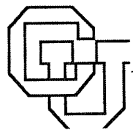
A Framework for Workflow Architectures

Kwang-Hoon Kim
Clarence A. Ellis

CU-CS-847-97



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR(S) AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE AGENCIES NAMED IN THE ACKNOWLEDGMENTS SECTION.

A Framework for Workflow Architectures

Kwang-Hoon Kim Clarence A. Ellis

CU-CS-847-97 OCTOBER 1997

University of Colorado at Boulder

Technical Report CU-CS-847-97
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309-0430

A Framework for Workflow Architectures

Kwang-Hoon Kim Clarence A. Ellis

October 1997

Abstract

This paper defines a multiple-level framework for the description of workflow management systems focusing on their software architectural aspects. The framework, consisting of generic-level, conceptual-level and implementation-level, provides common architectural principles for designing a workflow management system. Based on the framework, we suggest a new conceptual taxonomy for workflow management systems.

The taxonomy is formed by considering possibilities for centralization or distribution of data, control, and execution; along with considering how the major components of a workflow system, such as activities, roles, actors, and workcases, are represented and manipulated.

We believe that this framework and taxonomy are a significant contribution because they add clarity, completeness, and "global perspective" to workflow architectural discussions. The vocabulary suggested here includes workflow levels and aspects, allowing very different architectures to be discussed, compared, and contrasted. Added clarity is obtained because similar architectures from different vendors that used different terminology and techniques can now be seen to be identical at the higher levels. Much of the complexity can be removed by thinking of workflow systems in terms of the dimensions introduced here. The framework categorizes existing architectures and suggests a plethora of new architectures. As an example, we introduce the "Dispersed Actor-Oriented Workflow Architecture" as a promising new architectural structure suggested by our taxonomy.

The framework and taxonomy can be used for sorting out gems and stones amongst the architectures possibly generated. Thus, it might be a guideline not only for characterizing the existing workflow management systems, but also for solving long-term and short-term architectural research issues, such as dynamic changes in workflow, transactional workflow, dynamically evolving workflow, large-scale workflow, etc., that have been proposed in the literature.

Key words: *Workflow Management Systems, Workflow Enactment Engines, Workflow Management Architectures, Architectural Framework, Software Architecture*

1. Introduction

The design and implementation of workflow management systems is typically a large and complex task. Decisions need to be made about the hardware and software platform, the data structures and algorithms, and the interconnection of various modules used by various users and administrators. These design decisions are further complicated by requirements such as scalability, robustness, speed, and usability. In this document, we describe an architectural framework conceived to be helpful in the design and implementation of workflow, and in the understanding of the spectrum of possibilities for workflow architecture.

The framework developed herein is primarily concerned with workflow enactment engines, and with their software architecture. However, the PhD thesis being completed in this area [31] generalizes the notions that we present here, and in fact, the low-level architecture which we define in a later section is concerned with hardware and communication architecture as well as software. In general, the main reason that the concept of software architecture has started emerging in the software literature is that the software design problem goes beyond the algorithms and data structures of the computation as the size and complexity of software systems increases. In other words, the software architecture level of design in system development processes has become an important step when designing complex systems. So far, there have been several well-defined terms and notations to characterize architectural structures in the software development field.[13]

In the meantime, there has been an implicit body of work on the software architecture of workflow management systems. Usually, a particular notation is used to describe a particular workflow management system. So, not much progress has been made in developing a systemically well-defined terminology, notation and classification to characterize architectural structures in the workflow literature. Our concern in this paper is restricted to the workflow enactment part of a workflow management system, which typically has capabilities to create, manage and execute workflow procedure's workcases [14][16][30]. Many of the workflow architecture descriptions in the open literature represent only the implementation level details. [1][4][5][10][18] Others are simply documented at the conceptual level.[19][24][28]

This paper tries to systematically formulate a way (framework) to describe and design workflow management architectures, and suggests a conceptual-level taxonomy of workflow management architectures to grasp many kinds of architectures for workflow management systems including current existing systems as well as future systems. The purpose is to illustrate the current state of the workflow architecture work, and to examine ways in which architectural design can impact future workflow management systems. Future architectures will need to support highly advanced workflow features, like dynamic changes, transactional workflow management, and large scale workflow management. Our presentation is especially aimed toward showing possible future architectures that may be useful to support some of these advanced workflow features.

In the following sections, we define an architectural framework, consisting of generic-level, conceptual-level and implementation-level, to systematically form a workflow management system, and provide some architectural considerations that should be described on each level. Next, we suggest a conceptual-level workflow architectural taxonomy. Finally, a promising new architecture called "Dispersed Actor-Oriented Workflow" is presented in the end of the paper.

2. An Architectural Framework

This section introduces an *architectural framework* and *principles,* useful when designing a workflow management system. It suggests three levels of architectural description: *Generic-level, Conceptual-level* and *Implementation-level*. We specify what contents or criteria should be covered in each level as shown in Figure 1.

It is manifest that the development of an efficient workflow management system requires common architectural principles because of the following reasons:

- Common paradigms are quite important so that high-level relationships among different workflow systems can be understood and so that new systems can be more clearly distinguished from old systems.

