media design process. It can be modified by the workflow administrator, and also reused by other similar design processes.



| Attribute | Value |
|---|---|
| Name | Process |
| State | |
| Including Tasks | Specification Modeling Seed Generation Start End Grain Generation Simulation Mesh |
| users | |
| resources | |
| documents | |
| earliestStartingTime | |
| laestFinishingTime | |
| plannedStartingTime | |
| plannedFinishingTime | |

Figure 5. 6 Detailed Process of Media Design

## Workflow Instance Execution

Before a practical design process can start, the pre-defined workflow model should be instantiated. Then a process instance will be created which follows the workflow model. In this case, the whole design flow is:

At the very beginning, the start node is activated. Then the next node’s state is transferred to ready state. If this node can get enough resources as described in its attribute table, it is ready to go. The workflow administrator can set it as automatic node or manual mode. If it is automatic, this node can immediately run; otherwise, it should wait for a run command. When it is in the running state, its task engine will monitor its request as well as its state. As a service is requested, the task engine will send the request to the workflow engine. Then the workflow engine will communicate with the service manager and look up the service table. While the service name is searched and found, the service parameters will be passed from the task engine to the workflow engine, and then to the service manager. The service manager invokes the remote method provided by the application tools and feedbacks the results to the task.

Beside the asynchronous communication between tasks, the involved participants can also create synchronous collaborative session to carry out some design tasks in the same workplace when intensive interactions are necessary. They work intensely with one another, observing and understanding each other's intentions. All participants contribute with their special expertise at moments when they have the appropriate knowledge to handle the situations.

By adopting the aforementioned steps, a head/media design process, which is carried out by dispersed engineers from multiple disciplines, can be completed.

**Workflow Model Evaluation**

After the completion of the design process, the performance of the workflow model can be evaluated by analyzing the log data stored in the process database. Different views of the model's performance can be created, e.g., the diagram of execution time of every activity for each instance can be plotted, the designer can identify the activity that is most time-consuming, and the one that is the bottleneck of the process, etc. In this way, the workflow model can be modified and improved.

By combining with COCADE system, the whole design process can be defined and executed. Figure 5.7 shows the snapshots of the results of the system. They are the mesh (left) and computed results of magnetization distribution (right).

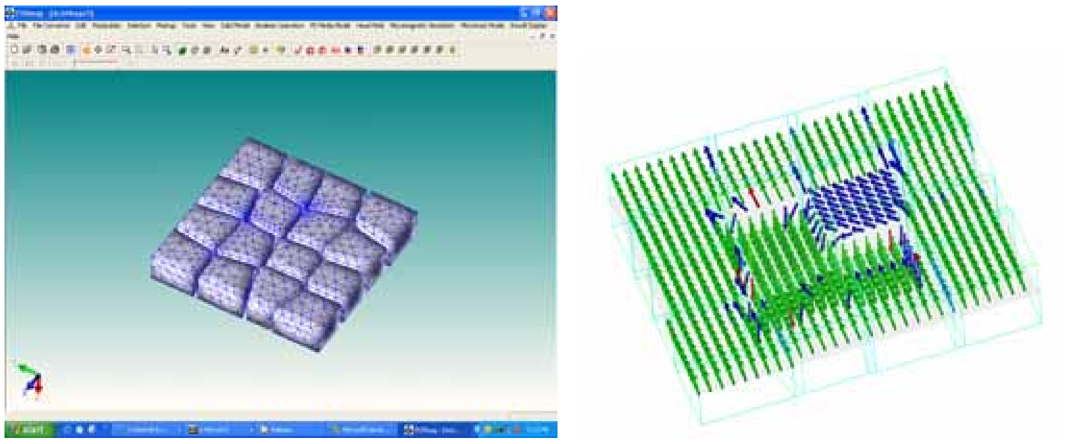


Figure 5. 7 System Snapshots

## 5.4 Summary

This chapter presents an overview of the prototype system that has been developed to validate the concepts proposed in this thesis. The prototype consists mainly of a build-time component and a run-time component. The build-time component provides a graphical user interface through which the workflow modeler can create, browse and update the workflow model. The run-time component mainly supports the functionality to enact the workflows, and call the CAD/CAE services provided by different engineers.

# Chapter 6. Conclusion and Future Work

This chapter summarizes the contributions during the course of the research and the directions for possible future work.

