3. The Reusability of Workflow Management Systems

In this section, we try to clarify the “reausability” of the components of workflow management system.

According to the introduction in section 2, a workflow management system comprises many components and these components include a set of objects. Therefore, the reuse of WfMS components should be the reuse of a “*framework*” [12]. A *framework* is the design of a set of objects that collaborate to carry out a set of responsibilities [12]. Frameworks usually provide the design of interfaces or abstract classes to represent what the developers should implement and the way that its functions are divided amoung objects. Therefore, to clarify the reusability of components is necessary for design a highly reusable workflow management system. Clarifying the reusability will help us to draw up the blueprint of the framework of a reusable workflow management system.

Refer to WfMC’s reference model [9] and OMG’s workflow management facility [47], we generalize the components of WfMS and illustrate them in Figure 11. In addition to the components introduced in reference model [9], we add two functions which are common in moden workflow systems – Failure Recovery [13] and Persistence. Failure Recovery is an important issue in workflow management, it is essential for some workflow system that emphasize stability and credibility (eg transactional workflow system). Usually, WfMS needs to records some information for evalution or recovery purpose and the persistence components are used to store information to the data storage.

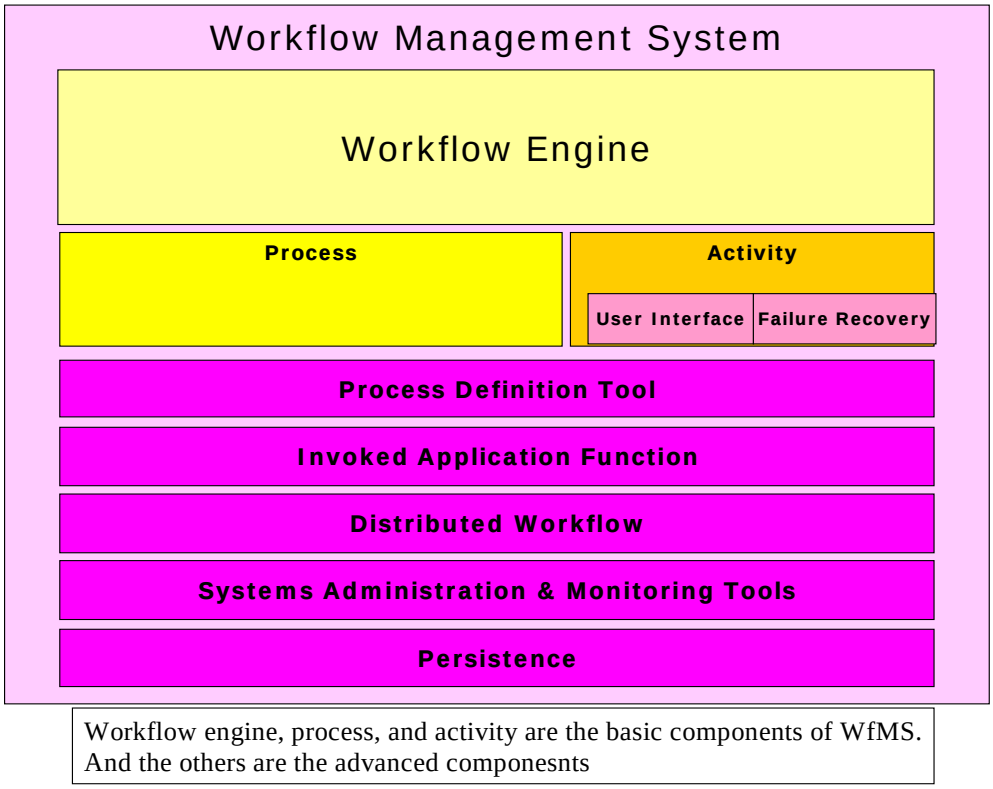

Figure 11: The Architecture of Workflow Management System

As shown in Figure 11, the basic compoents of WfMS include *Workflow Engine*, *Process*, and *Activity*; a simplest WfMS may only comprise these three compoents. The others are the advanced components; the advanced components include *Failure Recovery*, *Process Definition Tool*, *User Interface*, *Application Invocation*, *Workflow Interoperability* and *System Administration & Monitoring Tools*. These components may be not essential for the execution of WfMS, but with the popularization of WfMS in different domain, the advanced components become more and more important. Below, we try to define the reusability of several important components of WfMS.