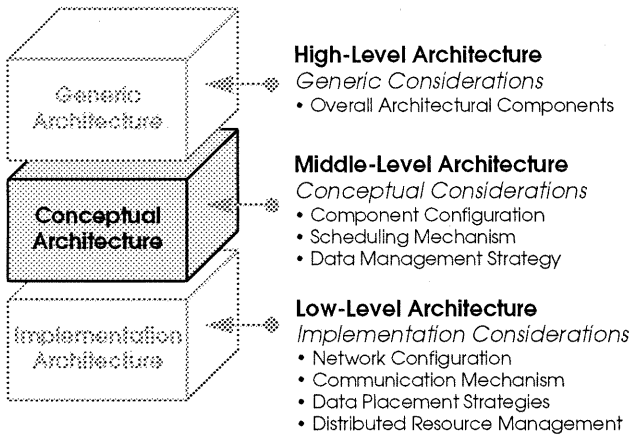


Figure 1. Architectural Framework for Workflow Management Systems

- A workflow management system is complex. So, a well-defined as well as a well-leveled architectural system description is essential to analyze important properties of a workflow management system.
- Detailed grasping of multiple-level workflow management architectures enables principled choices among design alternatives, and paves the path for accurate workflow comparison and benchmarking.
- Getting the right architecture is often crucial to the success of a workflow management system design; An incompatible architecture can cause significant problems.

In Figure 1, some considerations that should be taken into account on each level of the framework are introduced. Of course, important considerations other than the specified on each level may be worth deliberation. But, the specified considerations are essential points to be duly considered for the design and implementation of a workflow management system. Especially, the creation of a conceptual-level architecture is a most important phase, because the architecture created on this level is the representative architecture for the workflow management system.

3. Generic-level Considerations

At this level, a common high-level framework from which to view the architectural styles of workflow management systems is defined. The framework allows us to view an architecture of a specific workflow management system as a collection of ***generic components*** - or simply *dimensions*. Then, an architecture defined in this level represents a family of systems, each member of the family being defined differently at the middle and low levels.

More specifically, a generic architecture is defined by specifying its control, script, and data components as each being centralized, decentralized, or dispersed. Thus each of the boxes in the 3X3X3 cube of FIGURE 2 represents a different generic architecture. In other words, the generic functionality of a workflow management system consists of three dimensions and a structural pattern on each of them:

- *a pattern of control structure*: Who makes the control of decisions or schedule for workflow enactment? Who keeps the control information?
- *a pattern of data structure*: Where keeps the application data? Who works on them?
- *a pattern of execution structure*: Where are workflow scripts stored? Who executes them?

As illustrated in FIGURE 2, the structural pattern of each dimension can be: Centralized, Decentralized or Fully dispersed.

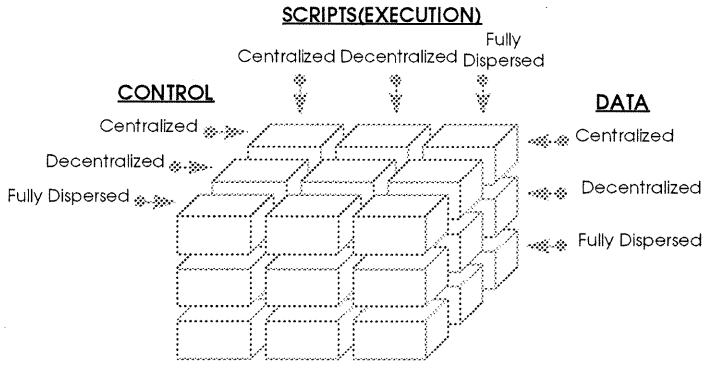


Figure 2. Dimensions for Generic Architectures

- Centralized: Data, control and execution are kept on a single site.
- Decentralized: Data, control and execution are kept on a fixed number of sites that are pre-determined.
- Fully Dispersed: Data, control and execution are (dynamically) kept on several sites and by active components.

Many hybrid mixes of the above structural components on each dimension are possible.

3.1 Typical Generic Architectures

FIGURE 3 shows a set of generic architectures that are appropriate for the design of a workflow management system. When we consider the current workflow management products, almost all fall into one of three of these generic architectural categories: mainframe (centralized in all aspects), client-single server (decentralized script execution, centralized control and decentralized application data), and client-multiple servers (decentralized script execution, decentralized control and decentralized application data).

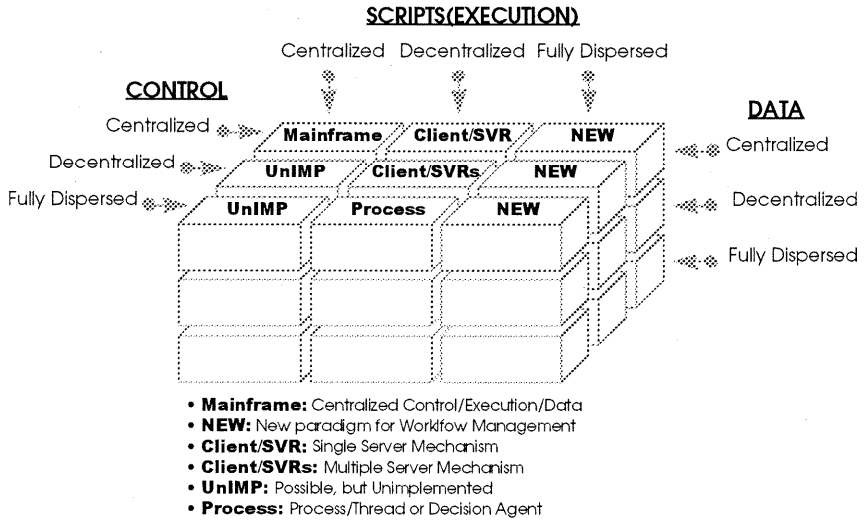


Figure 3. Typical Generic Architectures

Highly related work is the standard proposed by the Workflow Management Coalition [30]. This work is primarily concerned with defining interfaces between the workflow enactment engine and other external entities of clients, invoked applications, administration, monitoring tools, definition tools, and other workflow systems. This work defines workflow terminology, but is not directly concerned with the space of workflow enactment architectures.

