

國立交通大學

資訊工程學系
碩士論文

以代理人技術為基礎來設計一個新的工作流程
管理系統



Using Software Agents to Design a Modern
Workflow Management System

研究生：張瓊文

指導教授：王豐堅 教授

中華民國九十三年六月

以代理人技術為基礎來設計一個新的工作流程管理系統

Using Software Agents to Design a Modern Workflow

Management System

研究生：張瓊文

Student：Chiung-Wen Chang

指導教授：王豐堅 博士

Advisor：Dr. Feng-Jian Wang



A Thesis

Submitted to Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of Master

In

Computer Science and Information Engineering

June 2004

HsinChu, Taiwan, Republic of China

中華民國九十三年六月

以代理人技術為基礎來設計一個新的工作流程 管理系統

研究生：張瓊文

指導教授：王豐堅 博士

國立交通大學

資訊工程研究所

新竹市大學路 1001 號



傳統的工作流程管理系統基本上是一個單一中央集中管理式的架構，這點從現今的網際網路系統觀點來看顯得慢又不適宜，其流程的執行會受到事先定義好的資源分配所限制，在任務排班上面也缺少其調度的彈性；它跟所有使用者之間的操作介面都是固定的，缺少了因人因環境而異的可變通性。基於以上所提及的種種不方便因素考量下，這篇論文引進了代理人的觀念技術到工作流程管理的技術上，藉由代理人本身所具備的性質以及其運作的機制來提出一個新的工作流程管理系統的模式，此架構對上述所提到的不方便性，提供一個合理而且有彈性的解決方式。

關鍵字：代理人、工作流程、工作流程管理系統

Using Software Agents to Design a Modern Workflow Management System

Student: Chiung-Wen Chang

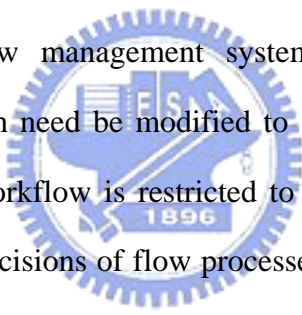
Advisor: Dr. Feng-Jian Wang

Institute of Computer Science and Information Engineering

National Chiao Tung University

1001 Ta Hsueh Road, Hsinchu, Taiwan, ROC

Abstract



The traditional workflow management systems (WfMSs) are based on a centralized architecture, which need be modified to suit for the Internet technology nowadays. For example, a workflow is restricted to pre-defined resource allocation constraints, and the routing decisions of flow processes cannot be changed easily and dynamically, i.e. lack of schedule flexibility. The operation data filled in a manipulation interface is generally fixed and carried to all participants based on a process generator. It is not flexible to provide suitable views and operations for different participants or roles. Besides, if the operation data is small, it is not necessary to apply current big applications to interact with participants. Thus, this thesis introduces agent and artifact technologies into WfMSs and presents a new WfMS architecture that could provide adequate and flexible solutions to cope with these shortcomings.

Keywords: agent, workflow, workflow management system (WfMS)

誌謝

本篇論文的完成，首先要感謝我的指導教授王豐堅博士兩年來不斷的指導與鼓勵，讓我在軟體工程及工作流程的技術上，得到很多豐富的知識與實務經驗。另外，也非常感謝我的畢業口試評審委員楊鎮華博士以及朱正忠博士，提供許多寶貴的意見，補足我論文裡不足的部分。

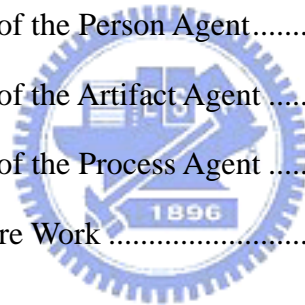
其次，我要感謝實驗室的夥伴們，有博士班建偉學長督導我寫論文，對論文適時給予了許多重要的思考方向及建議；而其他學長姐們熱心地參與幫忙和討論，讓我學得許多論文技巧，得以順利的撰寫論文。當然，值得一提的是我們這屆畢業生吉正、祖年及大立，在各方面彼此不斷的砥礪與照顧下，使得大家在各個領域的技術及理論上都能夠有所成長。

最後，我要感謝我的家人，因為有你們的支持，讓我能心無旁騖地讀書、作研究然後到畢業，由衷地感謝你們大家一路下來陪著我走過這段研究生歲月。文末也僅將我這一點點成就，獻給我在天上最敬愛的祖父。

Table of Contents

摘要.....	i
Abstract.....	ii
誌謝.....	iii
Table of Contents	iv
List of Figures	vi
List of Programs.....	vii
Chapter 1. Introduction	1
Chapter 2. Background	5
2.1 Workflow Technology and WfMS	5
2.2 Software Agents Technology	6
2.3 Related Research.....	7
Chapter 3. System Architecture.....	11
3.1 Overview.....	11
3.2 System Architecture — Design Phase	12
3.3 System Architecture — Execution Phase	14
3.3.1 Person Agent	15
3.3.2 Artifact Agent.....	18
3.3.3 Process Agent.....	20
3.3.4 Workflow Manager	22
3.4 An Example.....	24
Chapter 4. Flow Behaviors	28
4.1 Workflow Enactment	28
4.1.1 Automatic.....	28
4.1.2 Manual	29

4.2 Workflow Termination	30
4.2.1 Forced by External Power	31
4.2.2 Complete	32
4.3 The Detailed Interactions of Workflow Enactment	32
4.3.1 Accomplish a Task	33
4.3.2 Split a Task	34
4.3.3 Merge Tasks	36
Chapter 5. Implementation Issues	38
5.1 ISE Mobile Agent System	38
5.2 Interfaces of Workflow Manager	38
5.3 Agent Script Format	41
5.3.1 Script Format of the Person Agent	41
5.3.2 Script Format of the Artifact Agent	44
5.3.3 Script Format of the Process Agent	46
Chapter 6. Conclusion & Future Work	49
Reference	50



List of Figures

Figure 1.	WfMC's Workflow Reference Model.....	5
Figure 2.	System Architecture Overview.....	11
Figure 3.	System Architecture — Design Phase	12
Figure 4.	System Architecture — Execution Phase.....	14
Figure 5.	Instantiation of a Person Agent	16
Figure 6.	Lifecycle of an Artifact Agent	19
Figure 7.	Interactions of the Workflow Manager.....	22
Figure 8.	A Simple Model of Ask-for-leave Workflow	25
Figure 9.	Interactions of Enacting a Workflow Automatically	28
Figure 10.	Interactions of Enacting a Workflow Manually.....	29
Figure 11.	Interactions of Stopping a Workflow	31
Figure 12.	Interactions of Completing a Workflow.....	32
Figure 13.	Interactions of Accomplishing a Task	33
Figure 14.	Interactions of Tasks Splitting	34
Figure 15.	Interactions of Tasks Merging	36

List of Programs

Program 1. Goal of Person Agent	41
Program 2. Fact of Person Agent.....	41
Program 3. Procedure of Starting a Workflow	42
Program 4. Procedure of Querying a Workflow Progress	42
Program 5. Procedure of Replying a Workflow Progress	43
Program 6. Plan of Person Agent	44
Program 7. Plan of Artifact Agent.....	45
Program 8. Plan of Process Agent	48



Chapter 1. Introduction

Workflow management came mainly from the domain of office automation, where all kinds of documents need to be digitalized and transferred among co-workers. From paper works to e-form data manipulation nowadays, this technology has evolved for decades. Due to the capability of modeling, executing, and monitoring processes, workflow technologies attracted a lot of attentions and were deeply discussed. With the evolution of Internet technology, the requirements of workflow also increase in various applications. Besides, the communication information between the departments in an organization might be changed. Traditional WfMS technology needs large more efforts to support these demands. The defects of a traditional WfMS architecture can be discussed with four significant aspects: scalability, extensibility, flexibility and adaptability [14].

- Scalability

Basically, the conventional WfMS is a centralized architecture, which may not be useful for current technologies. For example, when large-scale information is applied by the users in an organization, the centralized server may be deemed as a bottleneck.

- Extensibility

In a conventional WfMS, the business process scenarios and data representations are almost restricted before execution. A WfMS is usually utilized to integrate diverse application software, so that kinds of mechanisms and adaptor programs for interoperability change frequently. It cannot be easily extended to meet new requirements and standards. Furthermore, the issues about exception handling for unexpected running workflows are emerging and

need to be studied in accordance with the extensibility.

- Flexibility

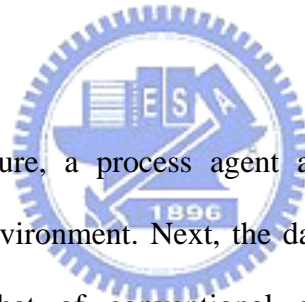
Flexibility of a WfMS affects how workflow processes are designed, executed and managed. Traditionally, initiation and completion conditions of each process activity can be statically described only. Artifact representations and routings are also defined in strict and limited rules. Besides, the client tools provide unitary interfaces to all users. A modern WfMS needs to support flexible process representations, i.e., which can be created, modified and deleted dynamically. Different and configurable interfaces might be provided for discrepancy of user groups based on their individual needs, preference and expertise.

- Adaptability

The enactment services of current WfMSs are usually restricted by predefined flow definitions. For example, the handling routines to all events need to be well defined and set in advance, so that the execution of workflows lacks flexibility in task scheduling and dispatching. However, the open environment changes frequently. There might thus result in unavoidable and unexpected cases. A modern WfMS should evolve with increasing (or modified) mechanisms.

From the agent's viewpoints, the business process scenarios are better modeled as the interactions among the system components, users and software agents. Namely, the traditional software modules described in the WfMC workflow reference model [2] can be assisted or even replaced with software agents completely. Here we adopt the major characteristics of software agents such as mobility, autonomy and intelligence to construct, simplify and empower the whole workflow system and its applications.

The agents adopted by our system are divided into three kinds. Process Agents facilitate managing and monitoring the enactment of workflows. Artifact Agents are designed to take charge of the artifact transmission and tasks accomplishment. Person Agents assist participants in accomplishing tasks. They are also responsible for artifact representation with the Client Tool. Furthermore, our agents can perceive the changes of external environments, so that they may adjust themselves timely and adequately based on predefined rules. That is, during workflow enactment, process agents can proactively analyze the workflow to make smarter routing decision, resource allocation and task scheduling. And person agents can proactively alter and determine better artifact representation. They will also provide timely predictions and decision advices for workflow participants according to the business rules and changing environment.



In our system architecture, a process agent acts as a decentralized service executor in the distributed environment. Next, the database access of artifact agents can be deemed different that of conventional centralized one. Although the instantiation of artifact agents need to refer databases, but after successful instantiation, they will maintain these data objects and act as data carriers during workflow execution. Afterwards, no more access to databases needed. Subsequently, person agents can bring more new ideas to the workflow enactment, since now the person agent is an active object in the system. That is, conventional passive operation can be replaced with active ones, such as the work list handling and interface manipulation now can be more humanized. The person agent can actively notify the user about the oncoming events, such as the urgent task or workflow failure, and not the user operates the tools to receive the result.

The rest of this thesis would be organized as follows. Chapter 2 introduces the

background including WfMSs and recent research work on agent systems. Chapter 3 clearly describes our system architecture including the interactions among three agents and system components thoroughly. Chapter 4 explains system design with typical workflow behaviors. Chapter 5 gives some implementation issues about our system. Chapter 6 concludes with the advantages of the proposed system architecture and future works.



Chapter 2. Background

2.1 Workflow Technology and WfMS

According to the Workflow Management Coalition's (WfMC) [1] definitions, a workflow is the computerized facilitation or automation of a business process, in whole or part. And the workflow management system is a system that completely defines, manages, and performs workflows through the execution of software, whose order of execution is driven by a computer representation of the workflow logic. Inside a WfMS, one of the significant parts is workflow enactment service [2]. It may consist of one or more workflow engines in order to create, manage and execute workflow instances. Besides, it also provides various interfaces to users and applications distributed across the workflow area. These applications include process definition tools, administration or monitor tools and invoked applications etc. Figure 1 illustrates the WfMC's Workflow Reference Model and its generic components.

