2. Introduction to Workflow System

In this chapter, we discuss the basic of Workflow and Workflow Management System to provide basic understanding of this technology. In this section, we mainly refer to Workflow Reference Model [9] published by Workflow Management Coalition (WfMC).

2.1 Introduction to Workflow

Workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal [6][9]. In [9], it defines the workflow as:

The computerized facilitation or automation of a business process, in whole or part.

2.1.1 Process and Workflow

Process is a sequence of activities; business processes are processes that achieve the goal (or the mission) of business. Most people associate business processes with the typical operation in a bank or an insurance company: the loan process in a bank or claims processing in an insurance company. To put it more precisely, business process is a sequence of activities performed by various persons and have a plain results. Moreover, batch jobs can also be considered some kind of business process.

The *process model* describes the structure of a business process in the real world. It defines all possible paths through the business process, including the rules that define which paths should be taken and all actions that need to be performed. This model is a template from which each process is instantiated; that is, an instance of the process model is created. We can use the term *process instance* for the instances that are

created from process models.

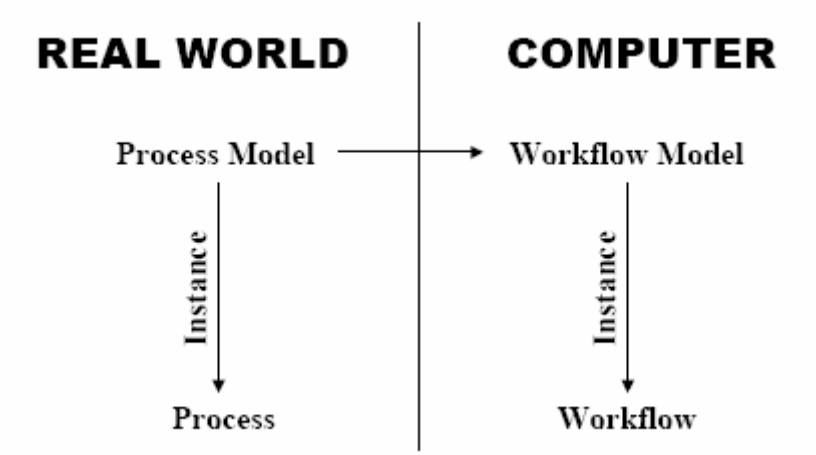

Figure 1: Process and Workflows

Business process may consist of parts that are carried out by a computer and parts that are not supported through computers.As shown in Figure 1; the parts that are run on a computer are called a *workflow model* [22]. The workflow model (also referred to as *workflow process definition*) is the computerized representation of the business process. It defines the starting and stopping conditions of the process, the activities, and control and data flows among these activities. Refer to [13], we define activity as following:

> *An activity is a logical step or description of a piece of work within a workflow, that can involve manual interaction with a user or workflow participant which includes the information about the starting and stopping conditions, the users who can participate, the tools and/or data needed to complete this activity, and the constraints on how the activity should be completed.*

Activities are usually organized into a directed graph that defines the order of execution among the activities in the process. Nodes and edges in the graph represent activities and control flow, respectively.

A workflow model may equal whole process model or just be a small part of a larger process model. The workflow model is a template for creating workflows just like process model is a template for creating processes; a *workflow instance* is the instances that are created form workflow models. In the most cases, a workflow may be "executed" by *workflow management system* (WfMS). We introduce workflow management system at section 2.2.

2.2 Introduction to Workflow Management System

A Workflow Management Systems (WfMSs) are software systems for supporting coordination and cooperation among members of an organization whilst they perform complex business tasks via workflows [6][9]. It is one, which provides procedural automation of business process by management of the sequence of work activities and invocation of appropriage human and/or IT resources associated with the various activity steps. In [9], it defines workflow management system as:

> *A system that completely defines, manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic.*

2.2.1 Components of Workflow Management System

According to [9][22], there are three kinds of components of WfMS: *Meta-model component*, *Build-time components* and *Run-time components*. Figure 2 illustrates the major components of workflow management system and the relationship between these components. In fact, this work mainly focuses on the design issues of run-time components. We will introduce these components at the following sections.