4. Conceptual-level Considerations

As presented in the Figure 1, the conceptual level is more detailed than the generic level, but is not concerned with concrete low level details (e.g. nodes and network connectivity) of the implementation level. This level presents a detailed conceptual picture of how scheduling and data storage and access work. An architecture defined at this architectural level is a specialized description of an instance of a generic architecture. Therefore, the conceptual distinctions among workflow management architecture details are specified on this level.

Each box in Figure 4 represents a different conceptual architectural possibility. At the conceptual level architecture, the following considerations should be basically considered to characterize and classify each of the different architectures:

a pattern of control for enactment schedules or decisions

major components in the workflow model

a type of process structure for each of the model components to be concretized in a conceptual architecture.

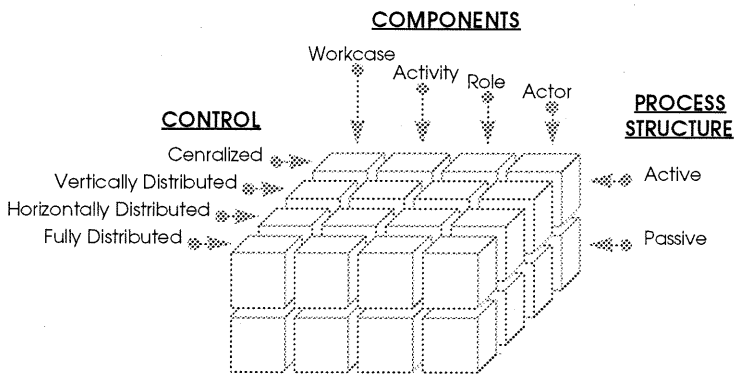


Figure 4. Dimensions for Conceptual Architectures

The first dimension of the conceptual model is the control dimension. The pattern of control consists of four possibilities in terms of the type of distribution: Centralized, Horizontally Distributed, Vertically Distributed and Fully Distributed control of workflow enactment. The vertically distributed control pattern means that multiple copies of an architectural component, of which take the control functionality in charge, exist on a conceptual architecture. The horizontally distributed control pattern means that the functionality of control is putting in charge of multiple architectural components on a conceptual architecture. The fully distributed control pattern is the combined pattern of both of the previous patterns.

The second dimension of the conceptual model is the components dimension. We advocate an open ended list of possible components. The list is something that may expand in the future as workflow systems encompass social and organizational aspects more comprehensively. A list of components could, for example, include a temporal component, an information component, a social component, a resource component, and others. In this paper, we consider only major components identified by the Workflow Management Coalition, and defined in standard workflow definitions [30]. Thus, the major components in our conceptual workflow model that are able to be either active or passive are workcases, activities, roles and actors.

The third dimension is the process structure. The type of process structure consists of an active process structure type and a passive process structure type. Some of the workflow model components are active if they are embodied as processes or threads in a conceptual architecture. While on the other, they are passive if they are represented as data in a conceptual architecture.

In this section, after more fully explaining about those dimensions, a considerable body of work on the conceptual level taxonomy of workflow management architectures will be done. At the same time, which of the current available workflow management architectures should belong to which branch of the taxonomy will be examined.

4.1 Detailed Explanations of the Conceptual-level Dimensions

Generally speaking, there are several criteria or dimensions being used to characterize architectures in terms of conceptual aspects. The optimal set of criteria, which is completely grouping architectures and fixing some common properties of each architecture, has been sought through the architectural framework. That is the very three dimensions presented in the Figure 4 and explained in the conceptual level of the framework: Components of the workflow model, a pattern of workflow enactment control and a type of process structure.

4.1.1 Active-Model *vs.* Passive-Model

The workflow model is able to be well represented by the entity-relationship model as shown in Figure 5. Among the components of the workflow model, the following five are what the control part of a workflow management system has to handle.

- An Actor is a person, group, or computing facility that can fulfill to execute, to be responsible for, or to be associated in some way with activities, procedures, resources, and other actors. So, actors have some dependent relationships each other, based on the execution sequence of their activities.
- An Activity is a work step of a procedure. An activity is either a compound activity, containing another procedure, or an elementary activity. each activity has associated with transition conditions, invoked applications, input/output repository, and jobs.
- A Role is a named designator for an actor, or a grouping of actors which conveniently acts as the basis for access control and execution control.
- A procedure is a predefined set of work steps, and a partial ordering of these steps. A work step consists of a header(identification, precedence, etc.) and a body(the actual work to be done).
- A Job is the locus of control for a particular execution of a procedure. The job is called a workcase; if a procedure is considered a Petri net, then a job is a token or related set of tokens flowing through the net. if the procedure is an object class, then a job is an instance.