3.1 Workflow Engine

The workflow engine in different WfMS products may not implement all functions listed in WfMC's Reference Model [9], but as the core of WfMS, workflow engine is

usually responsible for various and complex functions. Therefore, the reusability should be promoted by a *compositional design* [8]. That is, some functions of workflow engine can be implemented in other components or classes and only the basic functions (e.g. the methods that control the execution of workflow engine) are implemented within workflow engine. The workflow engine should be composed of the components or classes that implement the needed functions. Developers may change these components to meet the change of requirements instead of modifying the programs of workflow engine.

Therefore, the reusability of workflow engine should be that the workflow engine component can be reused immediately and the developers may only need to change the configuration instead of modifying the workflow engine class.

3.2 Workflow Process

Process comprises activities, data and objects used to navigate and execute this process. These elements comprise process are different with process definitions. Therefore, the design of process component should allow developers to combine activities with process and change these activities easily. Therefore, the compositional design may be suitable. However, in practice, the combination of process and activities is performed by system, not by developers. Therefore, in addition to the compositional design, a mechanism for workflow system to combine process with activities automatically is necessary.

In addition to activities, the process may need other functions, e.g. to keep the log of executoin or to manipulate database. Although the number of required functions may be not as many as that of workflow engine, we believe the compositional design is

also suitable for process component. Therefore, the reusability of workflow process is that the process component can be reused immediately and the developers may only need to change the configuration instead of modifing the process class. Moreover, the process component can load specific activities according to process definition during the execution.

3.3 Workflow Activity

Activity is the basic unit of workflow process, an acitivity in workflow process represents a task in business process in the real word. Therefore, the “task code” (sometimes, we also call it execution code) executed by an activity is different from that executed by another activity. In fact, the way to control activity instances changes rarely in a WfMS, therefore, we think the activity component should be reused to execute different execution code according to process definition.

In addition to executing task code, the activity plays a role of decision-making in flow control aspect. Flow control is important for the execution of process. There will be some braches in a complex process. When meeting branches, process should make a decision according to some information (e.g. the state of some activities in this process). Most of modern WfMSs allow developers to define the decision rules with programs. Often, the decision rules are executed by some special activities, we call these activities *transitions*. Some supports or mechanisms should be provided in WfMS for developers to write programs for transitions and support the execution of tranisition. In our opinion, the reusability of transition component is that a transition component can be reused when meeting similar branches in differen process and the decision rules can still be used without modification even if the process has been

changed (e.g. add or delete activities in process).

In this section, we also discuss two advanced components that are highly related to the activity component - User Interface and Failure Recovery.

3.3.1 User Interface

The user interface we introduce here corresponds to workflow client application in WfMC's Reference Model and is shown as a separate software component, responsible for the look and feel of the user dialogue and control of the local interface with the user [9]. It is common that some activities involve users in execution. For instance, most of the forms in an electronic official document system need user to fill in. User interface components show messages or forms to users and deliver the user response to system. The user interfaces of activities are various and users may access them via different device. Therefore, the reusability of user interface should be that the user interface can be generated easily and the developers need not to consider how to display the user interface in different device when designing user interface.

To be generated easily means that a GUI (graphic user interface) editor for user interface may be provided. Like some commercial products, e.g. Borland Jbuilder for Java windows programming and Macromedia Dreamweaver for homepage generation, we hope to provide a GUI editor for developers to edit user interface in the future. However, to generate user interface via GUI editor means that some window components like textfield, teaxarea, button and so should be offered to comprise user interface. The reusability of these window components should be the same as that of user interface component.