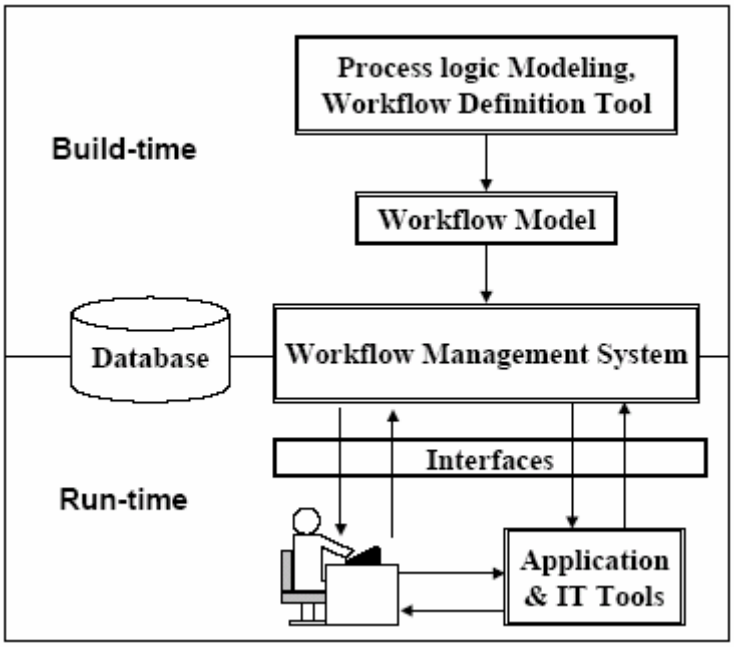

Figure 2: Major components of a workflow management system

2.2.2 Meta-Model Component

The *Meta-Model* of workflow definition describes the entities contained within a process definition, their relationships and attributes. Meta-model component is made up of constructs that available to user to model their environment, such as workflow models, organizational structures etc. It can be record in a table at relational databases or be a construction language scheme like workflow model languages. There are several kinds of workflow model languages already have been developed. We will have a brief introduction to those models.

2.2.2.1 Direct Graphics Model

Direct gragh model [6] is the most popular way to construct workflows. It uses nodes to represent activities, edges to represent control flow. The advantage of this model is intuition because it can be represented with graphics. Flow designer can use nodes and edges by intuition without training. Disadvantage of direct graphics model was it only could carry limited information out.

2.2.2.2 Wf-XML Model

Wf-XML model [10] uses XML language as a definition language for workflow models. It defined servel tag to help flow designer to construct workflows. The advantages of Wf-XML are easy to expend and exchange especially in distributed systems. However, because of the complex structure of XML document, it is hard for human to understand. Recently, Workflow Management Coalition (WfMC) also introduced a XML Process Definition Languages (XPDL) to support the representation of Workflow Model [11]. Because of the XML format and the Meta-model established by WfMC, the advantage of XPDL is its extensibility and interchangeability. Latter, we will introduce the details of XPDL in Section 8.2.1.

2.2.2.3 Petri Net Model

Petri Net models [35] are graphics and mathematical modeling tool applicable to many systems. They are promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic, and/or stochastic. As a graphical tool, Petri nets can be used as a visual-communication aid similar to flow charts, block diagrams, and networks. It is a good workflow-modeling tool because of first; it is formal semantics despite the graphical naturally. Second, it use state-based instead of event-based. Third, it is abundance of analysis techniques.

2.2.3 Build-Time Component

The *build-time component* of the workflow management system provides the functions and capabilities to define, test, and manage all workflow-related information [9][22]. It provides the functions to define user-specific constructs in terms of the

meta-model. Build-time functions are those, which result in a computerized definition of a business process. During this phase, a business process is translated from the real world into a formal, computer processable definition by the use of one or more analysis, modeling and system definition techniques. The resulting definition is sometimes called a workflow model, workflow metadata, or a workflow definition.

In general, the build-time component also provides functions to define and manage administrative and system-management-related information. Typical examples of this type of information are session-related information, such as the maximum amount of time a user can work with the workflow management system, or operational characteristics, such as the actions the workflow management system should take if the response time exceeds a defined limit.

The build-time of a workflow management system usually manifests itself in two different ways. One way is the definition of the workflow information by means of a *graphical end user interface* (GUI); the other is by a proprietary or standardized *workflow definition language.*

2.2.4 Run-Time Component

The *run-time component* navigates through workflow processes, activities and interacts with users and applications [9][22].

At run-time the process definition is interpreted by software which is responsible for creating and controlling operational instances of the process, scheduling the various activities steps within the process and invoking the appropriate human and IT application resources, etc. These run-time process control functions act as the linkage between the process as modeled within the process definition and the process as it is

seen in the real world, reflected in the runtime interactions of users and IT application tools. The core component is the basic workflow management control software, responsible for process creation & deletion, control of the activity scheduling within an operational process and interaction with application tools or human resources. This software is often distributed across a number of computer platforms to cope with processes, which operate over a wide geographic basis.

