A Systems Approach for Workflow Management Reality

Nikitas A. Assimakopoulos
Department of Informatics, University of Piraeus,
80, Karaoli & Dimitriou Str., GR-185 34 Piraeus, Greece.
FAX : +301-4179064, E-MAIL: assinik@unipi.gr

Abstract
It has been over ten years since the first workflow product was introduced. Despite the large number of workflow vendors and various research efforts all over the world, as well as the hype about the workflow market, workflow technology is still far pervasive. This paper approaches workflow systems and assesses the situation from the technical point of view, focusing on the development and enactment aspects of workflow processes. We discuss the current capabilities of workflow products, major issues that need to be addressed before workflow can be pervasive, as well as possible future trends and research that will help workflow succeed.
Keywords: Workflow Management, Wide Area Workflows, Transactional Workflows, Distributed Workflows, Dynamic Workflows

1 Introduction

1.1 Background

Computer technology has evolved to the extend that is being used successfully in application domains such as banking, finance, and telecommunication. Such applications of computer technology have greatly increased productivity and provided better services. Despite their success, these versions of computer applications have two major drawbacks.

First, they are monolithic in nature. All business policies and information accesses were originally hard-coded into the applications. These systems were difficult to maintain and enhance where business policies and data changed. The advance of the database technology has successfully separated data access from the applications. As database applications, the applications are more adaptive to data changes. On the other hand, business policies are still hard-coded and any change requires modifying the application code.

Second, they are isolated (i.e., stand-alone applications). The computer applications (especially those developed in old days) are usually designed and developed to work independently to solve specific problems. The advance of network and distributing technologies have made it possible for them to collaborate in primitive ways such as receiving and sending messages. There is, however, a great need to integrate those isolated information and process islands at a higher level so that they can collaboratively provide business solution that each individual application is unable to provide.

Workflow has been proposed to address the above problems of early computer applications. The basic idea behind the workflow technology is to separate business process and workflow management component from the existent applications to increase flexibility and maintainability. The major driving forces of workflow are two fold : first is the need for business re-engineering whose main purpose is to increase productivity, reduce cost, and respond to the changing environment more quickly, and second is the advent of technologies such as distributing computing, object technology, databases etc. that facilitate open and reliable information exchange and collaboration across the organization.

The separation of business policies from applications makes (workflow-based) applications easier to maintain and enhance, because changes in procedures can be made using workflow tools without having to rewrite the application, as well as providing several other advantages. For example, as business procedures are automated, production can increase dramatically. A workflow system supports policy-driven allocation of resources and can therefore adapt dynamically to change workloads. Since workflow processes can be understood by computers, it is also possible to develop workflow tools that track process executions and control process execution in more flexible ways. Another big advantage of workflow systems is that they simplify application development, not only because application components can be reused, but also because functions common to many applications such as recovery have already been provided by the underlying workflow management systems.

1.2 Workflow Systems

Workflow management is a diverse and rich technology and is now being applied to an ever increasing number of industries. Workflow is also a generic term which may refer to different things at different levels such as process modeling at the business process level, or process specification and enactment at the system level. In this paper, we discuss issues of process specification and enactement for various kinds of workflow systems (e.g., ad hoc, administrative, production, and collaborative workflow systems). The business perspective, such as issues of business re-engineering, process modeling, and BPR tools, will not be covered.

A workflow process is a coordinated (parallel and/or serial) set of process activities that are connected in order to achieve a common business goal. A process activity is defined as a logical step or description of a piece of work that contributes toward the accomplishment of a process. A process activity may be a manual process activity and/or an automated process activity. A workflow process is first specified using a process definition language and then executed by a workflow management system (WMS). A WMS is a system that completely defines, manages and executes workflow process through the execution of software whose order of execution is driven by a computer represetation of the workflow process logic.

It has been over ten years since the first workflow product was introduced. There are now at least several dozens of workflow products available on the market with certain workflow capabilities. Workflow technology has been used in a wide range of application areas such as banking, finance, insurance, health care, telecommunication, manufacturing, and document management.

Despite all these efforts and its usefulness, workflow is far from pervasive in the business and industrial world. While there are many reasons for this, some major technical ones include :

Infrastructure

Workflow systems are much more than just workflow engines that execute workflow processes. Successful execution of a workflow process requires proper supports from the underlying infrastructure. For example, technologies such as distributed computing, object orientation, and security are necessary for the workflow engine to invoke external applications (especially legacy applications). Unfortunately, distributed computing and object technologies such as Corba and ActiveX/DCOM have not been mature enough for real applications until recently.