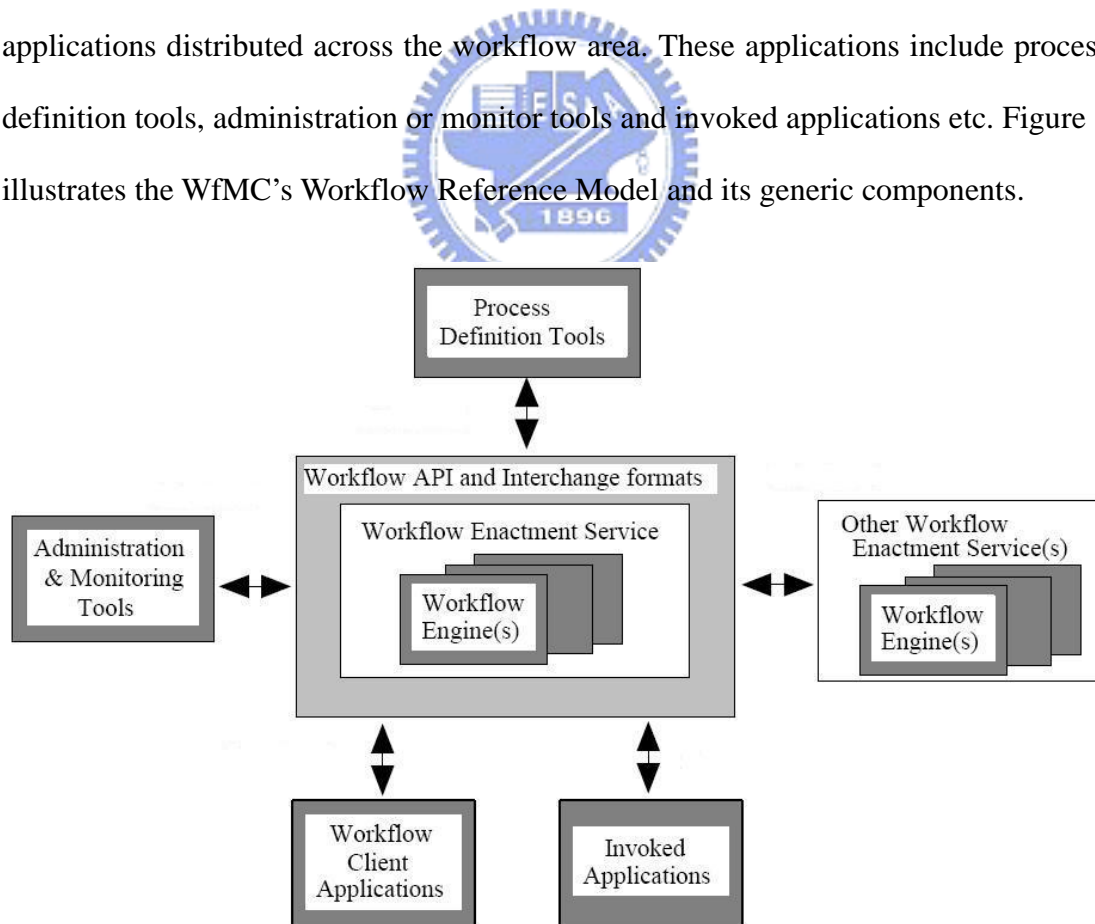


Figure 1. WfMC's Workflow Reference Model

2.2 Software Agents Technology

The idea of software agents was firstly used for the domain of artificial intelligence, and tries to use agents to simulate human behaviors. Soon, many domains of computer science adopted this concept in their research area. Jennings et al. [4] defined that an agent is a computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. Afterward, intelligent agents [5], mobile agents [16] and other new concepts of software agents were proposed. Their primary objective to study is that software agents own more power and better ways to accomplish tasks. And researchers generally agree that a so-called intelligent agent must demonstrate following properties [5] [6].

- **Autonomy**

Agents perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they have control over their own actions and internal state.

- **Reactivity**

Agents are able to perceive their environment and respond in a timely fashion to changes that occur in themselves.

- **Pro-activeness**

Agents are able to exhibit goal directed behaviors by taking the initiative where appropriate.

- **Social ability**

Agents are capable of interacting with other agents, and possibly humans.

Among those researches of intelligent agents, the BDI (Belief-Desire-Intention)

[17] theory is a very well known model. For BDI agents, the “belief, desire, intention” are said to be the internal mental states of an agent.

2.3 Related Research

Owing to the increasing shortcomings of the traditional WfMS, many researchers in the related areas have invested a lot of time in enhancing the workflow management and proposed various modified workflow management system architectures. Researches about introducing the software agent technology into the workflow management systems have been invested for years. Yuhong Yan et al. [8] got the related information about agent-oriented system architectures together and divide them into two ways. One of them is called *agent-enhanced*; its concept is similar to treat agents as tools for workflow automation. That is, they could only do something helpful for running workflows, such as acting as middleware for invoked applications. Because the objective of these agents is automation, they don't need to interact and communicate with each other. There always exists a workflow engine monitoring and controlling all these agents' behaviors. So these agents are more like a piece of ordinary software.

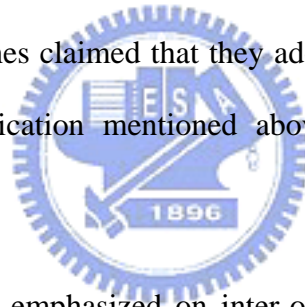
The other way is called *agent-based*; an agent based system means one in which the key abstraction used is that of an agent. At this time, software agents replace the behaviors and actions of business processes in the whole system. They not only did something helpful for running workflows, but also fulfilled the entire business process of workflow enactment. That is, the process logic is embedded in these agents, and they take full responsibilities of this WfMS. In other words, it means that they have all the means to analyze, automate, integrate and inspect workflows. Every agent is an independent individual and would have the ability to interact and communicate with

each other to accomplish the task. The software agent's high-level capabilities, such as learning, negotiation and mobility etc., may be also put into the system architecture considerations. Although some abilities of them are not mature today, but it is obvious to see that the WfMS with these high-level abilities would be more flexible and powerful.

In this thesis, we introduce a software agent technology for WfMSs and describe the *agent-based* WfMS architecture to cope with the fast increments on open systems. Firstly, an agent itself is an active object which runs in a loosely coupled distributed computing environment. Adding more agents and using the corporation mechanism of agents can solve the problems of scalability with large number of participant involvement. Secondly, when integrating with other applications and systems, the communication mechanisms and interfaces of agents provide more flexibility and extensibility than API calls. Thirdly, the interactions within a workflow may be much complicated and happen frequently. By making use of the agent communication languages such as FIPA ACL [18], SOAP [19] and KQML [20], the standards can be normalized with various interactions in a general form. Moreover, the autonomy, intelligence and pro-activeness of agents can provide personalized interfaces and timely help for workflow participants. Fourthly, the autonomy characteristic of agents makes an agent able to execute its own tasks or fulfill activities automatically.

Many research projects of agent orient workflow system have been presented and tried to fully use the agent capabilities in every dimension. One example is the TRP support environment (TSE) [21] proposed by Andersen Consulting, which has a central workflow authoritarian agent acting as a workflow engine and other authoritarian agents, actor agents etc. Its goal is to enable active collaborative work among participants workings on the TRP's component based software engineering

environment. Another example is Advanced Decision Environment for Process Tasks (ADEPT) [7] project proposed by British Telecom Lab, which focuses on enhancing the supply chain management. The system consists of multiple software agents that concurrently negotiate an agreement on how resources should be assigned to support a business process. Later, Weishuai Yang et al. [9] proposed an agent-enhanced workflow (AEWF) model, which aims at enhancing the interoperability of workflow management engines. It is based on the BDI agent model, and tries to use mobile agents to increase the capability of workflow interoperability. Subsequently, Leangzhao Zeng et al. [10] proposed an approach that combines agents with workflows to effectively integrate cross-enterprise workflows. It is in order to support virtual enterprise and business-to-business e-commerce. Besides, although some projects among these researches claimed that they adopt the *agent-based* viewpoints, but according to the classification mentioned above, these projects are actually *agent-enhanced* systems.



These researches are all emphasized on inter-organization workflows, such as supply chain management and business-to-business. But they don't really consider the significant problems happening to basic workflow definitions, such as the ones of scalability, extensibility, flexibility and adaptability mentioned in Chapter 1. Although, in order to solving conventional business-to-business (B2B) or business-to-customer (B2C) workflow behaviors, a sound service orient architecture has been proposed. However, these workflows are almost very simple and stationary, they don't even need to have a role model. A workflow becomes complicated just because of human involvement, such as cancel and countersignature. So here we primarily aim at presenting an *agent-based* system architecture for primitive workflows. And how to use agents to realize typical workflow behaviors such as start, stop, or enactment will

be described thoroughly in Chapter 4. On the other hand, these proposed systems still have fixed artifact representations only, which should be represented dynamically and adequately for varied process executors from current workflow's viewpoints. Besides, the increasing facilitation not only occurs in inter-organization workflows, but also happens to workflows inside an organization. Thereupon, all of them should be solved effectively and flexibly in a modern workflow management system.



Chapter 3. System Architecture

3.1 Overview

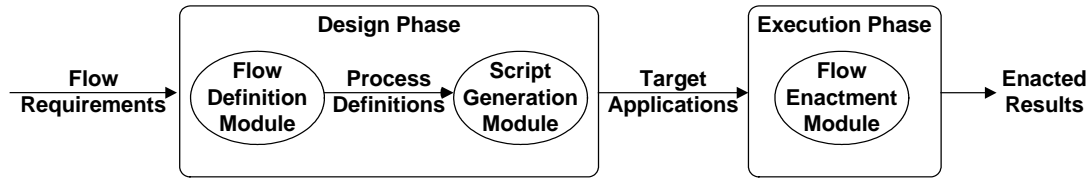


Figure 2. System Architecture Overview

As shown in Figure 2, our system can be applied in the design and execution two phases. For the design phase, the system contains the flow definition module and script generation module. The former facilitates the definitions of workflow processes thoroughly and the latter is used to compile the process definitions to agent script files (i.e. the target applications shown in the figure). These script sources are stored in the rear-end Script Repository. During the execution phase (i.e. the workflows are enacted), the flow enactment module is used to load the related script files and documents from the Script Repository and databases. Then, the agent platform could instantiate the corresponding agents to realize and accomplish the entire workflow. Figure 2 just simply introduces our system architecture. The detailed designs of our system for these two phases will be described clearly in the later two sections.

3.2 System Architecture — Design Phase

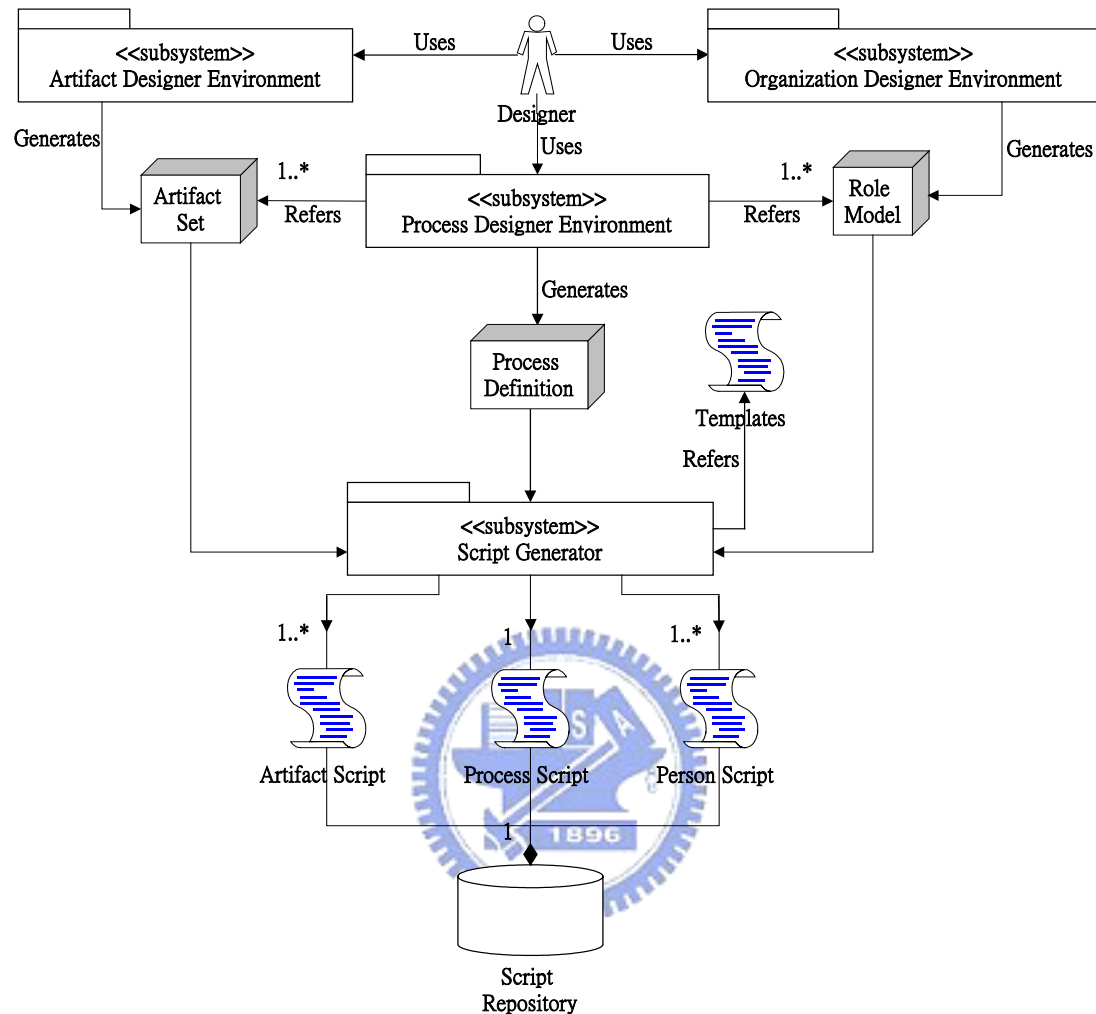


Figure 3. System Architecture — Design Phase

The activities in the design phase of our entire system are shown in Figure 3. The top half of the figure is primarily referred to the PLAN (Process LANGUAGE) Model [12] [13] and the system design of AgentFlow [11]. In this thesis, we extend the PLAN model and divide it into three sub-models called process, role and artifact individually. The process sub-model describes the functional and behavioral perspectives of a software process (i.e. workflow). It includes the methods of workflow initiation, completion and termination. It also models various workflow behaviors during workflow enactment. Besides, we add the time constraints for design