2.3 The Workflow Reference Model

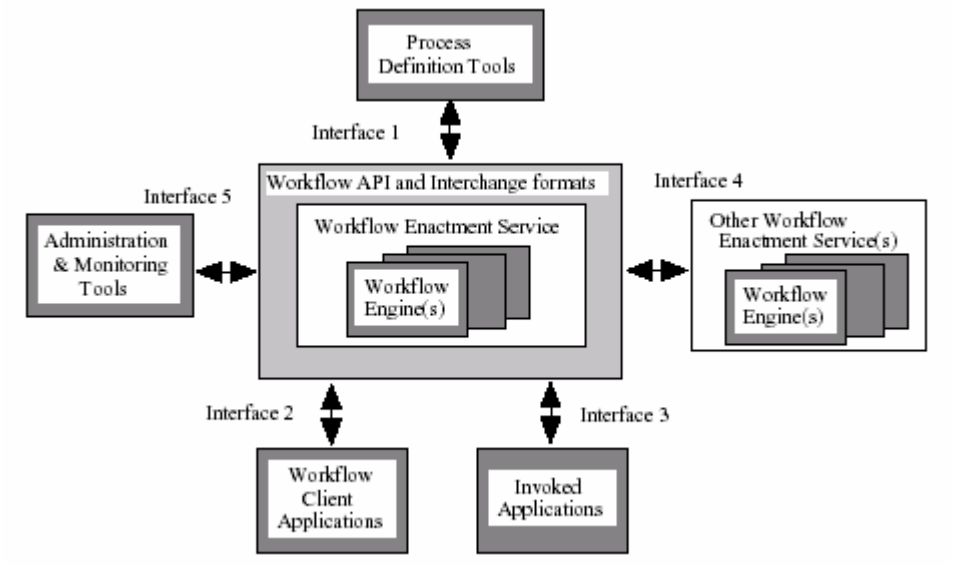

Figure 3: WfMC's workflow reference model

WfMC's reference model for workflow technology is developed from the generic workflow application structure by identifying the interfaces within this structure which enable products to interoperate at a variety of levels. All workflow systems contain a number of generic components which interact in a defined set of ways; different products will typically exhibit different levels of capability within each of these generic components [9]. The reference model with its major components and interfaces is depicted in Figure 3, and described below.

2.3.1 Workflow Enactment Services

The workflow enactment service provides the run-time environment in which process instantiation and activation occurs, utilising one or more workflow management engines, responsible for interpreting and activating part, or all, of the process definition and interacting with the external resources necessary to process the various activities [9]. Workflow enactment service comprises workflow engine, process, activity, and some data for execution purpose. At the following subsections, we introduce these components – workflow engine, workflow process and workflow activity.

2.3.1.1 Workflow Engine

Workflow Engine is a software service or "engine" that provides the run time execution environment for a workflow instance [9]. According to [9], workflow engine should provide facilities to handle:

- Interpretation of the process definition
- Control of process instances - creation, activation, suspension, termination, etc.
- Navigation between process activities, which may involve sequential or parallel operations, deadline scheduling, interpretation of workflow relevant data, etc
- Sign-on and sign-off of specific participants
- Identification of workitems for user attention and an interface to support user interactions
- Maintenance of workflow control data and workflow relevant data, passing workflow relevant data to/fromapplications or users
- An interface to invoke external applications and link any workflow relevant data
- Supervisory actions for control, administration and audit purposes

A workflow engine controls the executionof a set of process, or sub-process, instances with a defined scope – determined by the range of object types, and their attributes, which it can interpret within the process definition(s) [9].

2.3.1.2 Workflow Process

Business tasks are modeled as *workflow processes*, which are then automated by the WfMS. *workflow process definition* defines the starting and stopping conditions of the process, the activities in the process, and control and data flows among these activities. A *workflow process instance* is the execution of a workflow process definition by the WfMS [13]. The workflow enactment service may be considered as a state transition machine, where individual process instances change states in response to external events (eg suspension because of failure or exception). An illustrative basic state transition scheme for process instances is shown in Figure 4 [9].

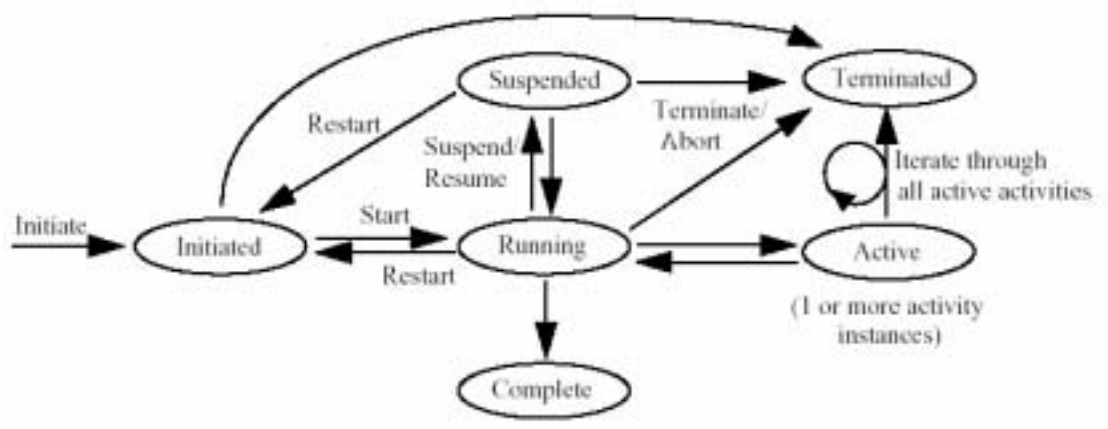

Figure 4: basic state transitions of a process instance

Refer to [9], the basic states are:

- *Initiated* - a process instance has been created, including any associated process state date and workflow relevant data, but the process has not (yet) fulfilled the

conditions to cause it to start execution.