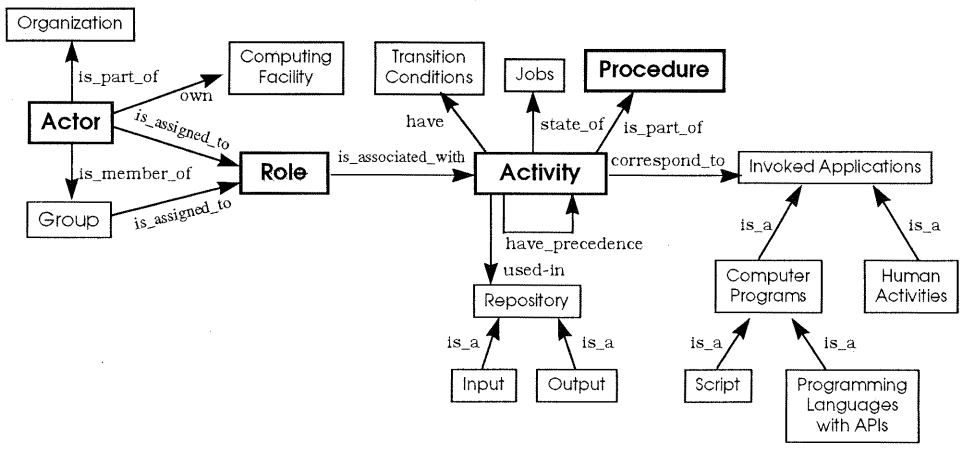


Figure 5. The Workflow Model

In the conceptual architectures, it is so crucial how these are transformed or embodied onto an architecture. In other words, they can be alive as an active component in an architecture. Or, they can just be an passive component, like data stored on database. The former is called active-model, the later is

called passive-model. Suggestively speaking, almost of the current commercialized workflow management systems are based on the passive-model. It is possible to compare their properties to each other as following:

- the configuration of a conceptual architecture is flexible in an active-model, meanwhile it is firm in an passive-model.
- while, in an active-model, an architecture is reflective of the structures of workflow procedures, in an passive-model, architectural components are organized by taking its functionality into consideration.
- For an instance, a workflow procedure is able to be realized, as a process or a thread, in a active-model workflow management architecture. While on the other, in the passive-model workflow management architecture, it is stored as data.

Again, the active-model concept should be divided into three sub-concepts: class-active, instance-active and hybrid, according to how workcases from a procedure are reflected in an architecture. In the class-active case, workcases are represented as a passive component. Workcases are becoming active components of an architecture in the instance-active case. The hybrid case is to combine both the class-active and the instance-active. The detailed explanation about these active-models and the typical architectures acting for each of the cases will be presented in the next section.

4.1.2 Workcase-based, Activity-based, Role-based vs. Actor-based

This dimension is based on how the workflow model is represented in a workflow management system. The representation is completely different in the active-model from in the passive-model workflow architecture.

In an active-model workflow management architecture, a workflow model is able to be embodied as an active component. At this time, it is possible for the embodiment to be accomplished through either a class-active pattern or an instance-active pattern. Of course, the hybrid pattern of both the class-active and the instance-active works out, too.

In the class-active pattern, the following alternatives are possible:

- Procedure-based embodiment: whenever a procedure is created, it becomes an active component.
- Activity-based embodiment: whenever a procedure is created, it is represented into a number of active components corresponding to the activities being included in itself.
- Role-based embodiment: whenever a procedure is created, it is represented into a number of active components corresponding to the roles being associated within itself.
- Actor-based embodiment: whenever a procedure is created, it is represented into a number of active components corresponding to the actors being associated with itself.
- Hybrid embodiment: whenever a procedure is created, it is represented into a number of active components corresponding to any combinations of four components.

In the instance-active pattern, the following alternatives are possible:

- Workcase-based embodiment: whenever a workcase is created, it becomes an active component.
- Activity-based embodiment: whenever a workcase is created, it is represented into a number of active components corresponding to the activities being included in itself.
- Role-based embodiment: whenever a workcase is created, it is represented into a number of active components corresponding to the roles being associated within itself.
- Actor-based embodiment: whenever a workcase is created, it is represented into a number of active components corresponding to the actors being associated with itself.
- Hybrid embodiment: whenever a workcase is created, it is represented into a number of active

components corresponding to any combinations of four components.

In an passive-model workflow management architecture, a workflow model is used to be transformed into the one of following alternative patterns:

- Workcase-based transformation: whenever a workcase is created, it is stored as data.
- Activity-based transformation: whenever a workcase is created, it is stored as data after being transformed into a set of activity data.
- Role-based transformation: whenever a workcase is created, it is stored as data after being transformed into a set of role data.
- Actor-based transformation: whenever a workcase is created, it is stored as data after being transformed into a set of actor data.
- Hybrid transformation: whenever a workcase is created, it is stored as data after being transformed into a set of combination data of the four components.

4.1.3 Centralized, Vertically Distributed, Horizontally Distributed vs. Fully Distributed

A pattern of distribution structure for the workflow enactment control and schedule must be able to become one of dimensions for the conceptual level taxonomy of workflow management architectures. There must be a component, playing the role of workflow enactment control and schedule, in a workflow management system. Roughly speaking, it is not too much to say that the component discharges almost of all functions of a workflow management system. The component is able to perform the scheduling work in a different fashion.

The first one is that the scheduling work for all instances(workcases) of a procedure is done by a single performer; The second is that the work is done by multiple performers each of which has an exactly same functionality. So, each performer takes over a portion of workcases and schedules their enactment; The third is that the work is completed by the cooperative work of several performers each of which has a different functionality. That is, this is to allot a portion of the work to each performer. So, each of the performers fulfills its own role or duty in the scheduling work for all workcases of a procedure; The final one is that the work is done by the combined fashion of both the second and the third. These scheduling fashions are called the centralized, the vertically distributed, the horizontally distributed and the fully distributed workflow enactment scheduling mechanism, respectively.

The distribution structures for controlling and scheduling of the workflow enactment are able to be differently realized in the active-model and the passive-model.

In a passive-model workflow management architecture, the distribution structures are realized as following:

Centralized: There is a server fully discharging the workflow enactment control and schedule for all workcases out of a procedure. Almost of all workflow management systems currently available in the market belong to this category of the distribution structure.

Vertically distributed: There are multiple copies of a server taking fully responsibility for the work. So, a portion of workcases being produced from a procedure is assigned into each of them. In this case, they need not to cooperate or communicate each other to do the work.