into the process sub-model: workflow designers can destine the time about the accomplishment or initiation of workflows and replacement of newer version workflows. The role sub-model illustrates the organizational perspective of a software process. It consists of the organization hierarchy, personnel property and communication among the roles. The artifact sub-model illustrates the informational perspective of a software process. It demonstrates the reference materials such as forms, documents and programs and the lifecycle (described in states) of an artifact. An artifact is defined as a product referred or generated by a workflow.

There are also three corresponding tools in our system: the Process Designer Environment, Organization Designer Environment and Artifact Designer Environment. Process Designer Environment facilitates the definitions of the states, paths, trigger events, rules, constraints, and documents allocation etc. of workflows. Organization Designer Environment provides tools, which can represent organization chart hierarchically, for defining the organization duties for the departments and roles inside a company. Artifact Designer Environment helps designers to arrange and edit the documents or e-forms of the workflow by collocating with off-the-shelf GUI components, attribute editors and script editors. By using these graphics modules and visualization tools, workflow designers can specify and design a workflow more flexibly and efficiently.

When the definitions of the entire workflow are completed, supporting environment would gather the documents, e-forms and organizational information and stockpile them into the Role Model and Artifact Set individually. In addition, it also groups the information about the workflow enactment, such as rules, events, constraints and allocations into the Process Definition. These three models are all described with the XML format for its easy understanding and compiling. With XML,

our system is more convenient and feasible to integrate external applications and agent systems, or to interchange process definitions with other process model languages such as XPDL (XML Process Definition Language) [3] [15] advised by WfMC. Subsequently, the Script Generator is responsible for analyzing these workflow definitions and then generating the agent script files, which are stockpiled into the Script Repository. The generation is done by referring to the pre-designed templates and using respective components. And the Script Generator and Templates are constructed according to the agent platform adopted. Hence, the information about the workflow enactment is independent of the agent platform. These modules designed increase the extensibility of our system. Therefore, an application workflow designed inside this phase is translated into executable agent script files.

3.3 System Architecture — Execution Phase

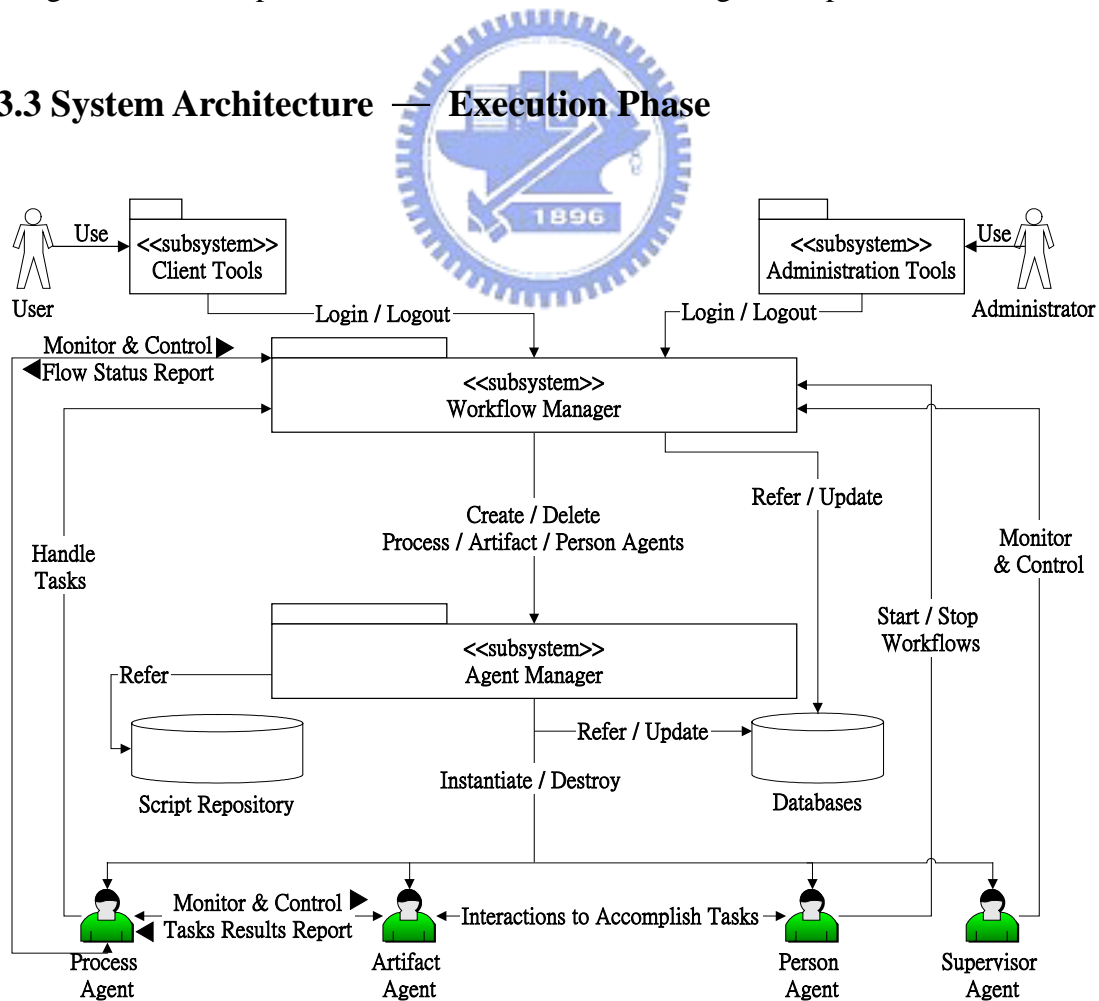


Figure 4. System Architecture — Execution Phase

The execution phase inside our system is shown in Figure 4. The primary four components in an application workflow system include the Person Agent, Artifact Agent, Process Agent and Workflow Manager and they will collaborate on the workflow enactment services. There are still some important components, such as the Script Repository for storing scripts and the Agent Manager for instantiating, destroying and monitoring agents also shown in the figure. The primary reason for separating the agent manager from the workflow manager is for flexibility. That is, the agent manager is constructed dependent on the agent platform adopted. It provides not only the services, such as naming, instantiation, destruction and location etc., but also a common set of API calls for the workflow manager.

Operators, which include users and administrators, use the Client Tool or Administration Tool to login our system. There are two implements provided for the interface manipulation and states monitor respectively. The former includes system logging, workflows starting and task handlings for operators. The latter provides the states of current enacted workflows and handling function such as restarting or stopping the running workflows. Besides, a supervisor can monitor and manage all running workflows by the Administration Tools, but an employee can only watch the workflows he instantiates or handles. In the next subsections, we will describe the person agent, artifact agent, process agent and workflow manager thoroughly with their designs and cooperation.

3.3.1 Person Agent

A person agent is an intelligent agent that acts as an interface between the system and the user. It represents the user interface function in a workflow system and facilitates related workflow operations. A person agent is instantiated by the agent

manager when the user is online, and destroyed when the user leaves. On the other hand, the Supervisor Agent in Figure 4 is actually similar to the person agent; it is instantiated by the agent manager when the administrator logs our system by the Administration Tool. Such a naming method is adopted for reducing the confusion possibility. Figure 5 below depicts the instantiation of a person agent.

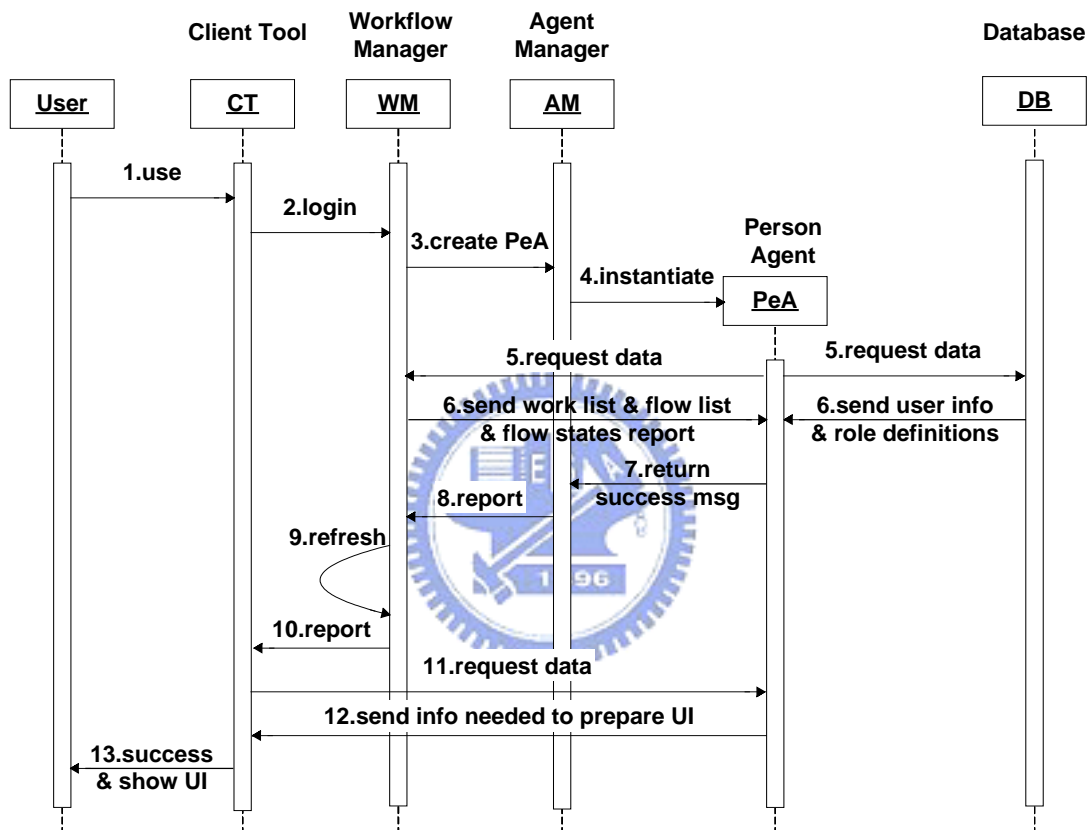


Figure 5. Instantiation of a Person Agent

The profile of a person agent is based on the Role Model. The information retrieved for a person agent includes contents and capabilities when the person agent is instantiated. The information is described below:

1. Contents

- Internal-record

It contains an agent-id and the information about the corresponding user, which includes name, phone, email and customized configurations etc. The

information is stored in the databases.

- Role

The role represents a position where a user occupies in the organization. The user/role relationships are also defined in the Role Model.

2. Capabilities

- Role-specific user interface

A person agent provides a role-specific interface according to the customized configurations of a corresponding role. It is decided by reasoning the related rules in a company's policy.

- Work list management

A work list is a set of tasks derived from the workflow manager and needs to be accomplished by a user. The tasks may be properly scheduled, i.e., can be sorted by priority, resource, deliver time or something else based on the user's requests. With this viewpoint, a person agent can act as a work list handler. When a user takes a task, the corresponding person agent can provide a proper user interface such as e-forms [11], which are carried by artifact agents. At the same time, the person agent communicates with artifact agents to report the user's decisions. Once the user is getting out of line, the corresponding person agent would send the logout messages to all artifact agents in its work list.