- *Running* - the process instance has started execution and any of its activities may be started (once any appropriate activity start conditions have been met)
- *Active* - one or more of its activities has been started (ie a workitem has been created and assigned to an appropriate activity instance)
- *Suspended* - the process instance is quiescent and no activities are started until the process has returned to the running state (via a resume command)
- *Completed* - the process instance has fulfilled the conditions for completion; any internal post-completion operations such as logging audit data or statistics will be performed and the process instance destroyed
- *Terminated* - execution of the process instance has been stopped before its normal completion; any internal operations such as error logging or logging recovery data may be performed and the process instance destroyed.

2.3.1.3 Workflow Activity

Individual activities within a workflow process are typically concerned with human operations, often realized in conjunction with the use of a particular IT tool (for example, form filling), or with information processing operations requiring a particular application program to operate on some defined information (for example, updating an orders database with a new record). Interaction with the process control software is necessary to transfer control between activities, to ascertain the operational status of processes, to invoke application tools and pass the appropriate data, etc. There are several benefits in having a standardized framework for supporting this type of interaction, including the use of a consistent interface to multiple workflow systems and the ability to develop common application tools to work with different workflow products.

Similar to workflow process, activity instances also change states in response to external events (eg completion of the activity). The state transition for activity instances is illustrated in Figure 5 [9].

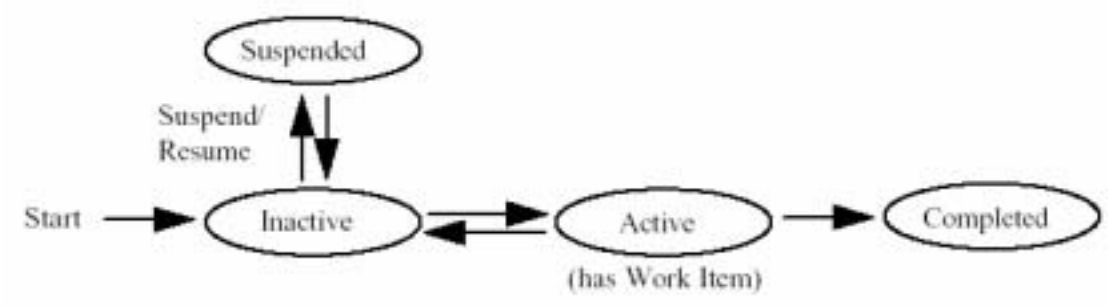

Figure 5: basic state transitions for activity instances

According to [9], the basic states of an activity instance are:

- *Inactive* - the activity within the process instance has been created but has not yet been activated (eg because activity starting conditions have not been met) and has no workitem for processing
- *Active* - a workitem has been created and assigned to the activity instance for processing
- *Suspended* - the activity instance is quiescent (eg as a result of a change_state_of_activity_instance command) and will not be allocated a workitem until returned to the running (inactive) state
- *Completed* - execution of the activity instance has completed (and any stopping conditions will be evaluated)

Most activities may be non-interruptible; i.e. once a workflow service has started a particular activity within a process instance, it may not be possible to suspend or terminate that activity. This means that suspension / terminate functions cannot be

completed until all active activities have completed and the process instance returned to a running state. In addition, it may be required to mark a set of activities as an atomic unit, which is either executed in entirety or the process instance ‘rolled-back’ to a restart point. The potential treatment of interruptable activities and atomic activity units with restart capability will require further consideration, and we will not discuss this issue in this paper.

2.3.2 Process Definition Tools

A variety of different tools may be used to analyse, model, describe and document a business process; such tools may vary from the informal ("pencil and paper") to sophisticated and highly formalised. Where a workflow product provides its own process definition tool, the resultant process definitions will normally be held within the workflow product domain and may, or may not, be accessible via a programming interface for reading and writing information. However, the workflow model is not concerned with the particular nature of such tools nor how they interact during the build-time process. Therefor, process definition tools may not be considered a part of workflow system, and the process definitions may be transferred between the products as and when required or may be stored in a separate repository, accessible to both products (and possibly other development tools).

The final output from this process modelling and design activity is a process definition which can be interpreted at runtime by the workflow engine(s) within the enactment service.

For today's workflow products each individual process definition is typically in a form specialised to the particular workflow management software for which it was

designed. Therefor, it is difficult to support the exchange of process definition information over a variety of workflow products. A standardised form is necessary to enable more flexiblility. The interface 1 of WfMC's Reference Model is established to support workflow definition interchange. The nature of the interface is an interchange format and API calls, which can support the exchange of process definition information over a variety of physical or electronic interchange media. According to [9], there are two aspects to WfMC's work in this area:

1. derivation of a meta-model which can be used to express the objects, their relationships and attributes within a process definition and which can form the basis for a set of interchange formats to exchange this information between products. Therefor, XPDL [11] is not only a XML definition language but a representation of a mata-model defined by WfMC.
2. API calls (within the WAPI) between workflow systems or between a workflow system and process definition product, providing a common way to access workflow process definitions. Access may be read, read/write or write only and may manipulate the set of standard objects defined within the meta-model or a product-specific set (for example defined in a product type register).