Standards

The lack of standarts has been one of the major obstacles to wide application of workflow technology. Unlike relational databases, each workflow vendor has its own workflow model, specification language, and API. Recent efforts by the Workflow Management Coalition (WMC) have made significant progress, but there is still a long way to go.

Complexity

Workflow application development is a complex task involving more than simply specifying a process definition, itself is a formidable task. Other and more difficult tasks include wrapping external applications to be invoked by the workflow engine, managing workflow resources, and setting up communication infrastructure. Unfortunately, current workflow systems provide little help for facilitating these tasks. Every major workflow applications require lengthy and intensive collaboration between the workflow vendors and the application developers.

Technology

Despite all the technical progress, workflow technology is still far from mature. For example, none of the existing workflow products or research prototypes can provide the same level of support as relational database management systems do for reliable and consistent process executions. It is true that many workflow applications do not need this level of support. But it is also important for workflow management system to have the ability so that mission critical applications that are currently implemented using other technologies (e.g., database) can be re-engineered using workflow.

There are other papers discussing the limitation of existing workflow products and outlining important research issues (Alonso G., Agrawal D., El Abbadi A., and Mohan C., 1997). This paper focuses on identifying the technical solutions so that workflow will be more pervasive. We discuss both the current capabilities of workflow products and the major issues that need to be addressed before workflow can be successful in the market place.

2 Workflow Products

In this section, we first summarize the major features, enabling technologies, and successful applications of the current generation of workflow systems. We then describe a few industry trends that we believe are important to the next generation of workflow systems.

2.1 Current Status

Workflow systems have evolved at least three generations. The first generation of workflow systems are monolithic applications of a particular application area (e.g., image or document management). Second generation workflow systems factored out the workflow components but were still tightly coupled with the rest of the products. Third generation workflow systems have generic, open workflow engines which provide an infrastructure for robust production oriented workflow. The workflow specification is given separately through a graphical user interface and is interpreted by the workflow engines. It has been a research effort (Assimakopoylos, 1988; Assimakopoulos, 1999; Assimakopoulos, 2000) for a fourth generation of workflow systems that will be part of the middleware and will offer workflow services among other services.

2.1 .1 Workflows Product Features

Workflows products of the early age (e.g., WorkFlo by FileNet) are image-based. The purpose of such systems is to automate and manage the flow of images, data, text, and other information throughout an organization.These systems are thus also data- or document-centric. The main function of a data-centric workflow process is to route the data (e.g., design document) around so that people can work on the data.

In recent years, most workflow vendors have either developed or relabeled their products as non image-based. Most of the workflow products are also process centric (instead of data-centric) in the sense that workflow processes formalize and enforce business policies. On the other hand, there are still needs for data-centric workflow products and some vendors focus on that market segment.

In the following, we summarize major features of the current generation of workflow products that are non image-based and process-centric. Note that this is not a complete list and the listed features may also not be supported by all workflow products. Nevertheless, we believe that these features characterize the current generation of workflow products and are supported (at least partially) by most workflow products.

Graphical representation

Perhaps the most significant improvement of the current generation of workflow products over earlier generations is the ability to specify and represent workflow processes as graphical maps. In the map, major steps, data and control flows, as well as other components of a workflow process are displayed graphically using icons and lines connecting icons. The idea is to provide an intelligible process view to non-programmers such as business analysts, re-engineering consultants, end-users and supervisors.

The workflow process map is now a standard component of all workflow products. But it differs significantly from vendor to vendor with respect to the information contained in the map and the way it is represented. For example, some products support only a single one-level map while others represent process maps hierarchically, i.e., that there is one main map which contains icons representing submaps. Some products manipulate workflow data explicitly on the map while others do not. The granularity of process maps is also different. For example, many products do not include specifics of the workflow activities. As noted in process maps in most workflow products fail to describe the workflow process with sufficient clarity and completeness.

Architecture

Workflow systems, by nature, are distributed systems. Most workflow systems employ three tier client/ server architecture and run on multiple platforms. There is a workflow engine which acts as a coordinator and stores meta information in the underline database. Other components of workflow systems such as the process monitor, the process starter and the process controller are all clients of the engine. External applications that perform workflow tasks can be both geographically dispersed and on different platforms.

The workflow engines in most existing workflow systems are still centralized in the sense that the entire execution of a process is handled by a single workflow engine (or a cluster of engines that share the same data storage). It is possible in some workflow systems to start a subprocess at a different machine as a step of the containing process execution. But the subprocess execution and the containing process execution can be considered as separate process executions with little interaction except data passing at the beginning and end of the subprocess execution. No workflow systems can currently support reliable and consistent process execution collectively by more than one independent (share nothing) workflow engine.