## 6.1 Contributions

The major contribution resulting from the research presented in this thesis consists of three major parts. Firstly, a workflow-driven infrastructure has been proposed that can effectively support distributed collaborative design process. Secondly, a workflow meta-model used to construct the distributed collaborative design process is introduced. It allows various aspects of design processes to be captured. Thirdly, the communication method has been investigated and a workflow-driven communication approach to better support the aforementioned infrastructure has been suggested.

The workflow-based infrastructure can effectively manage the distributed collaborative design activities and resources. In comparison with existing infrastructures, the workflow-based infrastructure has the following distinguishing features:

1. Process control. The infrastructure supports the effective control of distributed collaborative design process. By using WFMSs, design processes residing in different and distributed domains can be connected together. The representative workflow engines are selected to be responsible for the connection. For the tasks in the same process, the workflow engine of the process coordinates the behaviors of each task and interacts with the process database and service manager.

2. Service management and session management. The engineers define and register their own services via the service manager. The CAD/CAE application tools also occupy a place in the service table to be queried and called by the requesters. Services can be added and removed via the service table. By effectively organizing and automatically invoking design sessions, the design environment can be set up quickly. Thus the design time is vastly shortened. And the concurrency happening between the sessions can be controlled by the workflow engine.

3. Dynamic features. As all the workflow models and workflow instances are stored in the process database, the workflow engine can compare the current model with the new model when changes occur. The design processes do not need to pause for a long time to accommodate the new changes that take place dynamically. Once the workflow engine completes the verification, checking and comparison of new model, the work that has been done can be resumed and a lot of resources can be saved.

4. Simulation functions. Before the actual design activities, if the design process can be simulated, the deficiency can be found at early stage and the process can be improved. To implement this function, the rule handler needs to be developed further to accommodate the user-defined events and deal with exceptions. It will also be the emphasis of future work.

Because the workflow model has to be modeled well to suit the distributed collaborative environment, the workflow meta-model is established by considering the process meta-model and organization meta-model. These meta-models are explicitly defined in UML diagrams. In addition, the role-based organization model has been given in detail.

The communications between tasks and WFMSs are also discussed. A workflow-driven approach is used to carry out the communications. To support asynchronous communication, different communication models are compared and the event-based model is adopted. The states of tasks, subprocesses, and processes are used to drive the occurrences of user-defined events. A task engine as well as workflow engine is proposed to manage the state-oriented communication through the user-defined state-event table.

Finally, the feasibility of the main concepts presented in this thesis has been demonstrated by the implementation of a prototype.

## 6.2 Directions for Future Work

In the following, several directions for possible future work are identified.

***Support of Multiple Workflow Modelers***

In the distributed collaborative environments, multiple persons might be involved in the workflow modeling. Thus, adequate support for the development of multiple workflow modelers should be provided. To address this problem, the concepts of workspaces could be used. Workplaces are named repositories for artifacts, where workspaces differ according to the extent of access to the contents of the workspace. Artifacts can be moved among workspaces by check-in/check-out operations.

### *Workflow Network Construction*

The WFMSs in the distributed collaborative environments form a workflow network. Such a network integrates all aspects of the design process. The proposed infrastructure is a part of this network. To construct a complete and robust network, more detailed studies should be done.

### *Authorization Constraints*

It is usually not desirable that all users can perform such tasks as model modification or service changes. Rather, each user should only be allowed to carry out a well-defined scope of tasks. In this respect, authorization constraints should be definable which are enforced by the WFMS. Thus the approach presented in this thesis may be extended with the aforementioned concepts.

### *Prototype System*

Some functions are not completed in the system presented in this thesis, e.g., the interaction between workflow engines and the workflow simulation that is an important part of the WFMS. The system still needs to be enhanced and developed further.

### *Integration with Design Chain Operation Reference (DCOR) Model*

As DCOR is a rather newly launched model for design process management, workflow technology can surely find its place in integration with DCOR. It can contribute to the establishment of this model, in terms of model definition, model monitoring and model evaluation. All these can be studied in the further research.

# References

[1] Tian, G.Y., Taylor, D. Design and Implementation of a Web-based Distributed Collaborative Design Environment. In Proc. 5th International Conference on Information Visualisation, IEEE, 2001, London, UK, pp. 703-707.

[2] MacGregor, S.P., Thomson, A.I., and Juster, N.P. A Case Study on Distributed, Collaborative Design: Investigating Communication and Information Flow. In Proc. 6th International Conference on CSCW in Design, London, Ontario, July 2001, pp. 249 – 254.