Horizontally distributed: There are several servers taking partial charge of the work. So each sever has an exclusive responsibility on its allotted task out of the workflow enactment control and schedule work for all workcases from a procedure. There should be a kind of cooperation and communication among the servers to accomplish the work.

Fully distributed: In this case, workcases belonging to a procedure are handled by a combined fashion of the vertical and horizontal distribution. That is, multiple copies of servers are going to be exclusively discharging their own allotted tasks which are divided from the workflow enactment control and schedule

work. As a result, a portion of workcases should be assigned into a set of servers needed to complete the scheduling work.

In an active-model workflow management architecture, the workflow model itself and its component are a key subject doing the workflow enactment control and schedule work. The distribution structures are realized as following:

Centralized: This is the case of that a procedure or its workcases perform the workflow enactment controlling and scheduling work. In this case, a workcase-based active-model component in an architecture is used to be fully discharging all about the corresponding procedure such as monitoring, enactment controlling and scheduling for all workcases out of the procedure. Therefore, one workcase-based active-model component must be created whenever a procedure is defined in a workflow management system.

Vertically distributed: There are multiple copies of the workcase-based active-model component, corresponding to a procedure, in an architecture. So, a portion of workcases being produced from the procedure is assigned into each of them. In this case, they need not to cooperate or communicate each other to do the work.

Horizontally distributed: This is the case of that the components but procedures and workcases, such as activities, roles and/or actors, of the workflow model have the work controlling and scheduling the workflow enactment. For an example, an activity-based active-model component may be exclusively responsible on all about the corresponding activity. Therefore, there must be a kind of cooperation and communication among those components to accomplish the work.

Fully distributed: This is a case of that an architecture is configured with a combined fashion of the vertical and horizontal distribution.

Of course, the detailed realizations of these distribution structures are a little different in a class-active, an instance-active and a hybrid architecture.

4.2 Architectural Taxonomy of the Current Workflow Management Systems

So far, quite a few workflow management systems have been developed and commercialized in the workflow literature. Figure 6 is to present the conceptual-level architectural taxonomy for some of the current available workflow management systems. In the taxonomy, only two dimensions, the process structure and the pattern of workflow enactment control, are considered, because the dimension of the model's component is a little effective in the passive-model conceptual architecture on which the components are represented as data. In other words, the dimension becomes effective in doing taxonomy for the active-model conceptual architectures. But, there have been proposed a few workflow management systems professing an active-model conceptual architecture.

Almost all commercialized products, such as InConcert, Staffware, FloWare, Notes, ActionWKF, FlowWorks and FlowMark workflow management systems, are adopting a platform of the client-server architecture with a database management system[31]. It may safely be said that those systems are categorized into the passive-model and centralized conceptual architecture. Especially, the FlowWorks workflow management system[1] made by BULL and the FlowMark workflow management system[20][21][22] produced by IBM are typical examples for the conceptual architecture of passive-model, centralized enactment control pattern and workcase-based representation.

There is a good examples for the passive-model, vertically distributed and workcase-based conceptual workflow architecture. D. Alonso and et al[5] at IBM proposed an extended version of the FlowMark workflow management system, which is originally designed to enhance availability of the workflow system by handling failures. That is called a clustered server architecture for FlowMark. The architecture consists of several clusters, each of them with its own database on which the same workflow procedure information is stored, so that workcases out of a workflow procedure are disseminated into and handled by these clusters.

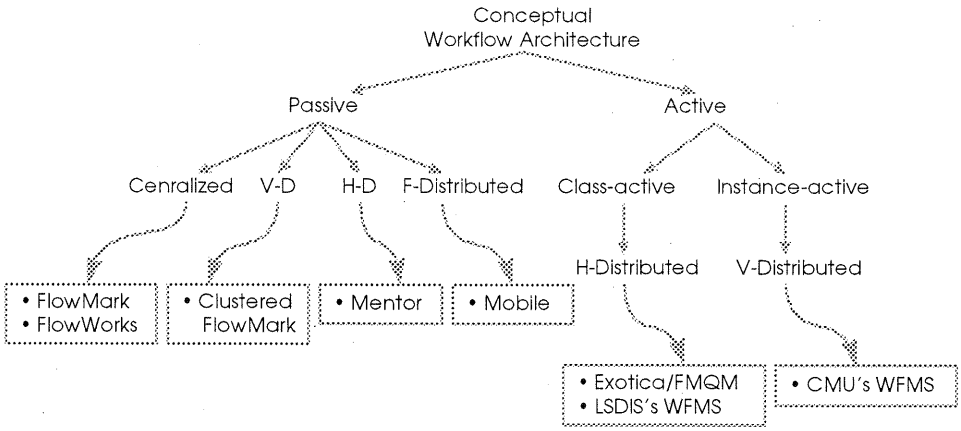


Figure 6. Architectural Taxonomy of the Current WFMS

There is a typical workflow management system being fitted into the passive-model and horizontally distributed workflow conceptual architecture. That is the mentor workflow management system's architecture proposed by Jeanine W. et al[17]. It is designed so as to allow for diversity and for scalability by an approach in which a workflow procedure can be partitioned into a number of subworkflows, each of which is able to be handled by a different server.

The typical workflow management system belonging to the passive-model and fully-distirbuted workflow conceptual architecture is the mobile workflow management system which has been proposed by Stefan J.[28]. It supports the specification of workflows through several modeling perspectives, such as functional, behavioral, informational, operational and organizational perspectives. Each of the perspectives is represented into a passive component and handled by the corresponding server on the architecture. And servers corresponding to the perspectives are able to be replicated according to the scale of workflow procedures and their workcases.