3.3.2 Failure Recovery

Some workflow sytems (e.g. transactional WfMS) which emphasize the stability and reliability may implement the failure recovery mechanism. Two different types of problems or anomalous situations can occur during workflow execution: exceptions and failures [29]. Exceptions are semantic failures that can be caused by a system failure or by a new situation introduced by the external environment. The other problem in managing workflow processes is that of failure recovery [30][31][32]. The goal of failure recovery is to bring the failed workflow process back to some semantically acceptable state so that the cause of the failure can be identified. The basic failure-recovery process includes the following three steps:

1. The execution of the workflow process is terminated and the workflow engine then decides the end compensation point (ECP) and the compensation set of the occurred failure.
2. Activities in the compensation set are compensated.
3. The workflow process is restarted from the ECP.

An ECP is a previously executed activity of the workflow process which represents an acceptable intermediate execution state where certain actions can be taken so that either the problems which caused the failure can be fixed or the execution path of the workflow can be altered to avoid the problem. Compensating an activity *A* involves executing a *compensation subroutine* that attempts to undo the effects of the previous execution of *A*. This compensation can be very expensive, so it is important to minimize compensation scope to avoid unnecessary compensation effort. In this paper, those activities that require compensation are called the *compensation set* [30] of a failure. Figure 12 illustrates the basic failure-recovery model.

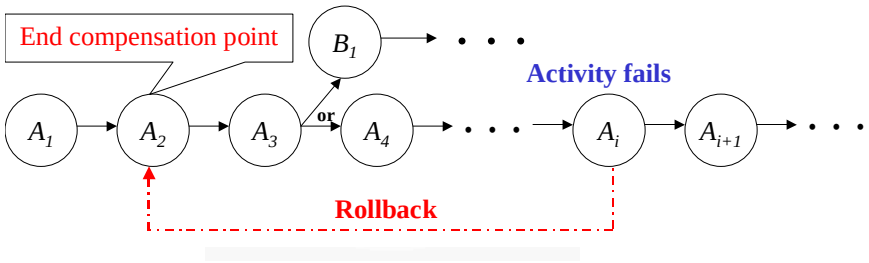

Figure 12: The basic failure-recovery model.

Generally speaking, the failure recovery definitions are included in activities that may cause the occurrences of failure. Some information needed for failure recovery, e.g. type of failure, ecp and compensation set, are included in failure-recovery definition. In the traditional failure recovery model, to change process (e.g. to add or delete activities in the proces) may affect the failure-recovery definitions in some activities. But we hope the failure recovery component can be reused even if the change of process. Therefore, the reusability of failure recovery component should be that the failure recovery component can still be used without modification even if the process has been changed.

3.4 Process Definition Tool

The process definition tool is used to create the process description in a computer processable form. As mentioned before, the process definition tool is used in build-time and is not essential for the operation of WfMS. Therefore, some workflow systems provide their own process definition tools, but some others don’t regard definition tools as a part of workflow system, flow designers may use tools provided by other venders. However, we think that the system-independent tools may have better reusability. An ideal condition is that a process definition tool can be used by different workflow systems, i.e. the reusability of process definition tool is that different workflow systems may be able to use the same process definition tool.

3.5 Distributed Workflow

The distributed workflow components are used for workflow interoperability (Interface 4 of WfMC's Reference Model [9]). According to [9] [14][15], Distributed Workflow Components allow different workflow systems (homogeneous or heterogeneous) to cooperate with each other (e.g. pass work items or resources between one another). Workflow systems produced by different vendors may be implemented with different technologies (e.g. CORBA or Java) or be implemented for different platforms; therefore, we think that the distributed workflow components should support the cooperation between workflow systems regardless of the differences of implementation technologies or platforms. The platform-independent components also have better reusability.

3.6 Persistence

Workflow Management Systems log the execution "history" of the workflows they execute for evaluation and recovery purpose [19]. The persistence components are used to store the history to database and retrieve it when needed. Because of using database, it is difficult for process component to have high level reusability. Different WfMS may log different history and the operation to manipulate may be different when using different database systems (e.g. MS SQL server, Oracle or Sybase). However, we may design the persistence component from the aspect of users. To a process designer, he/she may pay attention to the data needed to be kept rather than how to manipulate database systems. By contrast, the programmer who implements the codes to manipulate database systems may want to concentrate on the operations for different database systems rather than the workflow data needed to store. That is, the reusability of persistence component is that the persistence component can

provide *database-independence* operations for workflow users.