Data model

In WMC workflow reference model, three kinds of workflow data have been identified: *process control data* that is manipulated by the workflow management system only; *process relevant data* that is used by both the application and workflow management system; and *application data* that is used by the workflow application only. The idea is to separate business policies (e.g., control flow and data used in flow control) from application details (e.g., data used to perform a task).

All workflow products have their data models but some of them are quite different from the WMC model. For example, many workflow systems do not distinguish between process relevant data and application data. In these systems, workflow engines have accesses to all workflow data (including application data).

User model

A user model specifies each user's role and the role the user coordinate. The idea is to separate the concept of the logical role which is the specification of capabilities needed to perform a task, and the concept of physical resources, that have the capabilities to perform said tasks. Process designers specify roles for workflow tasks at design time. Similarly, specific resources which have the required capabilities will be assigned to the tasks at process execution time. The advantage is that the workflow process is not tied to specific resources which may change over the process lifetime.

There are still workflow products that do not distinguish between logical roles and physical resources. However, many workflow products do support this basic user model. Some even support a more complicated model that allows for the specification of a user's organization and manager, the function and the processes that the user is authorized to use, and other features.

Rule capability

Almost all workflow products allow process executions more complicated than simple sequences. Complex flow control requires workflow products to have rule capability. Most workflow products have built-in rule engines. But the rule specification can be quite different. Some products provide script languages for rule specification while others have graphical rule editors that are easier to use.

Tools

One of the advantages of using workflow over monolithic applications is that workflow management systems include tools for process monitoring, tracking, and controlling. Most workflow products provide process development tools and some even provide animation and simulation tools.

2.1.1 Standards and Enabling Technologies

Standards and enabling technologies are important factors that must be addressed before workflow technology can be pervasive. In the past few years, significant progress has been made with respect to workflow related standards such as WfMC, MAPIWF, and OBC and enabling technologies such as email, CORBA, and ActiveX/ DCOM.

WMC standards

WMC was founded in 1993 and is now considered the primary standard body for the workflow market. The standardization work of WMC is centered around the *workflow reference model* (see Figure 1).

The reference model specifies a framework for workflow systems, identifying their characteristics, functions and interfaces. The focus has been on specifying the five APIs that surround the workflow engine. These APIs provide a standard means of communication between workflow engines and clients (including other workflow components such as process definition and monitoring tools). So far, WMC has draft specifications for all APIs except interface 3. Most workflow vendors plan to support the WMC APIs and some vendors have already demonstrated the WMC APIs (e.g. , for interface 2) working with workflow engines.

Workflow interoperability and standards are vital as automation technology becomes more complex, and the Coalition's work in this industry is central to keeping up with the rapid progress. On the other hand, workflow standardization is still in its preliminary stage and has a long way to go.

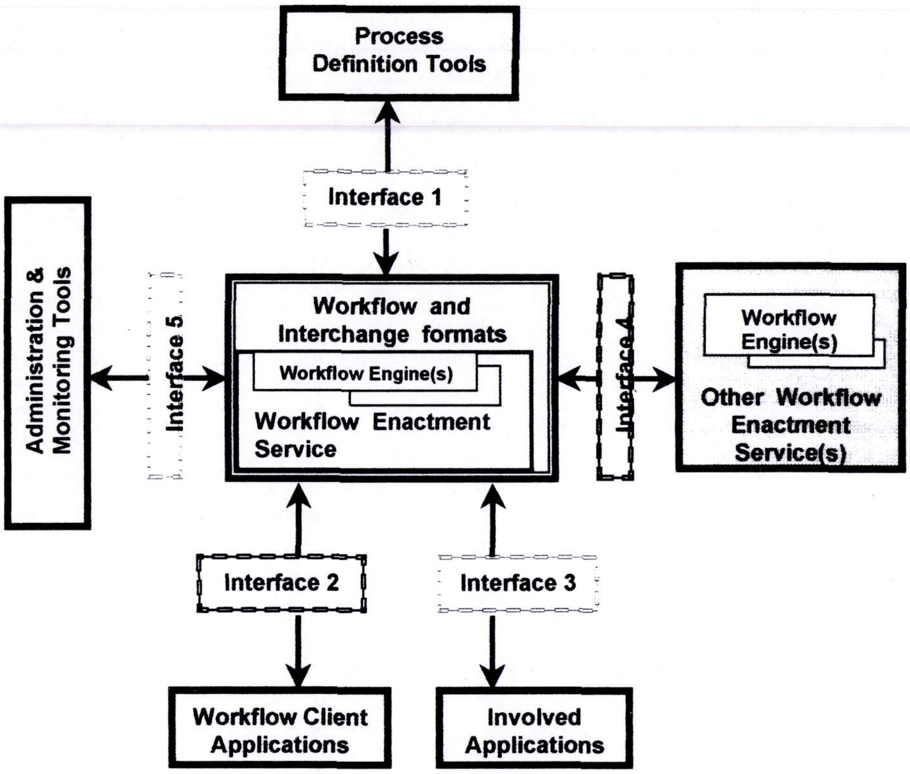
Process Definition Tools
Interface 1
Administration & Monitoring Tools
Interface 5
Workflow and Interchange formats
Workflow Engine(s)
Workflow Enactment Service
Interface 4
Workflow Engine(s)
Other Workflow Enactment Service(s)
Interface 2
Interface 3
Workflow Client Applications
Involved Applications