According to our literature survey, there have been three proposed workflow management systems professing the active-model conceptual workflow architecture. These are just Exotica/FMQM, a persistent message-based architecture for distributed workflow management by IBM[4], LSDIS's workflow architectures, CORBA-based enactment architectures for workflow management systems by the large scale distributed information systems Lab At the university of Georgia, and CMU's workflow architecture, a distributed, mediated architecture for workflow management by the software engineering institute at the carnegie mellon university[19].

In the Exotica/FMQM persistent message-based architecture, a workflow procedure is divided by several subworkflows, each of which is concretized as an active components in the architecture, and the active components are located into several nodes in network. Each of the active components will be in charge of coordinating the execution of workcases out of the corresponding subworkflow by examining the queue for incoming messages and creating an activity thread to manage the execution of the activity. Therefore, this workflow management system is a typical example of the class-active-model, horizontally distributed and active-based conceptual workflow architecture.

A.P. Sheth and his colleagues at LSDIS Lab have proposed five conceptual workflow enactment architectures for implementing a workflow management system. The fully distributed workflow enactment architecture among them is an another example for the class-active-model, horizontally distributed and activity-based conceptual workflow architecture, because an activity manager in the architecture is equipped with an individual scheduler(without a centralized scheduler) and controls the enactment of workcases through asynchronous IDL interfaces providing the communications among activity managers.

CMU's static and dynamic workflow architectures developed by K. Wallnau and et al are examples of the instance-active-model, vertically distributed and workcase-based conceptual workflow architecture. Because one of the key ideas expressed in the architecture is executing workflow engines represent the

instantiation of a workflow procedure description(model), and a single workflow procedure may be simultaneously instantiated by several workflow engines.

So far, the conceptual-level taxonomy for the current available workflow management architectures has been done according to the dimensions of conceptual-level framework. We believe that the active-model architectures, especially the instance-active-model architecture, should be more appropriate for satisfying the advanced workflow requirements, such as scalability, availability, and extensibility, to solve long-term and short-term architectural research issues like dynamic changes in workflow, transactional workflow, dynamically evolving workflow, large-scale workflow. But, there might be only a few workflow management systems adopting a platform of the instance-active-model. So, in the next section, we introduce the "Dispersed Actor-Oriented Workflow Architecture" as a promising new architectural structure suggested by our taxonomy. That is, it is categorized into the class-active-model, fully-distributed and actor-based conceptual workflow architecture.

4.3 The Dispersed Actor-Oriented Workflow Architecture

Figure 7 presents a conceptual-level architecture for the dispersed actor-oriented workflow management. This architecture is based on the generic workflow architecture described in the architectural framework, and the workflow reference model of the WfMC (Workflow Management Coalition). The novel idea of this architecture is to create one (or more) process for each actor. The key ideas of the architecture are the following:

An actor-oriented workflow model [31] is generated from a workflow procedure, which has been defined by the workflow definition subsystem. This model is designed to partition all workflow control information into sub-schema, and one sub-schema is distributed to each actor process at the beginning of enactment time.

During enactment, a workcase migrates from actor process to actor process. Each activity in the entire procedure is performed by one of the actor processes. The assignment of activities to actors is done dynamically and distributedly during each step of enactment.

The set of actor processes associated with a procedure are called a workflow engine cluster. Several actor-oriented workflow engine clusters for a workflow procedure are possibly created in order to enhance the availability of workflow enactment services. So, different workcases of a single workflow procedure may be simultaneously created by several actor-oriented workflow engine clusters.

Communication among actor-oriented workflow engines occurs through asynchronous message queues [4][19].

The concept of bridge, which is used to provide the interoperability between different products or platforms, is able to be constructed on each of the interfaces between the workflow enactment service and other components of the architecture.

The conceptual key point of the dispersed actor-oriented workflow architecture is on the partitioning logic by which a workflow procedure execution are partitioned into different actor-oriented workflow engines handled by the workflow enactment service. That is, the partitioning logic of the architecture is the actor assignments onto activities.

In general, the assignment of roles into an activity is fulfilled at the model definition time, which is called 'the build time' in some workflow management systems, but a real actor out of actors who are associated with the role is selected right after the activity is enacted by a scheduler or an enactment controller in the model execution time (or the run time). We would refer to the actor assignment scheme like this as 'the late actor binding' scheme. As the opposite scheme, we propose 'the early actor binding' scheme.

In the early actor binding scheme, the actor assignments into all activities of a workcase or an instance of a workflow procedure are fulfilled right after a workflow procedure is created, or the workcase or the instance is created. The former is called the static actor assignment, the latter is called the dynamic actor assignment. Then, it should be expected to do the efficient assignment of actors. These actor assignment

schemes are able to be implemented by constructing a static actor-oriented model and a dynamic actor-oriented model, respectively.