- Flow list management

A flow list is a set of workflows that can be worked by the user. It can be got from the workflow manager. And it is also dependent on the user's role(s) and related workflow enactment constraints. When a user wants to enact a workflow, he picks it from the flow list in the user interface and sends an instantiation request to the workflow manager through his corresponding person agent. The workflow manager then asks the agent manager to

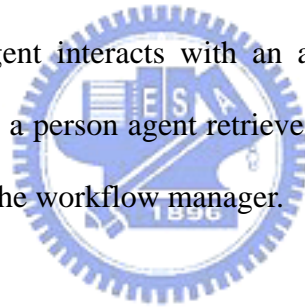
instantiate a respective process agent.

- **Flow states report management**

The report records the states of each workflow which has been worked by the user. The workflow states can be got from the workflow manager. The report includes the flow state such as finish, waiting, running, suspended. With the aids of reports, a user can monitor the workflow progress he worked. Besides, he also can cancel a workflow which has not finished.

- **Communication**

The communication targets are the process agent, artifact agent and workflow manager. Firstly, a person agent needs to interact with a process agent to retrieve the necessary data for detailed workflow progress representation. Secondly, a person agent interacts with an artifact agent for user interface representations. Lastly, a person agent retrieves the relevant data such as work list and flow list from the workflow manager.



3.3.2 Artifact Agent

An artifact agent is a mobile agent that acts as a data carrier, which migrates between sites and interacts with its process agent and person agents to accomplish the task. An artifact agent is instantiated by the agent manager when the process agent prepares to execute the tasks of a workflow, and destroyed when all tasks are accomplished. There may be more than one artifact agent, and the number of artifact agents of a workflow depends on the route. If there is a parallel (spilt) node in the route, the process agent has to duplicate enough artifact agents to complete the route. Besides, in the join node, the process agent will collect all necessary data to determine the next node and maintain the number of artifact agents to fit the route.

Figure 6 below depicts the lifecycle of an artifact agent clearly. And an artifact agent can repeat the actions numbered 8 to 10 before being destroyed. Furthermore, before receiving the request of a person agent, an artifact agent usually resides at the site it is generated for being together with the process agent, or locates itself in the task list owned by the workflow manager.

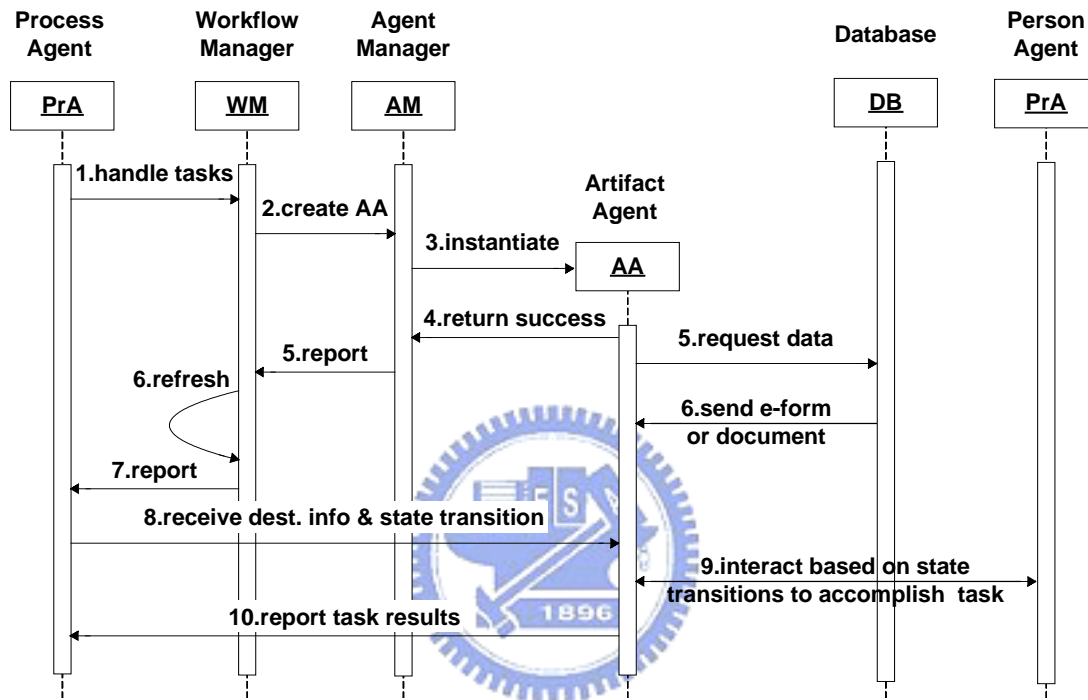


Figure 6. Lifecycle of an Artifact Agent

The profile of an artifact agent is retrieved from the Artifact Set and/or databases when the artifact agent is instantiated. The information contained in the profile can be divided into two parts:

1. Contents

- Internal-record

It contains an agent-id and the carried artifacts such as e-forms, documents or other data, which are necessary to accomplish the task. The data can be got from the databases and Artifact Set.

- State transition

It contains the rules to interact with person agents to accomplish the task. The rules can be obtained from its process agent. Note that the information about the lifecycle of an artifact agent is kept in its process agent. The state transition here is also helpful to the artifact representation.

2. Capabilities

- Communication

The communication targets are its process agent, person agents and the workflow manager. Firstly, an artifact agent needs to report task results and get the next route data from its process agent. Secondly, an artifact agent provides necessary artifacts to a person agent for user interface representations and receives the user's decisions from his person agent. Lastly, an artifact agent asks the workflow manager to report an online agent-id satisfied the conditions for executing the task.

- Mobility

An artifact agent could migrate between the sites indicated by its process agent.

- Offline handling

If the target person agent does not exist (i.e. user is offline), an artifact agent returns and reports to its process agent. If there exists another choice, the artifact agent would migrate to the new destination accordingly. Otherwise, the artifact agent will queue itself into the task list of the workflow manager.

3.3.3 Process Agent

In our model, a process agent is an intelligent agent that acts as a workflow enactment service executor in the traditional WfMS. For example, a process agent is instantiated by the agent manager when a workflow is enacted by the user or the

system, and destroyed when the workflow is accomplished or canceled. And the detailed sequence diagram of the instantiation of a process agent will be described in Chapter 4.

The profile of a process agent is based on the Process Definition when the process agent is instantiated. The information contained in the profile can be divided into two parts:

1. Contents

- Internal-record

It contains an agent-id and the workflow data including name, description, starter, state time and due time etc. These data can be got primarily from the Process Definition.

- Rules

The rules contain the information needed to enact the workflow respectively. The information can be obtained from the Process Definition.



2. Capabilities

- Artifact agent management

A process agent can instantiate and maintain enough artifact agents to work along the route. In the meanwhile, the process agent will monitor and interact with its artifact agents. When the workflow associated with an artifact agent is accomplished or canceled, the process agent then destroys the agent.

- Routing decision

A process agent can determine the routing paths based on the internal predefined rules, previous task results and by perceiving related external environment changes.

- Communication

The communication targets are the workflow manager, its artifact agent and person agent. Firstly, a process agent reports the workflow states to the workflow manager after its artifact agent(s) migrates to next node. Secondly, a process agent gives the next route data including roles and constraints etc., and state transition to its artifact agent according to current workflow state, and receives enacted task results from its artifact agent. Lastly, a process agent needs to interact with a person agent for preparing detailed workflow execution progress shown on the client tool.

3.3.4 Workflow Manager

The workflow manager, as implied by the name, is a component that manages all executable workflows in our system. It also maintains records of all running workflow instances in the system and updates them persistently. Besides, both the instantiation and destruction of process agents, artifact agents or person agents rely on the instructions of the workflow manager. Figure 7 below depicts the interactions of the workflow manager with other system components clearly.

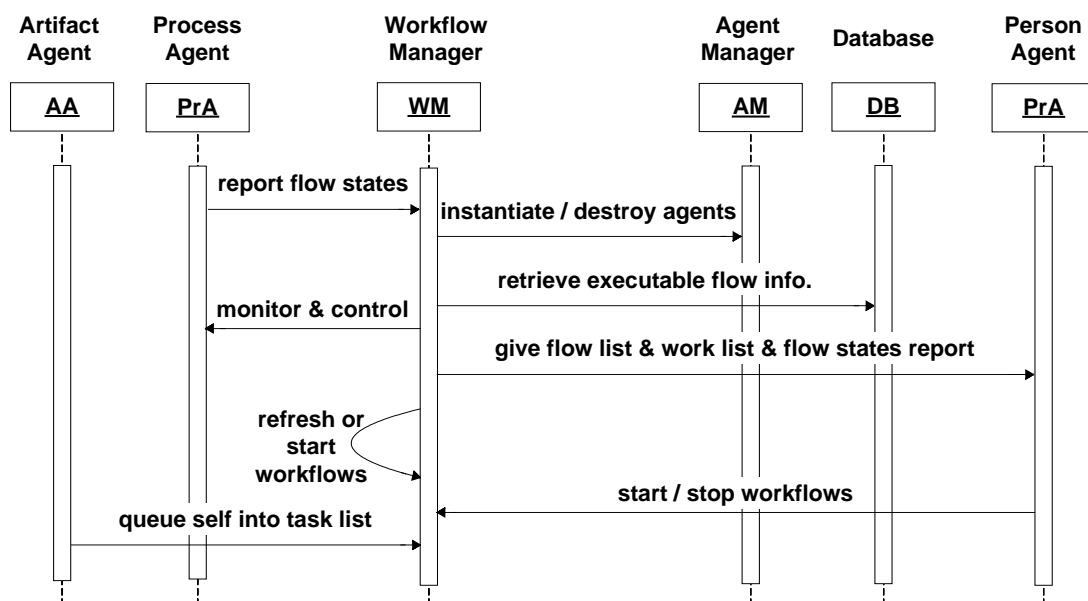


Figure 7. Interactions of the Workflow Manager

The profile of the workflow manager is loaded from the database when the system is started and would be persistently updated. It can be divided into two parts:

1. Contents

- Executable flow

It contains all available workflow definition for enactment in the system currently. The data are stored in the database.

- Flow state

It contains the states of all enacted workflow instances (i.e. process agents) in the system currently. The data are obtained from the reports of process agents.

- Task list

It contains the work list of all users in the system, i.e., the relationship of artifact agents and person agents.

- Agent list

It maintains a record of agent-id of all running artifact agents and person agents.



2. Capabilities

- Process agent management

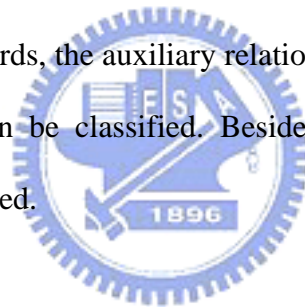
All the instantiation and destruction of workflows are monitored and maintained by the workflow manager. After completing the work in one site, the process agent reports its workflow state to the workflow manager. Besides, when a user's request for enacting or stopping a workflow is accepted, the workflow manager is responsible for instantiating or destroying the corresponding process agent. On the other hand, the workflow manager can also enact a scheduled workflow, where the timing constraints decide whether to continue the workflow itself or not.

- Task list management

There exist two situations for updating the task list. When the designated person agent of an artifact agent is offline, the artifact agent will queue itself into the task list. When the user appears to logout the system, those artifact agents, whose tasks have not completed yet, will queue themselves into the task list. In both cases the workflow manager will gather the agent-id of these artifact agents when they are queued into the task list.

- Agent list management

Both the instantiation and destruction of artifact agents and person agents are settled by the workflow manager. That is, requests from the process agent or client tool will be propagated to the agent manager by the workflow manager. So the workflow manager can keep and maintain the lists of artifact agents and person agents. Afterwards, the auxiliary relationship between the process agent and artifact agents can be classified. Besides, the problems of re-login or multi-login can be solved.



- Communication

The communication targets can be the process agent, person agent, agent manager and artifact agent. Firstly, the workflow manager gets flow states from reports of process agents and persistently updates them. Secondly, the workflow manager gives flow list, work list and flow states report to a person agent. Thirdly, the workflow manager would ask the agent manager to instantiate or destroy agents. Lastly, the workflow manager reports the satisfied agent-id to an artifact agent for executing the task.

3.4 An Example

The contexts mentioned above are emphasized on the description of individual component, and lack of global illustration of the actual works among these

components during the workflow execution. Here we use a simple ask-for-leave workflow as the scenario example to describe the handing of a workflow instance being enacted.