Fig. 1: WMC workflow reference model

MAPI workflow Framework

MAPI is a message API standard promoted by Microsoft and the MAPI workflow framework (MAPI-WF) is Microsoft's initiative to the WMC. The idea is to combine the functionalities of workflow systems and the flexibility of messaging systems so that applications can be deployed. It addresses interoperability issue between messaging systems and workflow systems. In a message environment, a workflow request (e.g., interface 4) can be packaged within some body part of a message. MAPI-WF provides a standard set of body parts and properties so that workflow package can be delivered to and from the workflow engine. Workflow components (e.g., workflow engines, workflow applications, and workflow tools) that conform to MAPI-WF can communicate via messaging systems such as Microsoft Exchange.

Given the popularity of messaging systems and the influence power of Microsoft, MAPI-WF will play an important role. Many workflow vendors have already expressed their intentions to support MAPI-WF in their workflow products. To the best of our knowledge, however, no vendors have actually demonstrated in their products.

Enabling Technologies

The two most important enabling technologies for workflow systems in recent years are object technology and distributed computing technology. Unlike other software systems such as database management systems, by nature workflow systems are distributed and open. To perform a workflow task, the workflow engine needs to invoke remote workflow applications. Object and distributed computing technologies such as CORBA and ActiveX/DCOM are very useful in wrapping, managing, and invoking heterogeneous applications.

Several workflow products have used CORBA and ActiveX/DCOM as transport services to invoke remote applications. There are also research work investigating CORBA-based workflow enactment system which supports a scalable software architecture, multi-database access, and an error detection and recovery framework.

2.2 Industry Trends

Workflow is a young rapidly changing area with existing workflow products evolving with new features and new products being introduced almost daily. It is still not clear what the next generation of workflow products will be. In the following subsections, we list some industry trends that are both important and general enough that they may be adopted by most workflow vendors. In the next section, we discuss some more advanced issues that are also important but are either not mature enough or general enough to that their adoption by workflow vendors in the near future is likely.

2.2.1 Open and Extensible Interfaces

As mentioned, workflow systems are distributed and open by nature. The organizations that will use workflow systems already have computer networks, applications (e.g., spreadsheets), data (stored in files, databases, etc.) and other information. To be useful, a workflow management system must fit into the organization's existing computing environments.

Most existing workflow products include application programming interfaces. External applications can be integrated with the workflow system, and external data can be used for workflow process execution via, e.g., application data handlers. To be a really open workflow system, extensible interfaces are needed for incorporating other existing resources and information that are needed in workflow process execution.

For example, events are one of the major means that workflow processes interact with each other and with the external environment. A telecommunication network management process must be able to react to alarms generated by the managed telecommunication network and generate events to effect changes in the network. Although, most workflow products still don't support events (especially external events that interact with external environments), some workflow systems do. Research projects that address this issue also exist.

Another example that requires extensible interfaces is the integration of existing corporate directories. Information about users and the corporate hierarchy is necessary for assigning resources to perform workflow tasks. Requiring users to register themselves to the workflow system as some products do is clearly not the best way. A more flexible method is to provide interfaces to integrate existing corporate directories into the workflow engine. This not only saves workflow application development time, but also makes later maintenance easier and avoids possible inconsistencies.

Some workflow products with extensible interfaces for integration of existing resource management systems already exists. In near future more products will incorporate the interface as this is a feature greatly appreciated by workflow application developers.

2.2.2 Process Development Environments

The development of workflow applications is generally difficult, due to their complexity. Current workflow products address this problem by providing graphical user interfaces for process design and management. GUI tools, however, only address one aspect of the problem which is relatively easy to deal with: specifying process templates. A harder problem is to integrate the workflow process with the computing environment. This is difficult because of the heterogeneity and complexity of the computing environments. To make things worse, most workflow vendors designed and marketed their products as generic tools trying to cover all application areas.