[3] Qiuli Sun, Kurt Gramoll. Internet-based Distributed Collaborative Environment for Engineering Education and Design. In Proc. 2001 American Society for Engineering Education Annual Conference & Exposition.

[4] Gun-Dong F. Pahng, Nicola Senin, David Wallace. Modeling and Evaluation of Product Design Problems in a Distributed Design Environment. In Proc. 1997 ASME Design Engineering Technical Conferences, September 1997, Sacramento, California.

[5] Michael P Case and Stephen C-Y Lu. Discourse Model for Collaborative Design. Computer-Aided Design, Vol. 28, No. 5, pp. 333-345. 1996.

[6] Lee, J. Y., Kim, H., and Han, S. B. Web-enabled Feature-based Modeling in a Distributed Design Environment. In Proc. 1999 ASME Design Engineering Technical Conference, September 1999, Las Vegas, Nevada.

[7] Charles S. Smith and Paul K. Wright. Cybercut: A Networked Manufacturing Service. In Proc. 1st International Conference on Managing Enterprises-Stakeholders, Engineering, Logistics and Achievement, July 1997, Loughborough Univ. UK.

[8] Toshiki Mori and Mark R. Cutkosky. Agent-based Collaborative Design of Parts in Assembly. In Proc. 1998 ASME Design Engineering Technical Conferences, September 1998, Atlanta, Georgia.

[9] http://www.ptc.com, January 2001.

[10] http://www.enovia.com/, January 2001.

[11] http://www.sdrc.com/, January 2001.

[12] Li, L., Chenhui Yang, Tangqiu Li. Managing Dynamic Shared State in Virtual Space for Collaborative Design. In Proc. 6th International Conference on Computer Supported Cooperative Work in Design, July 2001, Ont., Canada, pp. 61-65.

[13] Van Der Wolf, P., Bingley, P. Dewilde, P. On the Architecture of a CAD Framework: The NELSIS Approach. In Proc. 1st European Design Automation Conference, 1990, pp. 29-33.

[14] Bushnell, M., Director, S.W. Automated Design Tool Execution in the Ulysses Design Environment. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, March 1989, pp. 279-287.

[15] Brockman, J.B., Director, S.W. A Schema-Based Approach to CAD Task Management. In Proc. 3rd IFIP WG 10.2 Workshop on Electronic Design Automation Frameworks, 1992.

[16] Chan, F., Spiller, M., Newton, R. WELD – An Environment for Web-Based Electronic Design. In Proc. The International Conference on Computer Aided Design, 1998, pp. 146-151.

[17] Lavana, H. A Universally Configurable Architecture for Taskflow-Oriented Design of a Distributed Collaborative Computing Environment. 2000. PhD

Thesis – Electrical and Computer Engineering, North Carolina State University, Raleigh.

[18] C-Lab. Astai(R). Available at http://www.c-lab.de/astair/.

[19] John Welsh, Dr. Bipin Chadha, Jeffreay Stavash. Distributed Collaborative Design Approach to Address Total Ownership Costs. MCE (Mission Critical Enterprise) Systems Symposium, May 1999.

[20] Hammer, M., Champy, J. Reengineering the Corporation: The Manifesto for Business Revolution. New York: Harper Business, 1993.

[21] Barua, A.C, Sophie Lee, H. and Whiston, A. B. The Calculus of Reengineering. Information Systems Research, pp. 409-428. 1996.

[22] David Hollingsworth. Workflow Management Coalition The Workflow Reference Model. Available at http://www.wfmc.org.

[23] Shung-Bin Yan and Feng-Jian Wang. A Cooperative Framework for Inter-Organizational Workflow System. In Proc. 27th Annual International Computer Software and Applications Conference, November 2003, Dallas, Texas, pp. 64-71.

[24] Steven P. MacGregor, Avril I. Thomson, Neal P. Juster. A Case Study on Distributed, Collaborative Design: Investigating Communication and Information Flow. The Sixth International Conference on Computer Supported Cooperative Work in Design Advance Program, June 2001, Ont., Canada.

[25] Lindemann, U., R. Anderl, H.Gierhardt and G. M. Fadel. 24hr Design and Development – An Engine Design Project. In Proc. CoDesigning 2000, Sept. 2000, Coventry, UK.