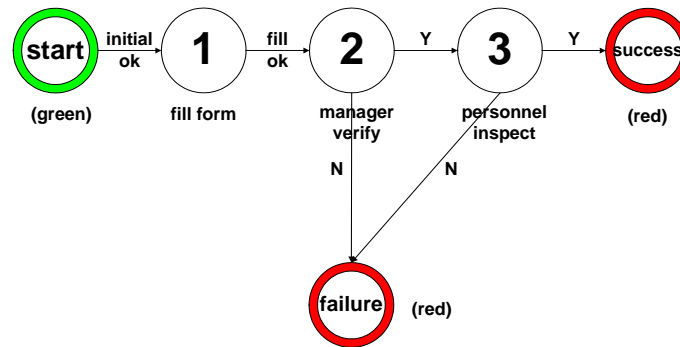


Figure 8. A Simple Model of Ask-for-leave Workflow

The ask-for-leave workflow has three process nodes as shown in Figure 8, and the green-circled node represents the start state and the red-circled ones represent the final states. The remaining three nodes according to the sequence number are the filling form, manager verification and personnel inspection. Assume there is an employee called Aaron who wants to ask for leave. Following operations represent the sequence to create a workflow instance in our system. Firstly, Aaron must login our system to use the client tool, and his corresponding person agent (we call it as PeA_Aaron in the following context) is instantiated in the agent platform after password authentication. Secondly, he enacts the ask-for-leave workflow from the flow list provided by the client tools. Once PeA_Aaron receives the enactment instruction, it asks the workflow manager to create the corresponding workflow instance. Subsequently, the process agent (we call it as PrA_leave in the following context) will be instantiated to take charge of the enactment of the ask-for-leave workflow. So far, an ask-for-leave workflow instance has been created successfully in our system.

When PrA_leave is successfully instantiated, it begins to analyze the workflow definition according to the predefined rules and detects that the workflow is in the start node now. Then it prepares to handle the first task, i.e., the filling form stage. First of all, it asks the workflow manager to create an artifact agent to carry the required workflow data. When the artifact agent (we call it as AA_leave in the following context) is instantiated successfully, it then retrieves the data objects from the database. In this example, these data objects compose an e-form. According to the workflow definition, the operator of the filling form task is the workflow starter, i.e., Aaron, and Aaron's person-id is sent to AA_leave by PrA_leave. Next, AA_leave asks the workflow manager to report corresponding agent-id. If Aaron is still online, then PeA_Aaron will be notified about the new item added to the work list and ask the client tool to show the change.



Assume Aaron is online and picks up the ask-for-leave to execute. Simultaneously, PeA_Aaron notifies AA_leave to migrate to this site and interacts to decide how the data objects should be shown. In this stage, Aaron needs only the first part of the e-form, i.e., the verification, and inspection parts need not to be shown to Aaron now. So, the client tool receives the e-form and shows it on the screen for Aaron to fill. After Aaron completes the filling, AA_leave migrates back to report the new state to PrA_leave. Next, PrA_leave analyzes the state and determines the next route. According to the workflow definition, the next stage is the manager verification and its target operator is the manager of the form writer. So, AA_leave receives the corresponding role-id from PrA_leave and then asks the workflow manager to report an agent-id of the person agent of Aaron's Manager. Assume that there are two candidates called Bob and Cindy, but Cindy is offline currently. Then AA_leave will receive the agent-id of Bob's person agent.

Again, there will be a new item called ask-for-leave added on Bob's work list. When Bob executes this task, AA_leave receives the notice of Bob's person agent and migrates to this site to interact for adequate artifact representation. Now the first and second parts of the e-form need to be shown (i.e. the inspection part not shown here) and the first part is now read-only. When Bob completes the task, no matter what the decision is, AA_leave migrates back to report the result to PrA_leave. If the decision is "rejection", according to the workflow definition, PrA_leave detects the flow is going to complete. It sends the rejection result message to the workflow manager to notify the flow is going to complete. Subsequently, the workflow manager notifies PeA_Aaron the rejection result and asks the agent manager to destroy AA_leave and PrA_leave. Assume Bob's decision is "acceptance". According to the workflow definition, PrA_leave confirms that the route can be proceeded and the next stage is the personnel inspection. The target operator is restricted to the members of personnel department only. And AA_leave receives the corresponding role-id and asks the workflow manager to report an appropriate agent-id. Except the artifact representation, the following sequences are similar to the previous stage mentioned above. Now the e-form shown to the operator includes all three parts and the first two parts cannot be modified.

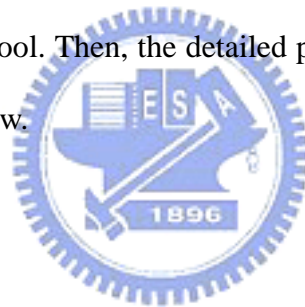
Assume the task of personnel inspection is completed, and the decision is "acceptance". PrA_leave receives the result and detects the flow is going to complete. Then, it sends the acceptance result to the workflow manager and notifies the latter that flow is going to complete. Subsequently, the workflow manager notifies PeA_Aaron about the acceptance result and then asks the agent manager to destroy AA_leave and PrA_leave. Finally, Aaron gets the result of the ask-for-leave workflow and the whole execution of an ask-for-leave workflow instance terminates here.

Chapter 4. Flow Behaviors

In this chapter, we will present the interactions among the Process Agent, Artifact Agent, Person Agent, and Workflow Manager with typical workflow behaviors, and use corresponding sequence diagrams to illustrate their responsibilities and relationships thoroughly in our system.

4.1 Workflow Enactment

There are two ways to enact a workflow in our system. They are called automatic and manual. The former means that our system enacts a scheduled workflow automatically at the expected time; the latter means that a workflow can be enacted by a user with aids of the client tool. Then, the detailed processes of these two ways will be described individually below.



4.1.1 Automatic

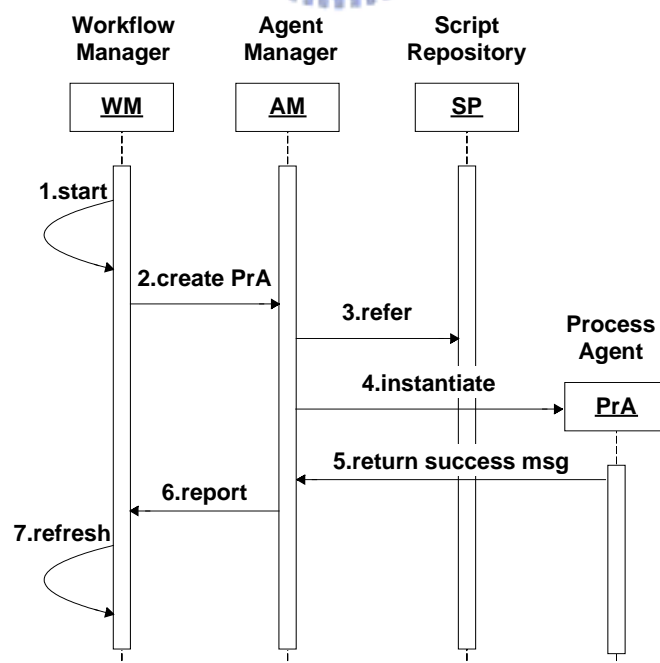


Figure 9. Interactions of Enacting a Workflow Automatically

As shown in Figure 9, when the workflow manager is noticed to enact a scheduled workflow, it asks the agent manager to create the related process agent. The agent manager then retrieves necessary script files from the Script Repository, instantiates the corresponding process agent, and reports results to the workflow manager. Next, the workflow manager refreshes its flow states and stores the information about the starter (now is the system), start-up time and states to the database simultaneously. Moreover, the workflow manager keeps monitoring the running workflow instance.

4.1.2 Manual

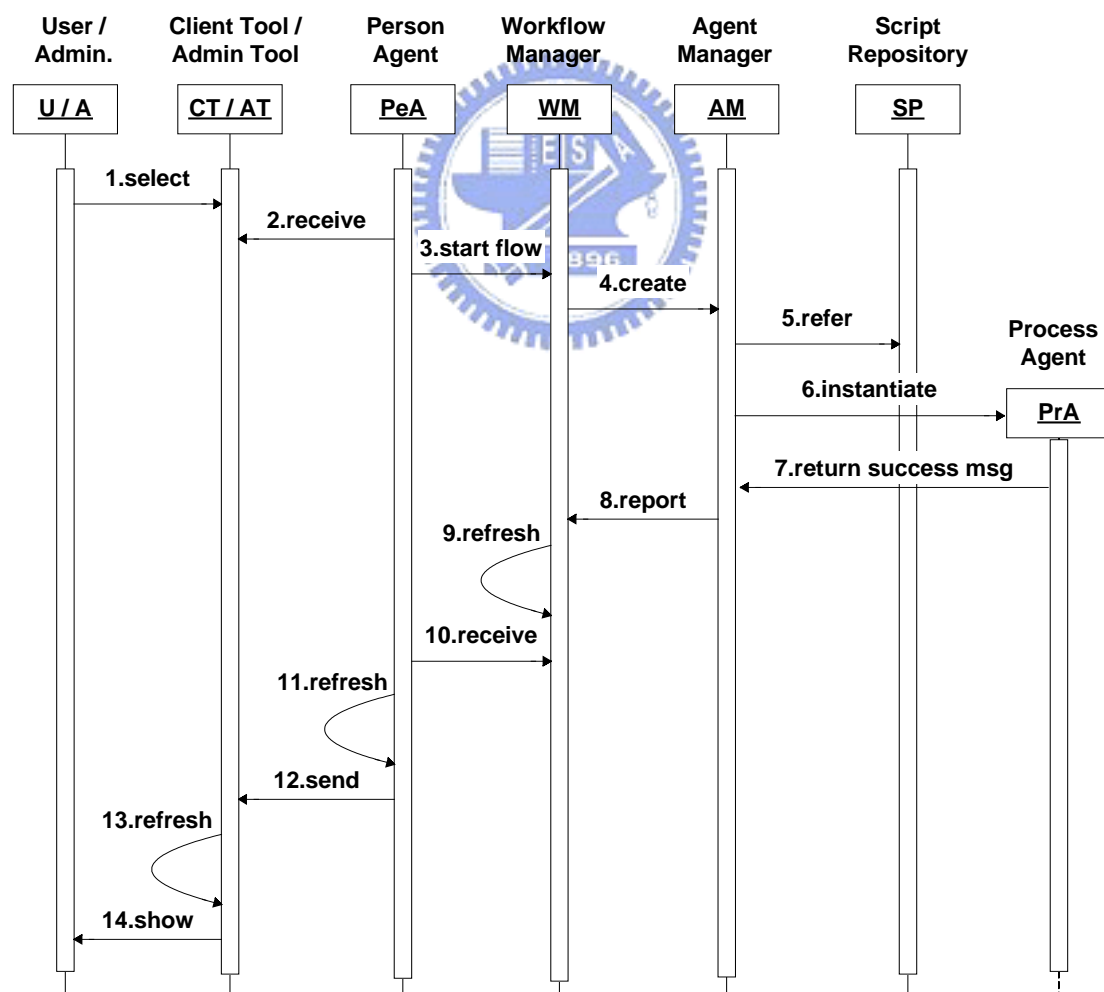


Figure 10. Interactions of Enacting a Workflow Manually

As shown in Figure 10, with aids of the client tool, a user knows the workflows worked by him. When a workflow was selected and enacted by a user, the corresponding person agent will receive the instruction through the client tool and inform the workflow manager to create the corresponding process agent. Besides, until the step where workflow manager refreshes its flow states, the subsequent actions are similar to the automatic one aforementioned. Then, the person agent receives the modifications of the flow states and updates its flow states report simultaneously. Afterwards, the client tool receives the messages from the person agent and refreshes its views to notify the user that the workflow is started successfully.

4.2 Workflow Termination

There are two ways for the termination of a workflow in our system. One workflow is terminated by external force and the other is to complete automatically. One example for the former is done by human involvements, such as canceling a workflow; the latter means that a workflow completes the work successfully. The detailed designs of these two ways would be described individually below.