We believe that workflow will not become pervasive until the complexity of developing workflow applications can be significantly reduced. One way of doing that is to provide a good development environment. A good environment must be domain-specific to provide commonly used process templates, commonly used data forms, tools to wrap and manage commonly used applications, basic communications infrastructure, etc. Currently, there are special-purpose workflow products available on the market for specific domains. For example, Araxsys has products specifically targeting a healthcare market (Araxsys, 1997), and Ariba's products focus on operating resource management (Ariba, 1997).

For the general-purpose workflow products, it is possible to develop special packages based on the general-purpose workflow engine. For example, FileNet has introduced VisualFlo/Payable for account payables. HP has introduced AdminFlow for business administration. Future such packages will be expected to cover application domains such as telecommunications, banking, and finance.

As mentioned, an important aspect of a workflow process development environment is to wrap external applications to be used by workflow process. The wrapped applications can be packaged into a library and then reused by workflow processes. Workflow process development can be further simplified if workflow activities can be reused. Workflow activities include more than just external applications to be invoked. Other information include: logical role specification that maps to the specific external application; data needed to perform the task; communication mechanisms; consistency and deadline specification, etc. Unfortunately, most workflow products do not support this level of reuse, as workflow activities contain process-specific information (e.g. position in the process) that cannot be reused. It is thus necessary to separate process-specific and process-independent parts of workflow activities. For example, HP's workflow product distinguishes between the two parts and allows the reuse of the process independent parts of workflow activities. This allows a special-purpose workflow environment such as AdminFlow to be easily developed based on the general purpose workflow engine.

2.2.3 Wide Area Workflow

The current generation of workflow products has been criticized for rigid process models, narrow application focus, and platform restrictions. These workflow products best service applications where business rules, process flows, and work participants are known in advance and rarely change. The advent of wide area networks and the World Wide Web has provided new opportunities for workflow. Most workflow vendors have provided web interfaces to their workflow products. There are also research projects trying to develop workflows on the web.

It has been predicted that one possible change for workflow technology is users' environments where workflow tasks are performed. Workflow users will have universal access to them via open interfaces such as email, telephone, fax, pager, Web, and intranets/ extranets to perform workflow tasks. The key is to separate workflow processes from the user environment so that changes on one side will not affect the other. The major difference between traditional and wide area workflows is that workflow users have control over what kind of information they receive and how they receive it. The advantage is faster response time and greater productivity by providing multiple access points to the same information and allowing users to use tools of their choice.

3 Workflow Research

Workflow is an active research area with research efforts occurring both in the academia and industry. However, workflow research, especially that from academia, has made little impact on workflow products. There are two reasons for this situation. First, early workflow systems, having evolved from different areas such as office automation systems, job control systems, and document management systems, have struglled to define basic models, architectures, and functionalities, while workflow researchers, most with strong database backgrounds, have focused on introducing advanced database techniques to workflow systems. In addition to this, workflow vendors have not been very successful in applying workflow technology to applications that require advanced database features such as ACID transactions. Workflow researchers have also failed to develop techniques that are flexible enough for workflow systems. Nevertheless, we believe that on-going research can address issues that are very important in making workflow pervasive. Our experience with customers shows that there are many workflow applications that require some level of transaction supports. The requirements, however, are very different from those in database systems. As a result, not only do existing database techniques need to be adapted to fit into the workflow environment, but new techniques also need to be developed to address issues unique to workflow systems.

In the following section, we discuss some of the important research issues. We will emphasize the differences between the database and workflow environments. Note that this is not a complete list of workflow research issues. The purpose here is to inspire research in these and other related areas.

3.1 Transactional Workflow

The concept of transactions was first introduced for database applications. A transaction is an execution unit with ACID properties : it maps a database from one consistent state to another (*consistency*); either all or none of its effects take place (*atomicity*); and the effects are made permanent once committed (*durability*). Multiple transactions may be executed concurrently, but the overall effect must be equivalent to some sequential execution (*isolation* or *serializability*).

Workflow models that support certain transactional properties have been viewed by many researchers as extensions to the relaxed transaction models (Eder J. and Liebhart W., 1995; Kamath M. and Ramamritham K., 1996). It has been proven both possible and very useful to incorporate transactional semantics such as recovery, atomicity and isolation to ensure correct and reliable workflow executions (Juopperi J., Lehtola A., Pihlajamaa O., Sladek A., Veijalainen J., 1996). Database techniques have been adopted to provide transactional properties for workflow processes. For example, failure atomicity is ensured via both forward recovery (Eder J. and Liebhart W., 1996) and backward recovery. Execution atomicity can also be ensured by specifying consistency units (or execution atomic units) of workflow processes and coordinating their executions to ensure M-serializability (Rusinkiewicz M. and Seth A., 1994).