[26] Vadhavkar, S. Team Interaction Space Effectiveness for Globally Dispersed Teams: Theory and Case Studies. Doctor of Science Thesis, Massachusetts Institute of Technology. 2001.

[27] Larsson, A., P. Torlind, A. Mabogunje and A. Milne. Distributed Design Teams: Embedded One-on-One Conversations in One-to-Many. In Proc. Common Ground: Design Research Society International Conference, 2002, London, UK.

[28] MacGregor, S. P., A. I. Thomson and N. P. Juster. Information Sharing Within a Distributed, Collaborative Design Process: A Case Study. In Proc. ASME Design Engineering Technical Conference, September 2001, Pittsburgh, Pennsylvania.

[29] MacGregor, S. P., A. T. Thomson and N. P. Juster. A Multi-level Process Based Investigation of Distributed Design. In Proc. Engineering Design Conference 2002, July 2002, KCL, London.

[30] Scrivener, S. A. R., D. Harris, S. M. Clark, T. Rockoff and M. Smyth. Design at a Distance via Real-time Designer-to-designer Interaction. Design Studies. Vol. 14, No. 3, pp. 261-282.

[31] Huang, J. Knowledge Sharing and Innovation in Distributed Design: Implications of Internet-based Media on Design Collaboration. In Proc. Design Computing on the Net'99 (DCNet'99), 1999, Sydney, Australia.

[32] Eisenstein, P. The parts Come Together. Professional Engineering: 31.

[33] Hietikko, E. and E. Rajaniemi. Visualised Data-tool to Improve Communication in Distributed Product Development Projects. Journal of Engineering Design, Vol. 11, No, 1, pp. 95-101.

[34] Cheng, N. and T. Kvan. Design Collaboration Strategies. In Proc. 5th International Conference, August 2000, Ampt van Nijkerk, pp. 62-73.

[35] Sonnenwald, D. H. Communication Roles That Support Collaboration During the Design Process. Design Studies, Vol. 17, No. 3, pp. 277-301. 1996.

[36] Maher, M. L. and J. H. Rutherford. A Model for Synchronous Collaborative Design Using CAD and Database Management. Research in Engineering Design, Vol. 9, No. 2, pp. 85-98. 1997.

[37] Court, A. W., S. J. Culley and C. A. McMahon. The Influence of Information Technology in New Product Development: Observations of an Empirical Study of the Access of Engineering Design Information. International Journal of Information Management, Vol. 17, No. 5, pp. 359-375. 1997.

[38] Pahl, G. and W. Beitz. Engineering Design - A Systematic Approach. London: SpringerVerlag. 1988.

[39] Pugh, S. Total Design: Integrated Methods for Successful Product Engineering. Wokingham: Addison-Wesley. 1991.

[40] French, M. J. Conceptual Design for Engineers. London: SpringerVerlag. 1999.

[41] Prasad, B. Concurrent Engineering Fundamentals: Integrated Product and Process Organization. New Jersey: Prentice-Hall. 1996.

[42] Lu, S. C and J. Cai. A Collaborative Design Process Model in the Sociotechnical Engineering Design Framework. Artificial Intelligence for Engineering Design, Analysis and Manufacture, Vol. 15, pp. 3-20. 2001.

[43] Adelson, B. Developing Strategic Alliances: A Framework for Collaborative Negotiation in Design. Research in Engineering Design, Vol. 11, No. 3, pp. 133-144. 1999.

[44] Vadhavkar, S. and F. Pena-Mora. Empirical Studies of the Team Interaction Space: Designing and Managing the Environments for Globally Dispersed Teams. International Workshop on the Role of Empirical Studies in Understanding and Supporting Engineering Design Work, NIST, Gaithersburg, MD, USA. 2002.

[45] McGrath, J. E. Groups: Interaction and Performance. New Jersey: Prentice-Hall. 1984.

[46] Olson, J., G. Olson, M. Storrosten and M. Carter. Groupwork Close Up: A Comparison of the Group Design Process with and without a Simple Group Editor. ACM Transactions on the Information Systems, Vol. 11, No. 4, pp. 321-348. 1993.

[47] Lipnack, J. and J. Stamps. Virtual teams: People Working Across Boundaries with Technology. New York: John Wiley & Sons. 2000.

[48] DeSanctis, G. and P. Monge. Communication Processes for Virtual Organizations. Organization Science, Vol. 10, No. 6, pp. 693-703. 1999.