4.2.1 Forced by External Power

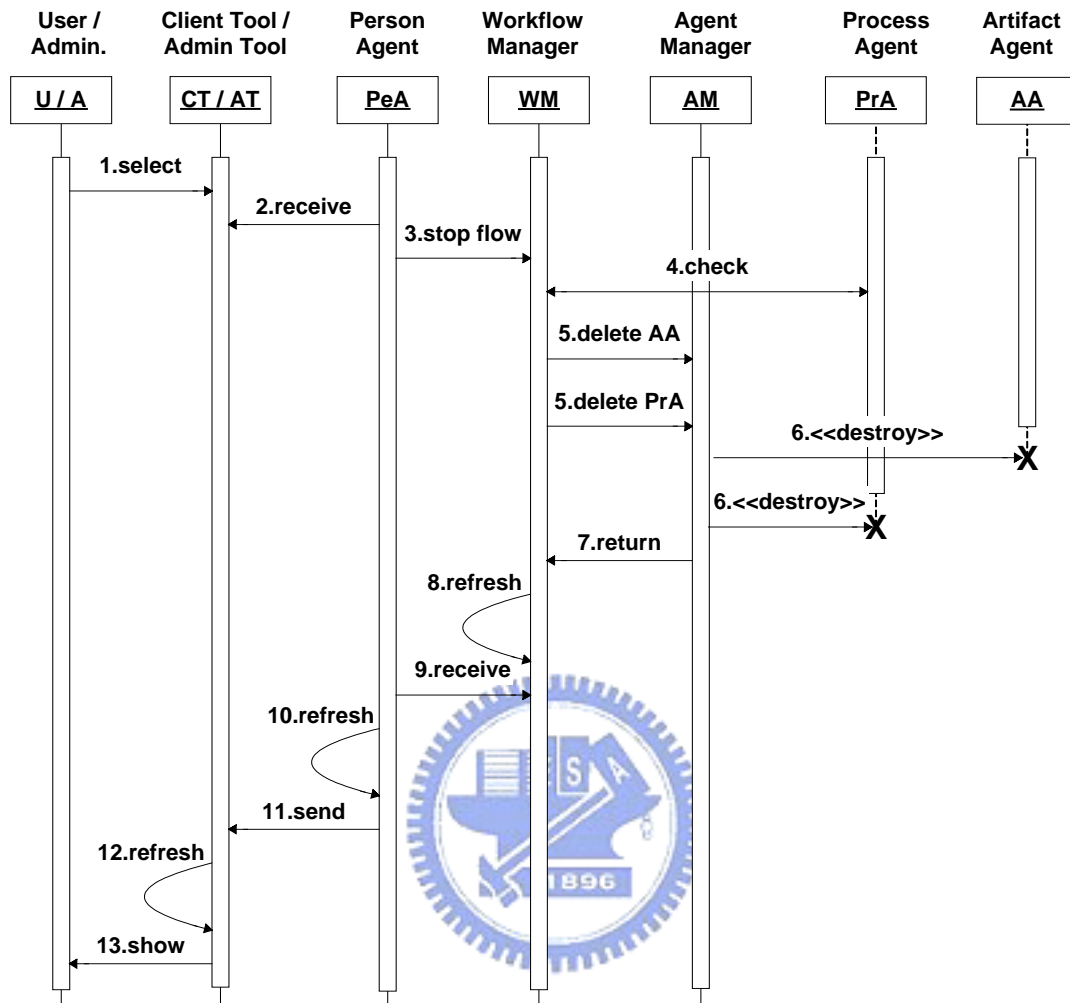


Figure 11. Interactions of Stopping a Workflow

As shown in Figure 11, with aids of the client tool, a user wants to issue an exceptional command to cancel a running workflow. When the person agent receives the cancel instruction, it informs the workflow manager to do the work. A successful way is followed as: in case the user is authorized the power, the workflow manager will check the workflow state with the corresponding process agent first. Then, the workflow manager asks the agent manager to destroy this process agent and its artifact agents. No matter the termination request succeeds, the person agent will send back a notice of result.

4.2.2 Complete

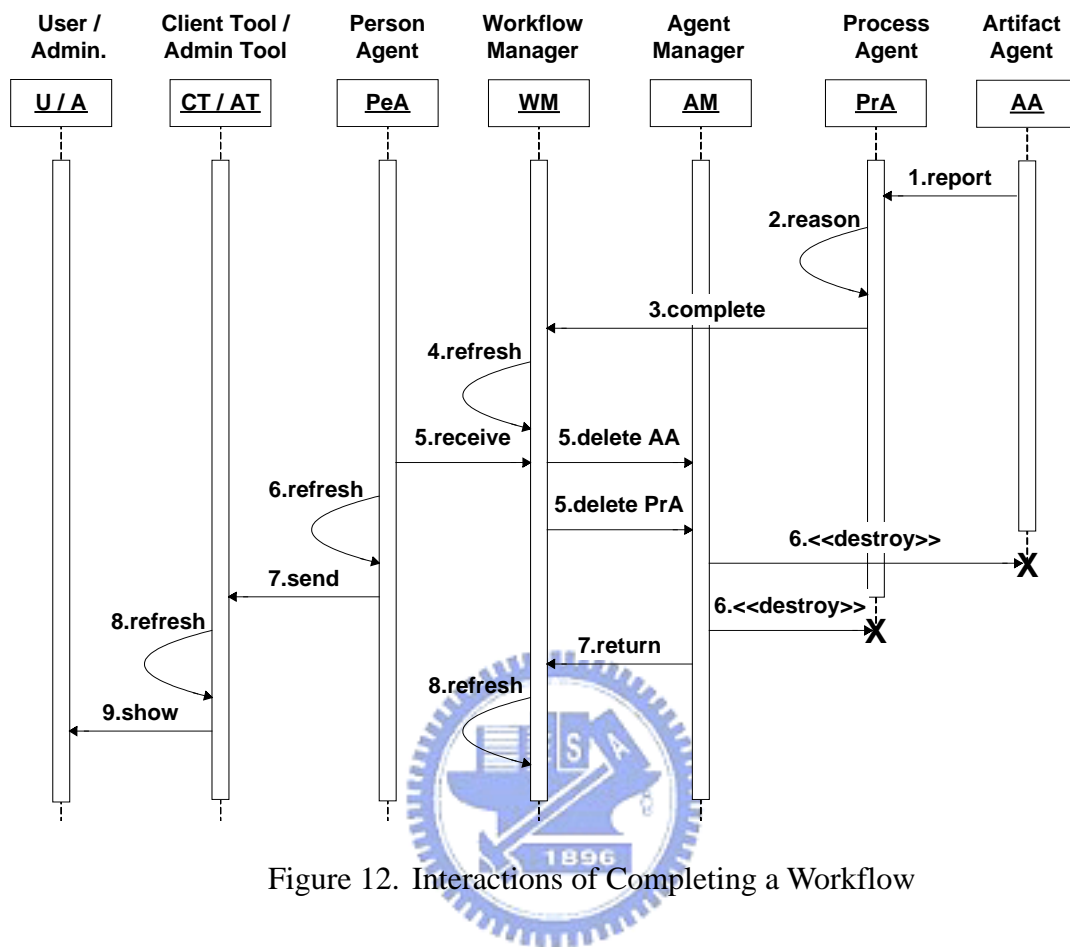


Figure 12. Interactions of Completing a Workflow

As shown in Figure 12, a process agent analyzes the results reported by artifact agent(s) and detects that the workflow has completed. It then informs the workflow manager the termination message. Next, the workflow manager asks the agent manager to delete the corresponding process and artifact agents. Besides, the workflow starter shall receive the termination reported by his person agent.

4.3 The Detailed Interactions of Workflow Enactment

In our system, the enactment of a workflow can be primarily fulfilled by the interactions of three agents. Tasks are accomplished based on interactions of the person agents and artifact agents. Basically, a task is similar to an activity, which is an atomic work unit to be completed in a workflow. On the other hand, the artifact

routing and task dispatching can be determined and monitored by the process agent, which owns the flow graph of each artifact. Artifact agents will obey these routing changes. Then, the detailed designs of these behaviors will be described individually below.

4.3.1 Accomplish a Task

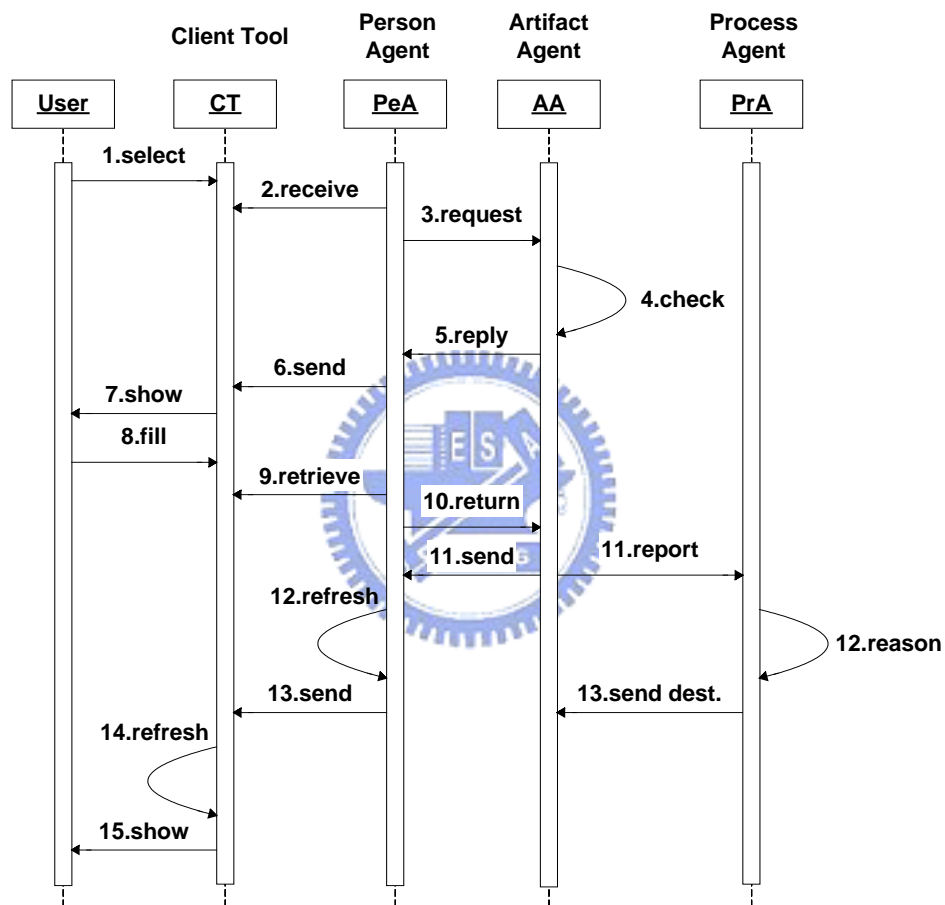


Figure 13. Interactions of Accomplishing a Task

Figure 13 depicts the running sequence of agents when the agents collaborate with each other to accomplish a task. First of all, the client tool shows the work list for a user and the work list is obtained from the person agent. A user can select a task from the list to perform. Then, the person agent receives the instruction from the client tool, and sends requests to the corresponding artifact agent for interactions.

Once receiving the request, the artifact agent migrates to this site, and starts to communicate with the person agent. Among the interoperations, the artifact agent determines the required data objects for manipulation, and sends them to the person agent. The person agent then undertakes to compose an adequate e-form and layout, and ask the client tool to layout. Now far the user can manipulate these artifact data.

Subsequently, the person agent will retrieve the filled e-form and send it back to the artifact agent for validation. If artifact is validated, the artifact agent sends the accomplishment message to the person agent, and then migrates back to the site where the process agent resides for reporting task results. Simultaneously, the person agent updates the work list of the user, and finally shows them to the client tool. A cycle of accomplishing an essential task is now complete.

4.3.2 Split a Task

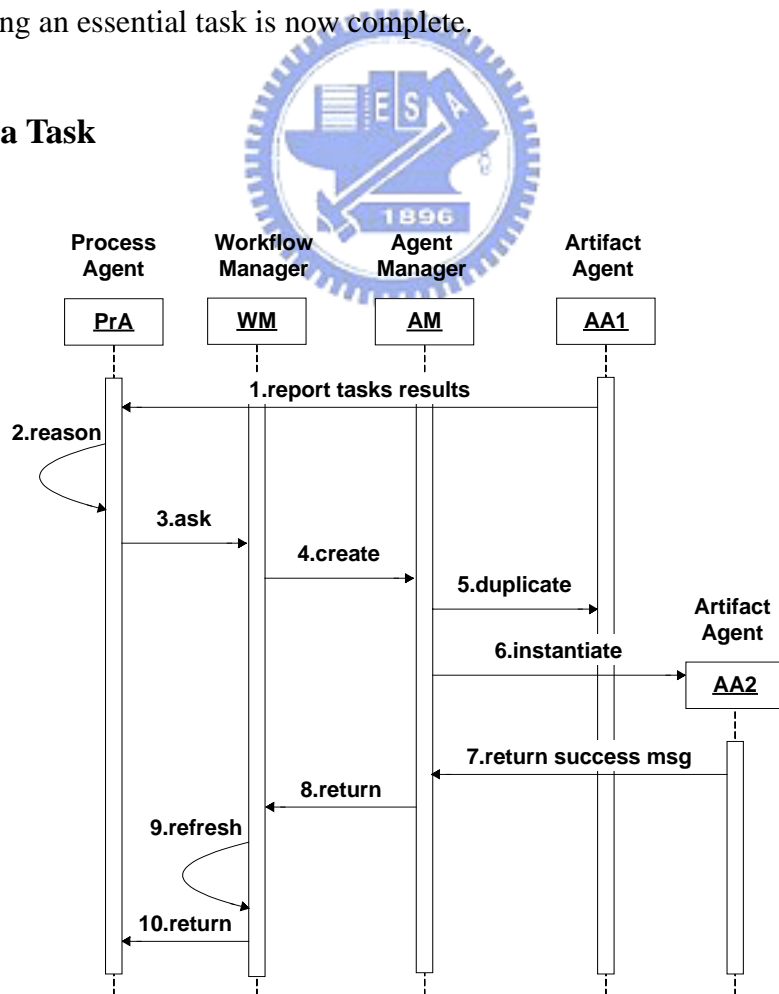


Figure 14. Interactions of Tasks Splitting

A general sequence route can be realized by repeating the actions depicted in Figure 13. And Figure 14 depicts the running sequence of agents and managers when they collaborate with each other to fulfill a parallel node in the route. First of all, the process agent keeps analyzing the results reported by the artifact agent. When a process agent detects the route is to be split (i.e. one to many), it determines to create more artifact agents to fit the route. The process agent then asks the workflow manager to create the artifact agents needed. There are two steps for the agent manager to instantiate the artifact agents. Firstly, it captures and saves the current states and data objects of the running artifact agent. Secondly, it instantiates new artifact agents with the same states and data objects, so that each new artifact agent is identical to the original one, except for the agent-id. A generated artifact agent is associated with a route, and a cycle of splitting an essential task is now complete.