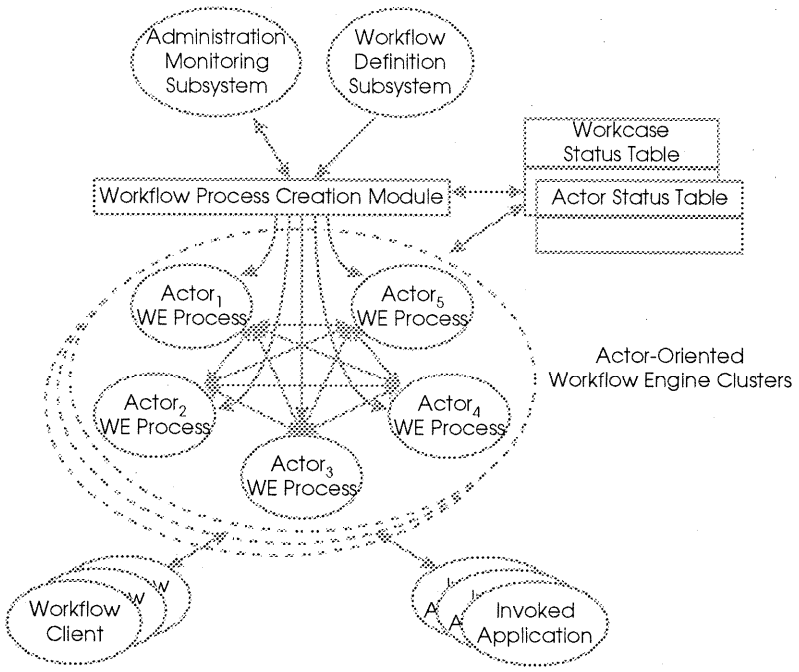


Figure 7. Dispersed Actor-Oriented Conceptual Workflow Architecture

A static actor-oriented model is constructed only once during the whole life of a workflow procedure. So, an actor assigned to specific activities of a workflow procedure has to perform all workcases or instances corresponding to the activities until the assignment is changed by the administrator. However, a dynamic actor-oriented model is able to be automatically constructed whenever a workcase or instance is created. Therefore, it is possible to accomplish a dynamic actor assignment based on a workcase or instance of a workflow procedure.

For the sake of the implementation aspects for the dispersed actor-oriented workflow architecture, it is possible to use some of the implementation alternatives and considerations which are briefly described in the next section.

5. Implementation-level Considerations

Until now, the possible spaces of the generic and conceptual architectures for a workflow management system have been defined. This has resulted in a large number of architectural options. In fact, the implementation generates even a larger number of options, because there are many possibilities that emerge as we consider the physical, rather than the conceptual components of our workflow system. Thus, at the conceptual level, we were concerned with (abstract) processes and abstract data types; at the implementation level, we are concerned with (concrete) processors, workstations, and networks. Tasks and data entities that conceptually are indivisible now need to be time-shared, or broken into smaller data sets. At the implementation level, it is necessary to specify which algorithms and data structures will reside on which users' workstations, and to give full consideration to techniques of load balancing and data caching and other techniques of resource management. The resource management alternatives for homogeneous distributed computing environments include a distributed file system, a distributed shared memory system, and a distributed scheduler for the load sharing functionality. On the other hand, the object request brokers, like DOMS (distributed object management system) and CORBA (common object request broker architecture), are alternatives for heterogeneous distributed computing environments. Full consideration of the implementation level is beyond the scope of this paper.

6. Conclusions

So far, hundreds of vendors in the software market have been claiming that their products have workflow management capabilities. And, based on these products, a lot of corporations are trying to conduct business procedure reengineering to automate their business procedures.

The high-level architectures for those products can be converged into some of generic architectures shown in Figure 2. At the same time, In terms of the conceptual aspects, those products are categorized into one of the conceptual architectures represented in Figure 4. They are also able to be distinguished from each other through how their workflow management system is implemented.

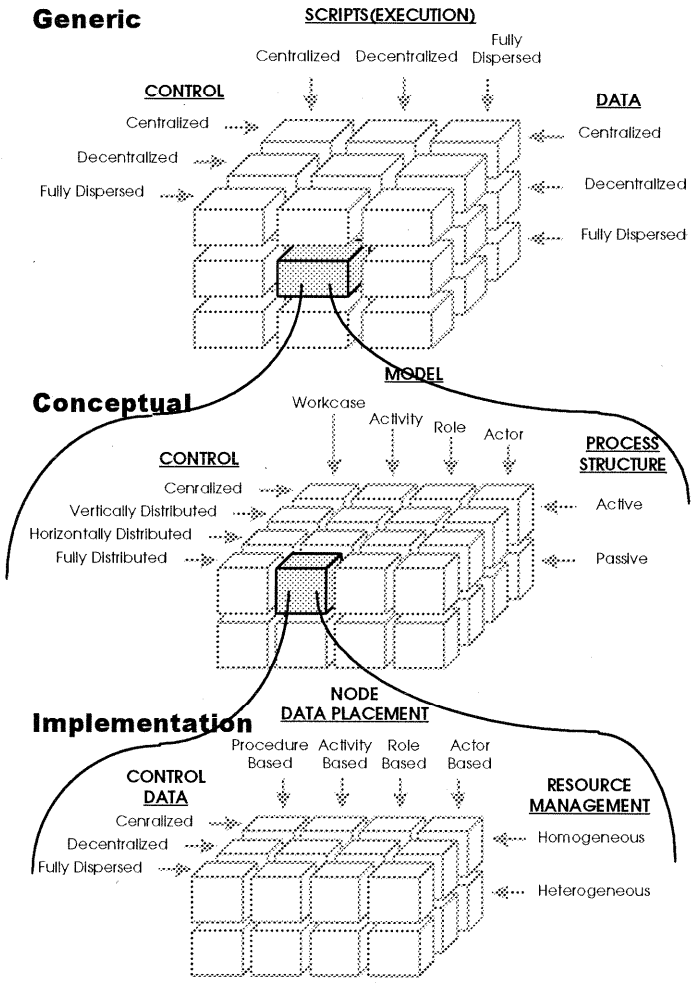


Figure 8. A Hierarchical Framework

In summary, the framework for drawing an architecture for a workflow management system has a hierarchical structure as shown in Figure 8. The first step of the framework is to describe the major interfaces and coordination policy for components of an high-level architecture. The outcome of this step is a generic workflow management architecture that must be independent of underlying implementation technology, but that must fix up the scope of implementation possibility by reflecting the current or future technologies. The main purpose of the second step is to refine the generic architecture in component-by-component by taking workflow-specific properties into consideration. Therefore, the workflow model, a pattern of workflow enactment control structure, and how each component of the workflow model is transformed and embodied onto a workflow architecture must be described in a conceptual architecture.