[49] Tagg, R. Workflow in Different Styles of Virtual Enterprise. In Proc. Workshop on Information Technology for Virtual Enterprise (ITVE 2001). January 2001, Gold Coast, Australia.

[50] F Casati, S Ceri, B Pernici, G Pozzi. Workflow Evolution. In Proc. 15th International Conference on Conceptual Modeling, ER'96, Cottbus, Germany, pp. 438-455.

[51] W M P Van Der Aalst, T Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. Theoretical Computer Science, Vol. 270, No. 1-2, pp. 125-203. 2002.

[52] W M P Van Der Aalst. How to Handle Dynamic Change and Capture Management Information? An Approach Based on Generic Workflow Models. International Journal of Computer Systems, Science, and Engineering, Vol. 16, No. 5, pp. 295-318. 2001.

[53] Shingo Yamaguchi, Qi-Wei Ge, Minoru Tanaka. Performance Evaluation on Change Time of Dynamic Workflow Changes. IEICE Transactions, Vol. E83-A, No. 11, pp. 2177-2187. 2000

[54] Clarence Ellis, Karim Keddara, Grzegorz Rozenberg. Dynamic Change within Workflow Systems. In Proc. Conference on Organizational Computing Systems, 1995, CA, USA, pp. 10-18.

[55] Clarence Ellis, Karim Keddara. ML-DEWS: Modeling Language to Support Dynamic Evolution within Workflow Systems. Computer Supported Cooperative Work, Kluwer Academic Publisher 9, 2000, pp. 293-333.

[56] Manfred Reichert, Peter Dadam. ADEPTflex-supporting Dynamic Changes of Workflows without Losing Control. Journal of Intelligent Information Systems, Vol. 10, No. 2, pp. 93-129. 1998.

[57] Manfred Reichert, Peter Dadam. Framework for dynamic changes in workflow management systems. In Proc. 8th International Conference and Workshop on Database and Expert Systems Applications (DEXA'97), 1997, Toulouse, pp. 42-48.

[58] Hoque, R. CORBA for Real Programmers. Academic Press/Morgan Kaufmann. 1999.

[59] D. W. Sun, L. W. Ruan, Z. J. Liu, J. M. Zhao, W. F. Lu and X. G. Ming. Concurrency in a Distributed Collaborative CAD/CAE Environment. In Proc.

11th ISPE International Conference on Concurrent Engineering: Research and Applications, July 2004, Singapore.

[60] Malone, T. W., Crowston, K. What is Coordination Theory and How Can It Help Design Cooperative Work Systems. In Proc. the conference on Computer-supported cooperative work, 1990, Los Angels (CA), pp. 357-370.

[61] Rosemann, M., zur Muehlen, M. Evaluation of Workflow Management Systems - a Meta Model Approach. The Australian Journal of Information Systems, Vol. 6, No. 1, pp. 103-116. 1998.

[62] Jim Farley. Java Distributed Computing. Cambridge: O'Reilly. 1998.

[63] http://www.meta-model.com/.

[64] Jinghuan Chen, Jaekyun Moon, and Kia Bazargan. A Reconfigurable FPGA-Based Readback Signal Generator For Hard-Drive Read Channel Simulator. In Proc. of the 39th Design Automation Conference (DAC 2002), June 2002, New Orleans, LA, pp. 349-354.

# Appendix

## List of Publications

I co-authored the following technical papers:

1. D.W.Sun, X.H.Xiong, L.W.Ruan, Z.J.Liu, J.M.Zhao, Y.S.Wong, "Workflow-driven collaborative session management in product life cycle management via internet". Paper presented in IEEE Engineering Management Conference IEMC 04, Singapore, Oct 2004.
2. D.W.Sun, X.H.Xiong, L.W.Ruan, Z.J.Liu, J.M.Zhao, W.F.Lu, X.G.Ming, "Concurrency in a distributed collaborative CAD/CAE environment". Submitted to ASME Journal of Product Research for publication.
3. Z.M.Qiu, Y.S.Wong, X.H.Xiong, Z.J.Liu, "Workflow Instance Transfer for Dynamic Workflow Change Management". Submitted to International Journal of Computing & Information Science in Engineering.
4. X.H.Xiong, Z.J.Liu, and Y.S.Wong, "Collaborative CAE Oriented Software Development for Electromagnetic Design and Analysis". Submitted to IEEE Computational Magnetics Conference 2005.