On the other hand, a workflow process is fundamentally different from a database transaction as discussed in Worah D., and Sheth A. (1997). First, a workflow environment is more complicated than a database and involves heterogeneous and distributed components, as well as human interactions. Second, a workflow process is structurally more complex than a database transaction, and the execution of a process may establish quite complex control and data flow dependencies among the activities of the process. A workflow process specification may include conditional branching, concurrent execution of activities, loops and other complex control structures.

Database recovery techniques such as logging have been successfully adopted in workflow systems. There are workflow products that support reliable workflow process executions. Ensuring atomic and consistent process execution, however, is still missing from workflow products and remains an open research issue. In this subsection, we discuss new issues in workflow compensation and concurrency control as the result of the above differences between the database and workflow environments.

3.1.1 Compensation

Compensation has been used to simulate the transactional properties for long-running database applications that would be too expensive to be implemented as single transactions. The idea was to implement such an application as a *saga* or sequence of ACID transactions so that resources needed only at a particular stage could be released after the corresponding transaction completes. Atomicity was simulated by corresponding already transactions in reverse order.

In workflow systems, compensation is used to deal with process activity failures. When a process activity instance fails, the workflow management system is responsible for bringing the process execution to a designed *save point*, which is a previous execution step of the process. The save point represents an acceptable intermediate state of process execution and also a decision point where certain actions can be taken to either fix the problem that caused the failure or choose an alternative execution path to avoid the problem. To roll back workflow process execution, compensation activities will be invoked to undo the effects of the completed activities. Compensation is more complicated (and thus interesting) in workflow systems than in database systems for two reasons. First, compensation specification (i.e., when, what, and how to compensate) is more difficult, due to the complexity of workflow processes and activities. The compensation activity can be as complicated as (or even more complicated than) the original activity that needs to be compensated. Second, optimization of compensation processes (i.e., what activities don't need compensation) is important. Unlike database transactions which can be compensated and re-executed easily and efficiently, workflow compensation can be very costly. Therefore it is very important to avoid unnecessary compensation as much as possible.

Existing research on the issue has been focused on the static specification of compensation scopes. For example, it has been discussed enhancement to IBM FlowMark which allows the process designers to specify spheres of compensation to determine the scope and extent of compensation in case of activity failures. The failure of an activity may cause the compensation of just the failed activity, the entire containing sphere, or the containing sphere and other dependent spheres. A similar approach has also been proposed in Assimakopoulos (1992) for hierarchical workflow processes. The compensation scope is determined in a bottom-up fashion: first to the designated save point in the transaction containing the failed activity, then to the designated save point of the higher level containing the transaction if the current level transaction can not hundle the failure.

An interesting issue is how to make use of run time information in order to further avoid unnecessary compensation. As we mentioned, the purpose of compensation is to undo that which caused the failure so that the execution can resume. Thus, a workflow activity needs compensation if it contributs to the failure, and /or its re-execution is different from the original execution. Compensation scopes specify the activities that *might* affect the failed activities in some execution environments. However it is possible that an activity in a statically specified compensation scope did not contribute to a particular failure. For example, an activity a_1 affects a subsequent activity a_2 if another concurrent activity a_3 occurred before a_1. There is no need to compensate a_1 if a_2 failed before a_3 has started. This information, however, will only be available at run time. Identifying unnecessary compensation and avoiding it at run time is difficult because other nodes may be affected. But in many cases it is worth the effort because compensation and re-execution of workflow activities can be very expensive.

In general, we assume some kind of static relationship between the original execution and its compensation. For example, the same compensation strategy will be used for a workflow activity independent of the cause of failures. A compensation activity is defined for each activity or a group of activities that need compensation. This, however may not be true in real life. There can be many different ways to recover a failed execution according to the cause of the failure, and the compensation process can be structurally independent of the original execution. Little research has been done in the area. The most fundamental issue of compensation is, perhaps, the correct criteria for workflow process execution and compensation. In database systems, a compensation is correct if everything between the save point and the failure point is compensated and in the exact reverse order of the original execution. In workflow systems, we need a more relaxed criterion for optimization purposes. For example, the order requirement could be relaxed for compensation that are commutable. A good understanding of correct compensation is essential to efficient workflow compensation and may even be application-depended.

3.1.2 Concurrency Control

Concurrency control is classical technique in databases which ensures execution isolation of a transaction from other conflicting transactions. Although, concurrency control has been considered either unnecessary or too costly for many workflow applications, it can be very important for some workflow applications where mission-critical operation requires a consistent view of the execution environment (Juopperi J., Lehtola A., Pihlajamaa O., Sladek A., Veijalainen J., 1996).