Finally, in the third step, a technology-specific realization of the conceptual architecture must be speculated for an implementation level workflow management architecture.

After all, the architectural framework should not only be used to characterize and evaluate the existing workflow management systems, but also play an important role as a guideline for the developers trying to design and implement a new workflow management system.

7. References

1. "FlowWorks, The Bull Workflow Product: Architectural Design and Functional Specification", BULL L.P.M., Oct. 1991
2. Akhil Kumar & J. Leon Zhao, "A Framework for Dynamic Routing and Operational Integrity Controls in a Workflow Management System"
3. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Gunthor and C. Mohan, "Advanced Transaction Models in Workflow Contexts", IBM Research Report
4. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Gunthor and C. Mohan, "Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management", Proceedings of IFIP Working Conference on Information Systems for Decentralized Organizations, Aug. 1995
5. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Gunthor and C. Mohan, "Failure Handling in Large Scale Workflow Management Systems", IBM Research Report RJ9913, Nov. 1994
6. Amit Sheth, "Workflow Automation: Applications, Technology and Research", Tutorial Notes, SIGMOD Conference, May 1995
7. Clarence A. Ellis, "Workflow Technology", Tutorial Notes, 1995
8. Clarence A. Ellis and Marc Bernal, "Officetalk-D: An Experimental Office Information System", Technical Report, Xerox Palo Alto Research Center
9. Clarence A. Ellis, Gary J. Nutt, "Office Information Systems and Computer Science", ACM Computing Surveys, Vol. 12, No. 1, March 1980
10. Clarence A. Ellis and Carlos Maltzahn, "Chautauqua: Merging Workflow and Groupware", Jun. 1996
11. Clarence A. Ellis and Gary J. Nutt, "Multi-Dimensional Workflow"
12. Clarence A. Ellis and Gary J. Nutt, "Workflow: The Process Spectrum", Proceedings of NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, May 1996
13. David Garlan and Mary Shaw, "Introduction to Software Architecture"
14. Diimitrios Georgakopoulos, Mark Hornick, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure", Distributed and Parallel Databases, 3, pp. 115-153, 1995
15. Dimitrios Georgakopoulos and Mark F. Hornick, "A Framework for Enforceable Specification of Extended Transaction Models and Transactional Workflows", International Journal of Intelligent and Cooperative Information Systems", Vol. 3, No. 3, pp. 225-253, 1994
16. Gustavo Alonso and Hans-Joerg Schek, "Research Issues in Large Workflow Management Systems", Proceedings of NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, May 1996
17. Jeanine Weissenfels, Dirk Wodtke, Gerhard Weikum and Angelika Kotz Dittrich, "An Overview of the Mentor Architecture for Enterprise-wide Workflow Management", Proceedings of NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions,

May 1996

18. Kamath, G. Alonso, R. Gunthor and C. Mohan, "Providing High Availability in Very Large Workflow Management Systems", 5th International Conference on Extending Database Technology, March 1996

19. Kurt Wallnau, Fred Long and Anthony Earl, "Toward a Distributed, Mediated Architecture for Enterprise-wide Workflow Management", Proceedings of NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, May 1996

20. Kwang-Hoon Kim, "Practical Experience on Workflow: Hiring Process Automation by FlowMark", IBM Internship Report, IBM/ISSC Boulder Colorado, 1996

21. Kwang-Hoon Kim, "Experiences on FlowMark: Hiring Process Automation by FlowMark", Technical Notes of Computer Science Department, University of Colorado at Boulder, 1996

22. Kwang-Hoon Kim, "Practical Experiences and Requirements on Workflow", Lecture Notes Asian '96 Post-Conference Workshop: Coordination Technology for Collaborative Applications, The 2nd Asian Computer Science Conference, Singapore, 1996

23. Marek Rusinkiewicz, Amit Sheth, "Specification and Execution of Transactional Workflows"

24. Miller, A.P. Sheth, K.J. Kochut and X. Wang, "CORBA-Based Run-Time Architectures for Workflow Management Systems"

25. Mohan, Dick Dievendorff, "Recent Work on Distributed Commit Protocols, and Recoverable Messaging and Queuing", Data Engineering, Vol. 17, No. 1, Mar. 1994

26. Mohan, "Tutorial: state of the Art in Workflow Management System Research and Products", Tutorial Notes, 5th International Conference on Extending Database Technology, March 1996

27. Mohan C. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, and R. Gunthor, "Exotica: A Project on Advanced Transaction Management and Workflow Systems", ACM SIGOIS Bulletin, Vol. 16, No. 1, Aug. 1995

28. Stefan Jablonski, "MOBILE: A Modular Workflow Model and Architecture"

29. Stefan Jablonski and Christoph Bussler, "Workflow Management: Modeling Concepts, Architectures and Implementation", International Thomson Computer Press, 1996

30. Jari Veijalainen, et el., "Research Issues in Workflow Systems", ESPRIT III LTR Project Report, Oct. 1995

31. Kwang-Hoon Kim, "Architectures for Large Scale Workflow Management Systems", Thesis Proposal, University of Colorado at Boulder, Feb. 1997