4.3.3 Merge Tasks

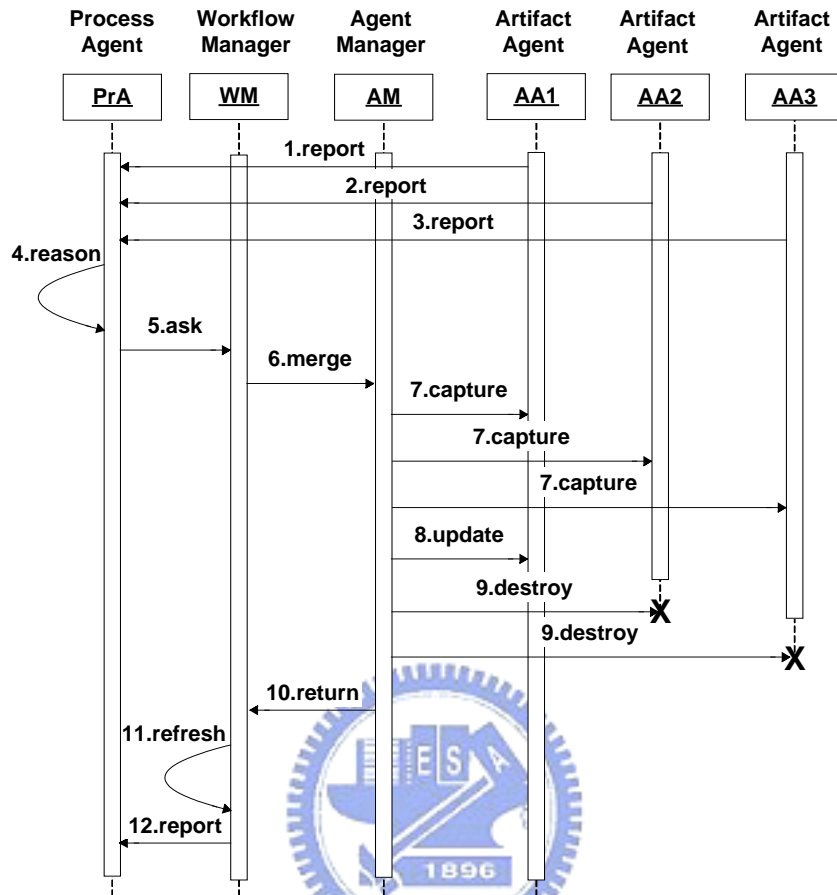


Figure 15. Interactions of Tasks Merging

A join node usually appears in the following workflows after a parallel node. Figure 15 depicts the running sequence of agents and managers when they collaborate with each other to fulfill a join node in the route. First of all, a process agent keeps analyzing the results reported by these artifact agents and checking for the satisfaction of merging conditions. If the conditions are satisfied, the process agent asks the workflow manager to handle the merge case. Again the workflow manager propagates the request to the agent manager. The detailed actions of the merge instruction include: firstly, the agent manager captures all the states and data objects of artifact agents being merged. Secondly, it chooses an arbitrary artifact agent and updates its contents with the merged results. Thirdly, it destroys other remaining artifact agents. So far the

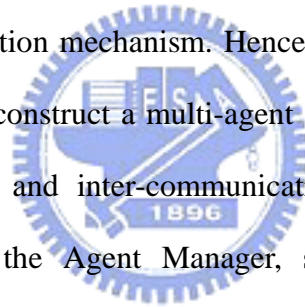
number of effective artifact agents of the process agent is reduced to one, and a cycle of merging essential tasks is now complete.



Chapter 5. Implementation Issues

5.1 ISE Mobile Agent System

ISE Mobile Agent System (ISE stands for Internet Software Engineering Lab.) is the agent platform adopted in our system, which is based on JAM [23] architecture. JAM is a recent agent architecture implementation derived from PRS [22] (Procedural Reasoning System) and PRS is the first implemented general agent architecture modeled from BDI agent theory. In the original design of JAM architecture, an agent has no built-in method to find out other agents in the system. Nor does JAM have the infrastructure that facilitates communications between agents. All that JAM agents have is the TCP/IP network supports built in Java, and the mobility capability inherited from JAVA serialization mechanism. Hence, the ISE Mobile Agent System extends JAM architecture to construct a multi-agent system and improves the power by adding location tracking and inter-communication mechanisms. Then, it can provide many services for the Agent Manager, such as naming, instantiation, messaging, location tracking and destruction.



5.2 Interfaces of Workflow Manager

Although the whole enactment of a workflow depends on the cooperation of process agents, artifact agents and person agents, the workflow manager actually still plays a requisite and important role in our system architecture. It is responsible for recording, maintaining and associating proper agents in the workflow execution. Here we describe the interfaces provided by the workflow manager. They can be classified into four categories.

- For the client tool

- `String login (String userID, String passwd, String CTid)`
- `boolean logout (String CTid)`

The method `login()` is called by the client tool when an end user logs in our system. It will identify the user's identity and instantiate the corresponding person agent if the password authentication succeeds. The method `logout()` is called when the user logs out our system. The workflow manager interacts with the person agent, obtains the latest relevant data, such as work list and flow list, and stores them back to the database. Then the workflow manager will ask the agent manager to destroy the person agent.

- For the person agent

- `Map getFlowList (String personID)`
- `Map getStatesList (String personID)`
- `Map getWorkList (String personID)`
- `boolean stopWf (String wfinstID)`
- `void startWf (String personID, String wfID)`

These methods are called by the person agent when an end user successfully logs in our system. The method `getFlowList()` returns the flow list that the current user can start. The method `getStatesList()` returns the states of each workflow, which has been started by the user. The method `getWorkList()` returns the work list that needed be accomplished by this user. The method `stopWf()` will firstly check and make sure the stop operation is allowed. If authorized, the workflow manager asks the agent manager to destroy the corresponding process agent and artifact agents properly. Lastly, the method `startWf()`, which is called when the user picks up a workflow to start from the work list, instantiates the corresponding process agent to take charge of the execution of the workflow instance.

- For the process agent

- `String newAA (String PrAid, String wfinstID)`
- `String handleAA (boolean type, String PrAid, String wfinstID)`
- `void reportState (String wfinstID, String state)`

These methods are called by the process agent for workflow enactment. The method `newAA ()` instantiates required artifact agents for task accomplishment of the workflow, so it is usually called at the time when the process agent was instantiated successfully. The method `handleAA()` is responsible for merging or splitting tasks, according to the argument `type`. The method `reportState ()` is used to report the workflow state kept by the process agent, so that the workflow manager can track and record the status of each workflow instance.

- For the artifact agent

- `String queryPeA (String type, String ID)`
- `void addList (String AAid, String wfinstID, String personID)`

These methods are called by the artifact agent for task accomplishment. If the target person agent is available, the method `queryPeA ()` returns the agent-id of an online person agent according to the incoming arguments. The method `addList ()` is used to queue the artifact agent itself into the task list of the workflow manager if the designated person agent is offline.

The interfaces mentioned above are all implemented with the JAVA RMI technology for the distributed environment. Besides those interfaces, there are still many other methods, such as the one for automatically starting a scheduled workflow, and other needed function calls, which are supported by the agent manager.

5.3 Agent Script Format

Except the person agent, the Script Generator in our system needs different script template of the artifact agent and process agent for each workflow to generate corresponding agent script files. Because of the ISE Mobile Agent System is based on JAM architecture, the script language will refer to the one used in JAM [24]. A script file is primarily composed of three major factors called goal, fact and plan. A goal means the behavior that the agent needs to achieve, perform or maintain. A fact contains the information to represent its beliefs. A plan defines a procedural specification for accomplishing a goal. There may be a number of alternative plans for accomplishing the same goal. We will describe some points about each agent script format in the next three sub-sections. And the uppercase words appeared in the subsequent figures represent the preserve words defined in the script language.

5.3.1 Script Format of the Person Agent

```
GOALS:
    MAINTAIN personal_work;
```

Program 1. Goal of Person Agent

Program 1 shows the goal of the person agent. The goal is to maintain the personal work of the corresponding user.

```
FACTS:
    clientToolID    "cbF19ckE";
    roleID          "1079";
    personID        "765/1079";
```

Program 2. Fact of Person Agent

Program 2 shows the fact of the person agent. It includes the clientToolID,

personID and roleID.

```
TEST (&& (== $queryType "startFlow") (== $sender $client_ID));  
ASSIGN $process_Instance_ID (createAgent $message_body);  
EXECUTE sendMessage $clientID "startFlowOK" $process_Instance_ID;
```

Program 3. Procedure of Starting a Workflow

Program 3 describes the procedure of starting a workflow. The meaning of the first line is to check the satisfaction of the action type. If matched, the program goes on. Otherwise, it jumps to next procedure to check the satisfaction of the action type. In the second line, we use a primitive-action interface `createAgent` to create a corresponding Process Agent. It can be transformed to the method `startWf()` mentioned in section 5.2. To simplify the explanation, the process of parsing the string `$message_body` into the two incoming arguments format of the method `startWf()` is omitted. The third line's work is to return the result and the agent-id of the process agent to the client tool. The method `sendMessage()` is the primitive function for communication provided by the ISE Mobile Agent System. Its prototype is shown below and the first incoming argument represents the target agent to communicate with.

```
[Result: String result] sendMessage ([In: String agent_name] [In: String  
queryType] [In: String message_body])
```

```
TEST (&& (== $queryType "queryProgress") (== $sender $client_ID));  
EXECUTE sendMessage $message_body "queryProgress" "";
```

Program 4. Procedure of Querying a Workflow Progress

Program 4 is the procedure of querying a workflow progress. Once the person agent receives the instruction and the action type is matched, it asks the process agent to report the workflow execution progress. Note that now the content of the string

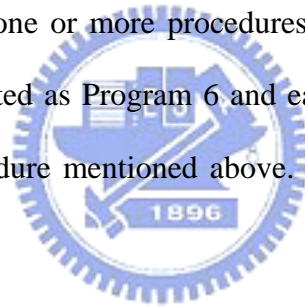
\$message_body is the agent-id of the process agent.

```
TEST (== $queryType "replyQueryProgress");
ASSIGN $message_body (+ $sender " " $message_body);
EXECUTE sendMessage $client_ID "replyQueryProgress" $message_body;
```

Program 5. Procedure of Replying a Workflow Progress

Program 5 is the procedure of replying a workflow progress. It happens when the person agent receiving the results reported by the process agent. In the second line, we prefix the string \$sender (here is the agent-id of the process agent) to the string \$message_body (here is the received workflow progress). It facilitates the client tool to distinguish return results.

A plan is composed of one or more procedures shown above. The plan of the person agent can be represented as Program 6 and each {... ...} pair will be exactly mapped to a particular procedure mentioned above. And the number of plans in an agent can be more than one.