The problem of concurrency control in workflow systems is, however, a little bit different from that in database systems. The purpose of concurrency control in database systems is to ensure execution isolation of database transactions which consist entirely of atomic read/write operations that are visible to the DBMS. In workflow systems, the WMS ensures the execution isolation of workflow activities which consist of atomic read/write operations as well as external executions that are invisible to the WMS. The WMS is responsible for the consistency of the overall execution environment which includes both the internal database that is visible to the WMS, and the external systems which are invisible to WMS, as well as their cross consistency.

The fundamental issue of concurrency control in workflow systems is correctness criteria. Serializability, as used for database transactions, is too strict, for most workflow applications. The main reason for this is that workflow activities are generally long duration. It is unacceptable in many workflow applications to schedule conflicting activities sequentially as for read/write operations in database transactions. Relaxed correctness criteria (which might be application domain specific) are essential in specifying and enforcing the correct workflow process executions. Kamath M. and Ramamritham K., (1996) discussed some of the existing research on the subject. Some existing research addresses the problem by specifying and enforcing data and execution dependencies among workflow activities. There is also research that adopts database techniques, but allows flexible specification of consistency requirements with respect to scope and granularity. For example, it allows for groupping a collection of workflow activities of a workflow process into a consistency unit and uses traditional concurrency control to ensure isolation of consistency units (in terms of serializability). Correct execution of activities inside a consistency unit is ensured by enforcing the proper data and execution dependencies.

3.2 Distributed Workflow Execution

Workflow systems are, by nature, distributed systems. First, external applications that perform workflow tasks are often geographically dispersed. The workflow management system itself can also be distributed. The most common form of distributed WMS is function distribution. In such a system, different workflow components that perform various workflow functions such as process definition, process execution, process monitoring, and resource assignment run at different sites. WMS components interact with each other via messages or remote procedure calls. Another form of distribution is to perform a workflow function by multiple functionally equivalent WMS components that share common storage. For example, the execution of a workflow process can be collectively performed by several workflow engines sharing the same data storage for process definitions and execution states. Such a system provides better scalability and is resilience to workflow engine failure, but is still vulnerable to the data storage failure.

The most difficult form of distribution is to have multiple independent WMSs (sharing no common data storage) collectively execute a workflow process. In such a distributed system, each WMS is itself a complete workflow system with its own engine and data storage. There is no centralized server keeping all the information about a process execution. Such a system may be preferred for performance or reliability reasons. The system is more efficient because workflow activities can be executed by the WMSs that are close to the corresponding external applications (thus reducing communication cost between the WMSs and applications) and because the WMSs access process definitions and execution states locally (thus reducing communication cost between the WMSs and the data storage). It is also more reliable because the failure of one or more WMSs (including the corresponding data storage) does not stop workflow process executions. The overall system is functional as long as one of the WMSs is still running. There are two issues that are key when implementing such a distributed workflow system: data replication and execution coordination. Data replication is necessary to ensure reliable process execution. For example, the process execution can survive a single WMS failure if the process definitions and execution states are replicated at more than one independent (e.g., primary and backup) WMSs. Data replication (especially that of process execution states), however, can be very costly. Data replication can be provided by the WMSs, or the underlying systems. The advantage of the workflow system level replication is flexibility. For example, workflow processes can be executed at different levels of reliability from no replication (efficient but vulnerable to single WMS failure) to full replication (expensive but resilience to single WMS failure).

Execution coordination is needed when more than one WMS collectively execute a workflow process. For example, execution of a workflow activity by one WMS may cause the entire process execution to be suspended, affecting all other WMSs. The key is to transfer process information to a site when it is needed and in the right order. Static information such as process definitions can be replicated to all relevant site, but run time information such as process instance states has to be transferred at run time from sites to sites. This can be done in two ways : by circulating all information pertaining to a process and its execution across different sites, or pre-compiling the process definition to determine at which sites the different activities are to be executed. The advantage of the former approach is flexibility in the sense that the WMS can choose to execute a workflow activity at any site according to the run time execution environment. The disadvantage is possible high communication cost, as the information package can be very large. The later approach, on the contrary, can be efficiently implemented, because only relevant information is transferred to the site, but is inflexible. For example, if a workflow activity is pre-assigned to a site which is not accessible at the time, other sites cannot take over the execution as they do not have the information. Another problem with this approach is that most workflow products assigned resources to a workflow activity at run time. The site that is pre-assigned to execute a workflow activity at process specification time can be far away from the resources (e.g., computer applications) to be invoked.

Concurrency control and compensation may also be complicated when the WMS is distributed. For example, executions of conflicting activities at different sites have to be coordinated to ensure the consistency of the overall execution. Locking is generally not acceptable as workflow activities are often long running. Serializability, on the other hand, may not be needed for the execution. New correctness criteria and coordination algorithms need to be developed to ensure correct and efficient process execution.