2.3.3 Workflow Client Applications

The workflow client applications is the software entity which interacts with the end-user in those activities which require involve human resources. In the workflow model interaction occurs between the client application and the workflow engine through a well defined interface embracing the concept of a *worklist* - the queue of work items assigned to a particular user (or, possibly, group of common users) by the components comprise workflow enactment service (may be workflow engine or activity, depends on the design of workflow system). Where user interactions are

necessary within the process execution, the workflow engine or activity place items on to worklists for attention by the *worklist handler*, which manages the interactions with the workflow participants.

A worklist may contain items relating to several different active instances of a single process and/or individual items from activations of several different processes. A worklist handler might potentially be interacting with several different Workflow engines and several different enactment services. (According to individual product implementation, separate physical worklists may be maintained for each process type, or the worklist handler may consolidate the various worklists items into a single representation to the end-user). According to [9], the interface between the client workflow application and Workflow enactment service must therefore be sufficiently flexible in terms of its use of:

- process and activity identifiers
- resource names and addresses
- data references and data structures
- alternative communications mechanisms

to contain these variations of implementation approach.

Refer to [9], a generic structure for workflow client application is shown in Figure 6.

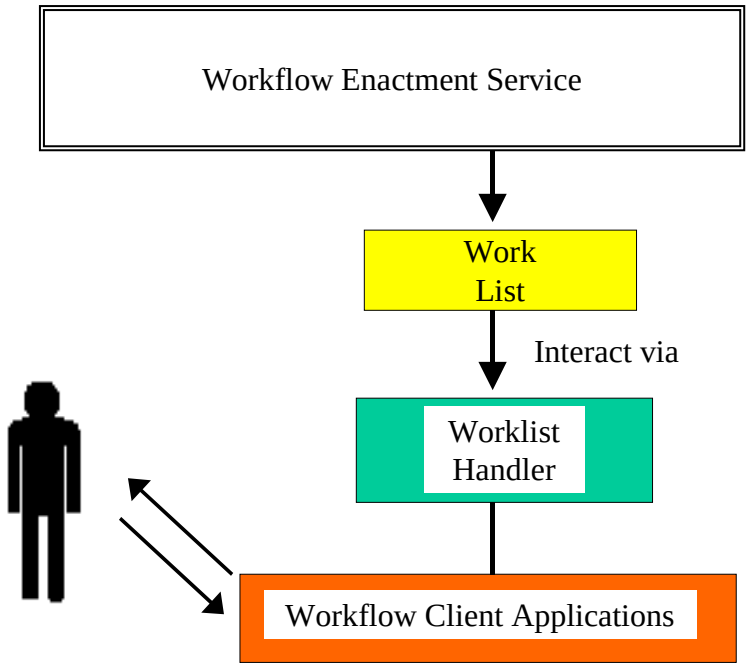

Figure 6: a generic structure for workflow client application

2.3.4 Invoked Applications

For invokeing external applications, WfMC's work focuses on developing a catalogue of interface type, together with a set of APIs for use in future workflow specific applications, as shown in Figure 7 [9]. *Application Agents* are introduced to contain a variety of method invocation behind a standard interface into the workflow enactment service. In the future, it is expected that *workflow enabled applications* will be available. For instance, DCOM or RMI may provide techical solutions for the implementation of Application Agents.