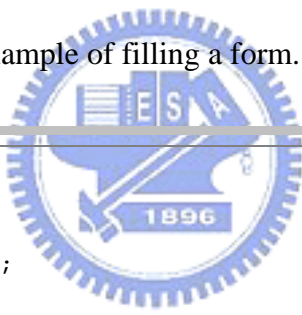
```
Plan: {
GOAL:
    MAINTAIN personal_work;
BODY:
    RETRIEVE clientToolID $client_ID;
    ASSIGN $waiting "TRUE";
    WHILE: TEST (== $waiting "TRUE"){
        WAIT: TEST (FACT MESSAGE $sender $queryType $message_body);
        RETRACT MESSAGE $sender $queryType $message_body;
        OR {... ..
        }{... ..
        }{... ..
        };
    ASSIGN $queryType "";
    ASSIGN $sender "";
```

```
    ASSIGN $message_body " ";
};
```

Program 6. Plan of Person Agent

5.3.2 Script Format of the Artifact Agent

The goal of the artifact agent is to maintain the workflow data objects and related handlings of the corresponding artifact. The fact of the artifact agent contains the agent-id of its leading process agent and the states, such as "filled", "audited", "logged" and "over". These states are initialized to FALSE. The fact also includes the data objects of the artifact carried by this agent, such as the field name, type, value and related filling constraints. Hence, the fact and plan of each artifact vary. Program 7 below gives a simple plan example of filling a form.



```
Plan: {
GOAL:
    MAINTAIN artifact_work;
PRECONDITION:
    FACT filled "FALSE";
    FACT audited "FALSE";
    FACT logged "FALSE";
BODY:
    RETRIEVE processInstanceID $prAID;
    ASSIGN $waiting "TRUE";
    WHILE: TEST (== $waiting "TRUE"){
        WAIT: TEST (FACT MESSAGE $sender $queryType $message_body);
        RETRACT MESSAGE $sender $queryType $message_body;
        OR {
01 TEST (&& (== $queryType "routeDecision") (== $sender $prAID));
02 ASSIGN $pID (parseMessage $message_body "/");
03 OR {
04 TEST (== $pID "0");
05 EXECUTE connectToAgent "RoleID" $message_body $pID;
```

```

06  ASSIGN $resultType (interactWith $pID);
07  }{
08  EXECUTE connectToAgent "PersonID" $pID $p;
09  ASSIGN $resultType (interactWith $pID);
10  };
11  WHEN: TEST (== $resultType "taskCompleted"){
12  UPDATE (filled) (filled "TRUE");
13  };
14  ASSIGN $waiting "FALSE";
15  EXECUTE sendQuery $pRAID $resultType "";
    };
    ASSIGN $queryType "";
    ASSIGN $sender "";
    ASSIGN $message_body "";
  };
}

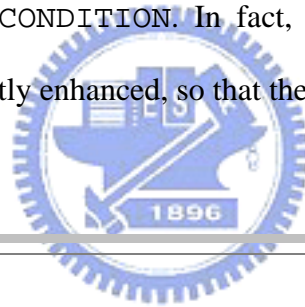
```

Program 7. Plan of Artifact Agent

Note that in this plan we use PRECONDITION to assure the triggered conditions. Line 01 means that the artifact agent only accepts the route decision sent by the leading process agent, whose agent-id is in its fact. The work of Line 02 is to parse the string \$message_body to get a particular roleID or designated personID, so lines 03 to 10 can take corresponding handlings respectively. The primitive-action interface connectToAgent will be transformed to the method queryPeA() mentioned in section 5.2. And the method interactWith handles the agent migration, representation and manipulation of data objects. Lines 11 to 15 mean that if the string \$resultType is taskCompleted, then the filled state in the fact will be changed to TRUE (i.e. the form has been filled) and the string \$waiting is changed to FALSE. This plan finishes here since the while loop condition cannot be satisfied any more.

5.3.3 Script Format of the Process Agent

The goal of the process agent is to guide the enactment and related handlings of the corresponding workflow routine. The fact of the process agent contains the states, such as "initialized", "filled", "audited" and "logged", and information of the starter and corresponding artifact agents (i.e. agent-id). These states are initialized to FALSE. The fact also includes a derived list for recording the workflow execution progress. Again, the fact and plan of each workflow vary according to different workflow definitions. Program 8 below gives a simple plan example of handling an essential task. For simplicity, here we use PRECONDITION to determine the plan sequence. And the method `findAgent` is used to determine the target person agent according to the incoming argument `$CONDITION`. In fact, the plan sequence and routing decision approach can be greatly enhanced, so that the process agent will behave more dynamically and intelligently.



```
Plan: {
GOAL:
  ACHIEVE test_flow;
PRECONDITION:
  FACT  INITIALIZED "TRUE";
  FACT  FILLED      "FALSE";
BODY:
  RETRIEVE artifactInstanceID_1 $artInsID_1;
  RETRIEVE findCond_1 $CONDITION;
  ASSIGN $personID (findAgent $CONDITION);
  EXECUTE sendMessage $artInsID_1 "routeDecision" $personID;
  ASSIGN $waiting "TRUE";
  WHILE: TEST (== $waiting "TRUE"){
    WAIT: TEST (FACT MESSAGE $sender $queryType $message_body);
    RETRACT MESSAGE $sender $queryType $message_body;
  OR {
```

Procedure 1

```
TEST (== $queryType "queryProgress");
RETRIEVE derivNow $now;
RETRIEVE derivLast $last;
ASSIGN $message_body (+ $now "/" $last);
EXECUTE sendMessage $sender "replyQueryProgress" $message_body;
}{
```

Procedure 2

```
TEST (== $queryType "taskCompleted");
OR {
  TEST (== $sender $artInsID_1);
  ASSERT artIns_1 "TRUE";
};
WHEN: TEST (&& (FACT artIns_1 "TRUE")){
  ASSIGN $waiting "FALSE";
  UPDATE (FILLED) (FILLED "TRUE");
  RETRIEVE derivLast $Last;
  RETRIEVE derivNow $Now;
  UPDATE (derivNow) (derivNow (first $Last));
  UPDATE (derivLast) (derivLast (rest $Last));
};
}{
```

Procedure 3

```
TEST (== $queryType "taskFailed");
OR {
  TEST (== $sender $artInsID_1);
  ASSIGN $personID (findAgent $CONDITION);
  EXECUTE sendMessage $artInsID_1 "routeDecision" $personID;
};
}{
```

Procedure 4

```
TEST (== $queryType "flowCancelled");
EXECUTE deleteAgent $artInsID_1;
UNPOST MAINTAIN test_flow;
ASSIGN $waiting "FALSE";
};
ASSIGN $queryType "";
```



```
    ASSIGN $sender "";  
    ASSIGN $message_body "";  
};  
};  
}
```

Program 8. Plan of Process Agent

The code segment in procedure 1 shows that the process agent reports current workflow progress to the person agent. It responds to the one mentioned in sub-section 5.3.1. The code segment in procedure 2 is activated when the result reported by the artifact agent is taskCompleted. The operations include updating the filled state and changing the string \$waiting to FALSE. The plan finishes afterwards and another plan may be activated to determine the next routing decision. Besides, according to the reported result, splitting or merging tasks may be triggered. The code segment in procedure 3 is activated when the reported result is taskFailed. It will try to determine the next route and send the new target personID to the artifact agent again. Lastly, the code segment in procedure 4 is activated when one user cancels this flow. The way we used here is to stop the goal. Actually, there are many things needed to be further considered. Examples include as the stop authorization checking, logging, notification policy and the rollback manipulation etc.

Chapter 6. Conclusion & Future Work

Many related research works have focused on the agent-assisted conceptual models of B2B or B2C workflows, however, the primitive agent model, operational behavior and collaborations are seldom detailed. In this thesis, we aim at designing an *agent-based* system architecture for primitive workflows. We model the business process scenarios as the interactions among the system components, users, and software agents, and exploit the design from the agent's viewpoints to cope with the inappropriateness appeared in traditional WfMSs.

Besides, the system components and software agents with their contents and capabilities are all clearly illustrated. The interactions among them to fulfill the typical workflow behaviors are also shown with sequence diagrams. Furthermore, some implementation issues, such as the interfaces between components and agent script templates, about the realization of our system, are addressed explicitly.

In the future, we plan to investigate the design of the workflow definition tools from the agent's viewpoints, since our goal is to utilize the properties and capabilities of software agents to enhance the development process of workflow-based applications. In addition, we are looking at the structure of the client tool for introducing the concept of active agents for better interface representations and manipulations.

Reference

- [1] [URL] The Workflow Management Coalition, <http://www.wfmc.org>
- [2] The Workflow Management Coalition, “Workflow Management Coalition The Workflow Reference Model”, Document Number TC00-1003, January 1995.
- [3] The Workflow Management Coalition, “Workflow Management Coalition Workflow Standard Workflow Process Definition Interface – XML Process Definition Language”, Document Number WFMC-TC-1025, October 2002.
- [4] Nicholas R. Jennings and Michael J. Wooldridge, Agent Technology: Foundations, Applications, and Markets, Springer-Verlag, February 1998.
- [5] Nicholas R. Jennings and Michael J. Wooldridge, “Intelligent agents: Theory and practice”, Knowledge Engineering Review, Volume 10, Number. 2, pp.115-152, Cambridge University Press, June 1995.
- [6] Nicholas R. Jennings and Michael J. Wooldridge, “Agent Theories, Architectures, and Languages: A Survey”, Workshop on Agent Theories, Architectures and Languages (ECAI'94), Volume 890 of Lecture Notes in Artificial Intelligence, pp. 1-22, Amsterdam, Netherlands, Springer-Verlag, January 1995.
- [7] Nicholas R. Jennings, Timothy J. Norman and Peyman Faratin, “ADEPT: an agent-based approach to business process management”, ACM SIGMOD Record, Volume 27, Issue 4, pp.32-39, ACM Press, New York, December 1998.
- [8] Yuhong Yan, Maamar Z. and Weiming Shen, “Integration of workflow and agent technology for business process management”, The Sixth International Conference on Computer Supported Cooperative Work in Design, pp.420-426, London, Ontario, Canada, 12 - 14 July 2001.
- [9] Weishuai Yang, Shanping Li and Ming Guo, “Mobile agent: enhancing workflow interoperability”, International Conferences on Info-tech and Info-net, Beijing,

China, Volume 5, pp.276-282, 29 October – 1 November 2001.

- [10] Liangzhao Zeng, Ngu Anne, Benatallah Boualem and O'Dell, Milton, “An agent-based approach for supporting cross-enterprise workflows”, 12th Australasian Database Conference, pp.123-130, Gold Coast, Queensland, Australia, 29 January – 2 February 2001.
- [11] [URL] Flowring Corp., <http://www.flowring.com>
- [12] Chen, M. -F., Liang, B. -S., Lin, J. -R., and Wang, F. -J., “Enacting a Software Development Process,” IEEE ICSCCS’97, pp. 3-12.
- [13] Chen, M. -F., Liang, B. -S., and Wang, F. -J., “A Process Centered Software Engineering Environment with Network Centric Computing,” IEEE FTDCS’97, pp. 234-239.
- [14] Gregory Alan Bolcer and Richard N. Taylor, “Advanced Workflow Management Technologies”, Software Process: Improvement and Practice, Volume 4, Number 3, pp.125-171, September 1998.
- [15] Layna Fischer, Workflow Handbook 2003, Future Strategies, Lighthouse Point, Florida, April 2003.
- [16] Danny B. Lange and Mitsuru Oshima, “Seven good reasons for mobile agents”, Communications of the ACM, Volume 42, Issue 3, pp.88-89, ACM Press, New York, March 1999.
- [17] Michael E. Bratman, Intention, Plans and Practical Reason. Harvard University Press, Cambridge, Massachusetts, 1987.
- [18] [URL] FIPA ACL Specifications, <http://www.fipa.org/repository/aclspecs.html>
- [19] [URL] Simple Object Access Protocol, <http://www.w3.org/TR/soap/>
- [20] Tim Finin, Yanis Labrou and James Mayfield, “KQML as an agent communication language”, Software agents, pp. 291-316, MIT Press, Cambridge, Massachusetts, 1997.

- [21] Jin W. Chang and Colin T. Scott, “Agent-based workflow: TRP Support Environment (TSE)”, Computer Networks and ISDN Systems, Volume 28, Issues 7-11, pp. 1501, May 1996
- [22] M. P. Georgeff and A. L. Lansky, “Reactive reasoning and planning”. In Proceedings of the Sixth National Conference on Artificial Intelligence, pp. 677-682, Seattle, 1987.
- [23] M. J. Huber, “JAM: A BDI-theoretic mobile agent architecture”. In Proceedings of the Third International Conference on Autonomous Agents, pp. 236-243, May 1999.
- [24] <http://www.marcush.net/IRS/Jam/Jam-man-01Nov01.doc>
M. J. Huber, “JAM Agents in a Nutshell”, 2001