3.3 Dynamic Workflow

One of the common assumptions made by most workflow research makes is the availability of pre-specified workflow definitions. Although there is research or even workflow products which allows for the modification of process definition at run time

(Casti F., Ceri S., Pernice B., and Pozzi G., 1996) it is still considered to be rare and costly.

Dynamic workflow systems are special workflow systems that have no pre-specified process specifications. They start with some initial activities. When an activity has completed, new activities will be selected according to the execution status and results of the current activity. In other words, the workflow is specified and executed at the same time, which is different from dynamic modifications of pre-specified workflow definitions.

Dynamic workflow systems are suitable for workflow applications where process specifications are frequently modified or cannot be pre-specified. For example, most product designs do not follow a strict process. They start with initial tasks (e.g., collecting requirements) and follow the general guidelines. Different tasks are performed in different orders according to the status of the design.

Dynamic workflow requires revisiting most of the research issues discussed before (as well as issues not mentioned in this paper). For example, specification of consistency requirements will be different, due to the lack of the whole picture of the processes. For the same reason, coordination (especially in distributed environments) of activity executions would be difficult. There has been very little research regarding dynamic workflow. Recently, some research efforts have tried to implement dynamic workflow systems using mobile agents. But such efforts are still in their early stages and have not addressed the aforementioned issues.

4 Conclusions

This paper tries to understand the situation in the existing computer systems where new applications are anticipated to join the current systems due to technology developments, and we presented a systems approach especially for the 4th generation workflow systems from a technical point of view focusing on the specification and enactment of workflow processes as an alternative in order to keep the existing data bases. While, there is a high demand for workflow systems for all kinds of computer applications, workflow in general is far from pervasive. It has been suggested that workflow systems is a solution to address many monolithic computer applications and that has attracted interest from both industry and academia.

Based on our knowledge and experience, we believe that the following factors have all contributed to the current situation: (1) unavailability of proper infrastructure; (2) lack of standards; (3) complexity of workflow process development; and (4) immaturity of workflow technologies. Despite all the problems great progress has been made in the last few years with respect to infrastructure, standards, and technologies. In the paper, we discuss both state of the products and state of the art of workflow management systems. The purpose of this paper is to inspire further research and development in some workflow areas that are important or essential to the pervasive of workflow systems.

References

Alonso G., Agrawal D., El Abbadi A., and Mohan C. (1997). Functionalities and limitations of current management systems. IEEE Expert (Special Issue on Cooperative Information Systems) 12(5).

Araxsys. (1997). The araxsys solution. http://www.araxsys.com .

Ariba (1997). Introducing ariba ORMS^{TM}. http://www.ariba.com .

Assimakopoylos, N. (1988). The routing and cost of the information flow in a System, Systems Practice, vol. 1(3), 297-303.

Assimakopoulos. N. (1992). A systems approach for hierarchically organized systems. Analyse de System, vol. 18(3-4), 3-13.

Assimakopoulos. N. (1999). An ever-changing systemic environment for Migrating Workflows. Proc. of the 3rd Iinter. Conf. of Computer Anticipatory Systems, Liege, Belgium, August 9-14, 1999, 212-220.

Assimakopoulos. N. (2000). The use of STIMEVIS in Business Process Reengineering with Workflow Specifications. Journal of Computer Anticipatory Systems, vol. 5, 269-291.

Casti F., Ceri S., Pernice B., and Pozzi G. (1996). Workflow Evolution. ER Press.

Eder J. and Liebhart W. (1995). The workflow activity model WAMO. Proc. 3rd Intl. Conference on Cooperative Information Systems, CoopIS, Vienna. R. Oldenburg Verlag, 249-265.

Eder J. and Liebhart W. (1996). Workflow recovery. Workflow Handbook, W. Liebhart (ed.) Wiley.

Juopperi J., Lehtola A., Pihlajamaa O., Sladek A., Veijalainen J. (1996). Usability of Some Workflow Products in an Inter-organizational Setting. IFIP WG8.1 Working Conference Proc. on Information Sysrems for Decentralized Organization, 134-148.

Kamath M. and Ramamritham K. (1996). Correctness issues in workflow management. Distributed System Engineering, vol. 3(4), 303-327.

Rusinkiewicz M. and Seth A. (1994). Specifiacetion and execution of transactional workflows. In Modern Database Systems : The Object Model, Interoperability, and Beyond, W. Kim (ed.), Addison-Wesley.

Worah D., and Sheth A. (1997). Transactions in trnsactional workflows. In Advanced Transaction Models and Architectures, S. Jajodia and L.Kerschberg (eds.) Kluwer Academic Publisher.