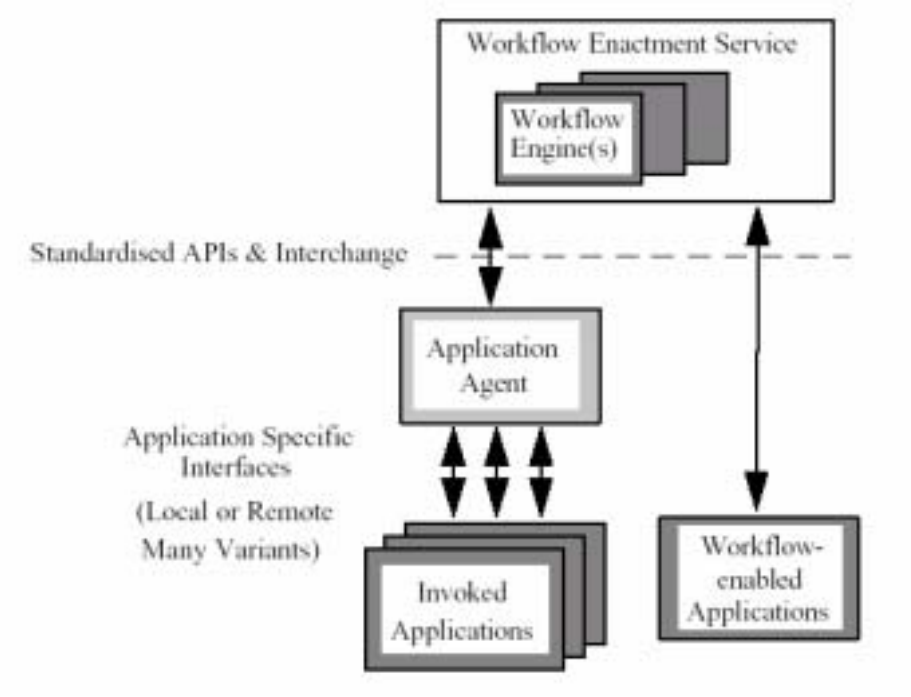

Figure 7: Invoked Application Interface

2.3.5 Workflow Interoperability

Allowing different workflow systems (heterogeneous or homogeneous) to cooperate seamlessly is a key objective of WfMC. For this purpose, a variety of interoperability scenarios have been developed. Interoperability is described at a number of levels, ranging from simple task passing to full workflow application interoperability with interchange of process definitions and workflow relevant data. The general nature of the information and control flows between workflow systems is shown in Figure 8 [9]. Also, the distributed object technology may provide techical solutions for implementation.

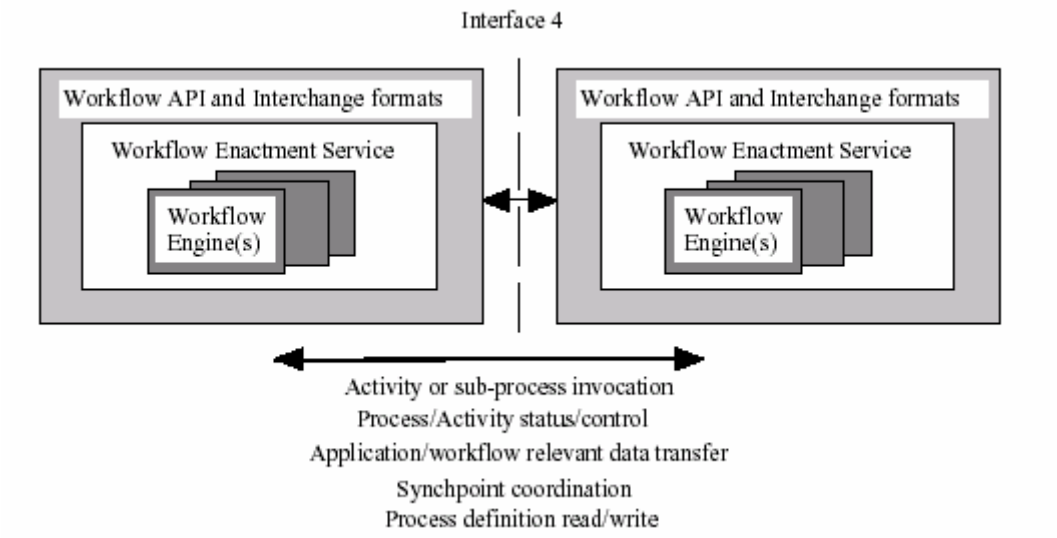

Figure 8: Workflow interoperability Interface

2.3.6 System administration & Monitoring Tools

The final area of the proposed reference model is a common interface standard for administration and monitoring functions which will allow one vendor's management application to work with another's engine(s). This will provide a common interface which enables several workflow services to share a range of common administration and system monitoring functions, as illustrated in Figure 9 [9].

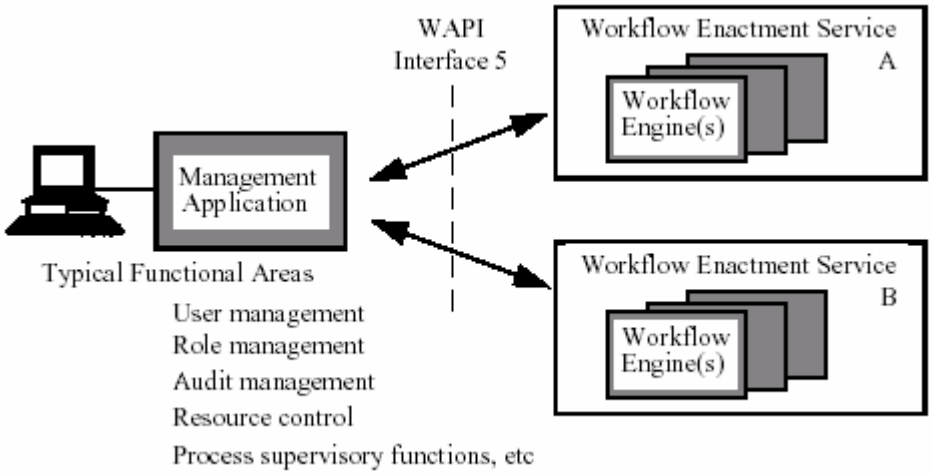

Figure 9: The Functions of Systems Administration & Monitoring Tools

2.4 The OMG Workflow Management Facility

The *workflow management facility* is proposed by Object Management Group (OMG) and is based on the Reference Model and architecture defined by Workflow Management Coalition (WfMC). With the workflow management facility [47], the OMG adds fundamental runtime functionality of a workflow management system to an ORB (Object Request Broker) environment. An ORB environment enables objects to communicate with each other independently of their location. Objects that communicate via an ORB can run in the same address space, in different address spaces, on different machines, or on different hosting environments. And they can be implemented with different programming languages. Each object which is to be available to other objects by means of an ORB must be described by metadata. The language used to specify this metadata is called interface definition language (IDL). The information that is specified to describe an object by means of IDL comprises all of its attributes, methods, the exceptions that might occur when a method does not perform successfully, etc. The most famous approach to ORBs is CORBA (Common Object Request Broker Architecture) introduced by OMG. Therefore, the workflow management facility proposed by OMG is based on CORBA and is represented with IDL. Based on the functions introduced in the specification, workflows can be started, suspended, or resumed, activities can be started or completed and workflow-related events can be recorded.

Below, we introduce the major interfaces of the workflow management facility briefly. Figure 10 shows these major interfaces and the relationships between them.

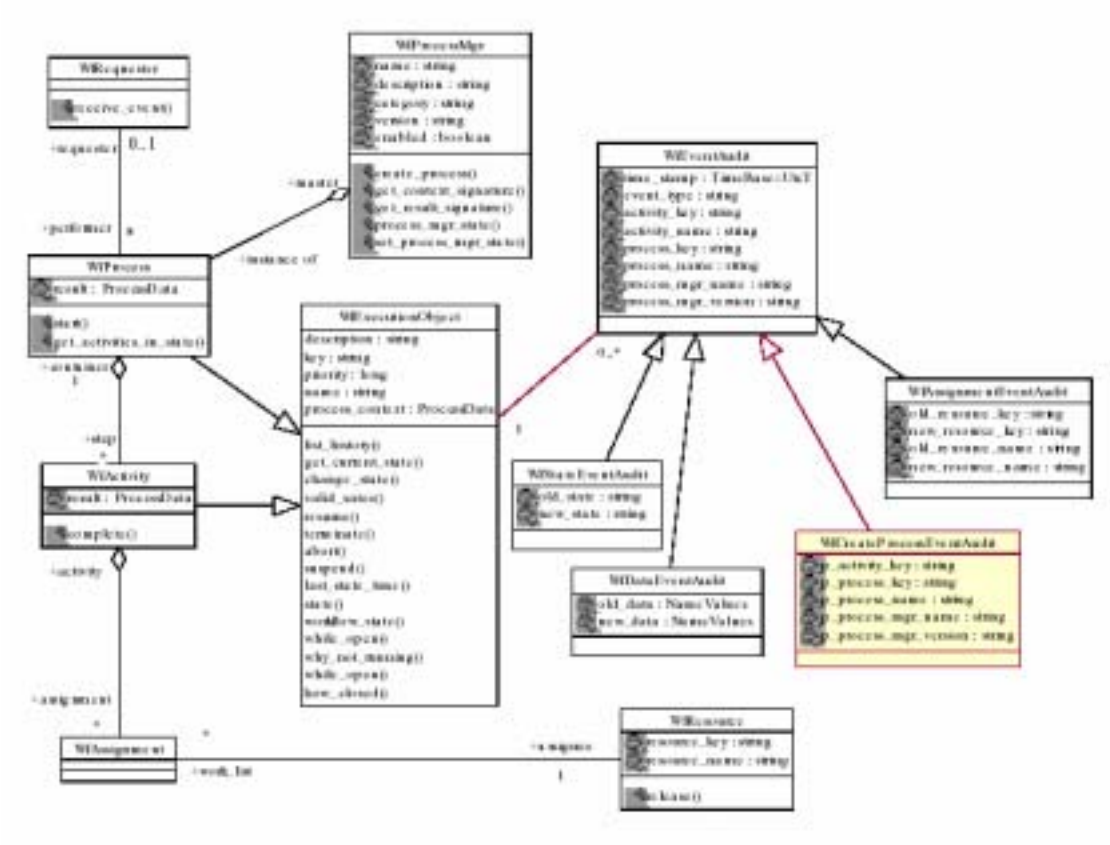

Figure 10: OMG Workflow Management Facility Model [47]

2.4.1 WfRequester Interface

`WfRequester` is the interface that has a direct concern with the execution and results of a workflow process - it represents the request for some work to be done. Its performer, a `WfProcess`, is expected to handle its request and communicate significant status changes; in particular to inform the requester when it has completed performing the requested work. A single requester can have many processes associated with it. Often `WfRequester` will also be the interface to the object that starts the process. As a process starter some of the control actions on the process include setting up the context, starting the process, and getting results and status.

2.4.2 WfExecutionObject Interface

`WfExecutionObject` is an abstract base interface that defines common attributes,

states, and operations for `WfProcess` and `WfActivity`. It provides the capability to get and set and internal states. Operations are provided to get the current state and to make a transition from the current state into another state. Operations are also provided for specific state transitions. These operations are suspend, resume, terminate, and abort. States returned by these operations should not be confused with the "state of the process" which is calculated by the top level `WfProcess`. States returned by these operations pertain only to the object they are returned from. For example, regardless of what activity is currently enabled, a process as a whole can be paused and resumed. The propagation of state change of a `WfProcess` object down to `WfActivity` objects or subprocesses is implementation and process definition dependent.

The interface includes name, description, priority, and key attributes. It also provides an operation for monitoring `WfExecutionObject` executions by returning, based on filter specified, event audit records that represent the history of the execution. Other operations include methods for getting and setting context.

2.4.3 WfProcessMgr Interface

A `WfProcessMgr` represents a template for a specific workflow process; it is used to create instances of a workflow process. Logically it is the factory and locator for `WfProcess` instances. It provides access to the meta information about the context a process requires and the result a process produces. Navigationg the relationship from `WfProcessMgr` to the `WfProcess` interface locates all actual instances of a workflow process. Invoking the `create_process()` method creates a new instance of a workflow process.

2.4.4 WfProcess Interface

A `WfProcess` is the performer of a workflow request. All workflow objects that perform work implement this interface. This interface allows work to proceed asynchronously while being monitored and controlled.

The `WfProcess` interface specializes `WfExecutionObject` interface by adding an operation to start the execution of the process, an operation to obtain the result produced by the process and relationships with `WfRequester` and `WfActivity`. In general, the context of a `WfProcess` is set when it has been created (using an appropriate `WfProcessMgr` factory) and before it is started. The context includes information about process navigation data, the resources to use, and the results to produce. An implementation of the WfM Facility may or may not allow updates of the process context after the process has been started.

A `WfProcess` is created (using a `WfProcessMgr`) by a user or automated resource and associated with a `WfRequester`. A `WfProcess` will inform its `WfRequester` about status changes such as modification of its state and its context using the requester's `receive_event` method. Execution of a `WfProcess` is initiated using the `start` method; execution can be suspended (and resumed) and terminated or aborted before it completes. While the work is proceeding, the `state` method on the `WfProcess` may be used to check on the overall status of the work. The `result` method may be used to request intermediate result data, which may or may not be provided depending upon the details of the work being performed.

2.4.5 WfActivity Interface

`WfActivity` is a step in a process that is associated, as part of an aggregation, with

a single WfProcess. It represents a request for work in the context of the containing WfProcess. There can be many active WfActivity objects within a WfProcess at a given point in time. The WfActivity interface specializes WfExecutionObject with an explicit complete operation to signal completion of the step, and with an operation to set the result of the WfActivity. It also adds relationships with WfProcess and WfAssignment. After activated, the instance of WfActivity may update the context via the set_process_context method, and the set_result method is used to feed back activity results into the process.

2.4.6 WfAssignment Interface

WfAssignment links WfActivity objects to WfResource objects. These links represent real assignments for enacting the activity. This interface may be specialized by resource management facilities that interpret the context of the activity to create and negotiate assignments with resources. Assignments are created as part of the resource selection process before an activity becomes ready for execution. The lifetime of an assignment is limited by that of the associated activity.

2.4.7 WfResource Interface

WfResource is an abstraction that represents a person or thing that will potentially accept an assignment to an activity. Potential and/or accepted WfAssignments are links between the requesting WfActivities and WfResource objects. It is expected that this interface will be used to implement adapters for objects representing people and things implemented in user, organization, and resource models. Navigating the work_item relationship fins all assignments associated with a resource. By use of the release method, a resource can be informed that it is no longer needed for a particular assignment.

2.4.8 WfEventAudit Interface

`WfEventAudit` provides audit records of workflow event information. It provides information on the source of the event and contains specific event data. Workflow events include state changes, change of a resource assignment, and data changes. Workflow events are persistent and can be accessed navigating the history relationship of a `WfExecutionObject`. Workflow audit event objects are not part of the persistent state of their source workflow object.

A workflow event audit object is created when a workflow object changes its status (state change, process data change or assignment change); its lifetime is not limited by the lifetime of the event source object. Operations for managing the retention, archiving, and deletion of workflow events are not specified in this specification. The `WfEventAudit` defines a set of event properties common to all workflow audit events. In particular, it provides an identification of the source of the event in terms of (business) identifiers of the workflow entities `WfProcessMgr`, `WfProcess`, and `WfActivity`.

WfMC's reference model provides a high level abstraction for workflow management system. And OMG's workflow management facility provides more details for implementation aspect. The workflow management facility provides selected functionality of the client application interface (interface 2), the interoperability interface (interface 4), and the administration and monitoring interface (interface 5) of the reference model as an object-oriented framework in an ORB environment. However, OMG's workflow management facility does not standardize a metamodel for modeling business process or the exchange of models of business processes. Thus, the interface 1 of WfMC's reference model has not been addressed. And the invoked

application interface (interface 3) is also not specifically addressed.