Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 1997 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

December 1997

A Bottom-up Approach to Distributed Workflow

Hung Wing
The University of Queensland

Chengfei Liu
The University of Queensland

Robert Colomb
The University of Queensland

A Bottom-up Approach to Distributed Workflow

Hung Wing, Chengfei Liu and Robert M. Colomb

CRC for Distributed Systems Technology*
Department of Computer Science
The University of Queensland
Brisbane, Qld 4072, Australia
wing@cs.uq.edu.au

Abstract

In this paper, we describe a bottom-up approach to distributed workflow, which facilitates electronic commerce and education on the Internet. In general, a workflow management system is used to define, execute, and manage the data and control flow of a business process. Often, the knowledge and definitions of a business process which is being executed is centralised in one node, and this overall knowledge must be available before commencing a business transaction. There are several problems inherent in this centralised, top-down approach. First, communication bottleneck and system failure associated with the centralised node will bring all relating workflow applications to a standstill. This is because the knowledge of the workflow is the core of most workflow systems. Heavy network traffic arise when many client applications try to communicate with the centralised node for their parts of the business process definitions. Second, the high availability, flexibility and scalability of system configurations are limited in the centralised node. Third, there are trade applications which do not develop the overall integrated workflow up front, but rather through a series of incremental trading steps and gradual negotiations. These limitations emphasise the need for a bottom-up approach to distributed workflow; one which can support this sort of incremental trading. In short, this paper presents an approach to a distributed workflow which aims at the following: 1) promotion of incremental trade development; 2) avoidance of a communication bottleneck, and; 3) isolation of node failures from affecting the entire workflow.

1 Introduction

Worldwide co-operation coupled with increasing competition in every aspect of education has forced institutions and universities to be more flexible and efficient than ever before. Consequently, education enterprises need effective ways to mass-market their products and extend their operations to the open global markets, while still trying their best to minimise operating cost. This must be achieved without losing the quality expected in conventional education.

The demand then, is for a just-in-time, on-demand, team-based, networked, geographically dispersed, and automatic approach in providing 'on-line' education to promote open learning in the immediate future. Hoping to fulfill this demand, the following research provides a generic framework to support the distribution, sharing and management of trade documents in dealing with education brokerage on the Internet. In particular, a bottom-up approach to distributed workflow is examined and assessed.

An example of a bottom-up approach to workflow is an "arms-length" trade interoperability relating to education brokerage on the Internet. In this environment, a course customer, provider, and education broker (eg. virtual university) may not know each other's 'ways of doing business' (eg. trade procedure) prior to commencing a trade. In a sense, they see each other's data and control processes as 'black-boxes'. However, an integrated trade procedure can be formulated after a series of trading steps and negotiations. In a sense, their business process can only be formed on the fly. This is in contrast with the top-down approach in which an overall workflow must be known in advance and available for partitioning, monitoring and execution purposes.

*The work reported in this paper has been funded in part by the Cooperative Research Centre Program through the Department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

In general, most Workflow Management Systems (WFMSs) define and use the workflow in a top-down manner. This trend is probably due to the fact that workflow systems are known to be originated from the office automation and batch processing in the late 1970's. System such as form management (Tsichritzis, 1982), and electronic mail systems (Goldberg et al., 1992) in which the routines associated with the work process are fixed, well understood, and readily available for usage. It is however, harder to automate a business process that is not so easily captured and understood such as arms-length trade interoperability. Nonetheless, the huge potential market of electronic commerce on the Internet indicates that there are many benefits to be gained by overcoming the incremental trade development's barriers. There is, in other words, an urgent need to look at the workflow problem from the bottom-up point of view in order to support what we call incremental trading.

The definitions and semantics of the workflow being monitored or executed by most WFMSs are centralised in one node. The workflow system is thus, very vulnerable to communication bottleneck and not so resilient to node failure.

In short, our distributed workflow model has the following objectives:

- The promotion of arms-length trade interoperability in which each node can co-operate step-by-step with others, in order to know more about each other's ways of business. This will enhance their trading capabilities.
- The avoidance of communication bottleneck and a more resilient response to node failure
- Study of the feasibility of using the bottom-up approach in processing distributed workflow.

This paper is organised as follows: section 2 discusses other works related to this research. Section 3 describes the background concepts and problems associated with the top-down approach. Section 4 describes the bottom-up approach to workflow. Section 5 discusses some implementation issues. Section 6 gives an example of usage. Section 7 discusses the pros and cons associated with the different approaches. Section 8 provides concluding remarks.

2 Related Work

Different WFMSs are designed for specific purposes and to serve a particular type of workflow product. There are 4 particular types of workflows: 1) popular production workflow systems such as InConcert, FileNet, FloWare, FlowMark, ViewStar which are designed to automate business-critical applications. Most often these borrow some form of transaction processing capabilities; 2) Ad-hoc workflow systems such as Action and Keyfile which are designed to route and track routine office work based on unstructured information. A special type of ad-hoc workflow such as Action Workflow (Medina-Mora et al., 1992) is based on the customer-performer paradigm associated with speech act theory (Searle, 1969). It is designed to facilitate customer satisfactions through a series of workflow loops containing the following phases: preparation, negotiation, performance and acceptance; 3) Administrative workflow systems such as Jetform and Staffware which are designed to automate administrative work driven by paper forms, and; 4) Collaborative workflow such as Lotus Notes which is based on the document-oriented model, supporting the automation of business-critical processes that are not transaction oriented. A survey of the infrastructures of the current workflow systems can be found in (Georgakopoulos et al., 1995).

In general, conventional workflow models do not address data sharing, persistence and failure recovery. These features have been proposed in several advanced transaction models such as: Sagas (Garcia-Molina and Salem, 1987), ConTract model (Waechter and Reuter, 1992), and Flex Transaction model (Elmagarmid et al., 1990), and many others. These models aim at relaxing the ACID property of the transactional model (Elmagarmid, 1992) thus making it more suitable to the needs of the applications.

However, there is a special class of workflow models which combines both the transactional property and the complex organisational capability of workflow models into one integrated model (Breitbart et al., 1993; Georgakopoulos et al., 1995; Georgakopoulos and Hornick, 1994; Sheth and Rusinkiewicz, 1993; Sheth and Rusinkiewicz, 1995; Hsu, 1993). As an example, a transactional workflow (TWF) system such as InConcert (McCarthy and Sarin, 1993) and in (Kuo et al., 1996) exhibit some of the transactional features such as concurrency and recovery.

The above WFMSs and TWFs are designed to complement each other's functionalities. For example, Lotus Notes provides an excellent environment for combining forms, document management, messaging and replication without transactional capability, as well as a built-in higher level graphical process definition capability. Products like Action workflow, InConcert are capable of providing the missing graphical process definition and transactional capability.

In contrast, distributed workflow is not supported in InConcert. However, with the combination of FlowMark (IBM, 1995; Leymann and Roller, 1994), Message Queue Interface (IBM, 1993) and Notes (Lotus Notes, 1995) a top-down approach to a distributed workflow model can be achieved. One such project is Exotica/FMQM (Alonso

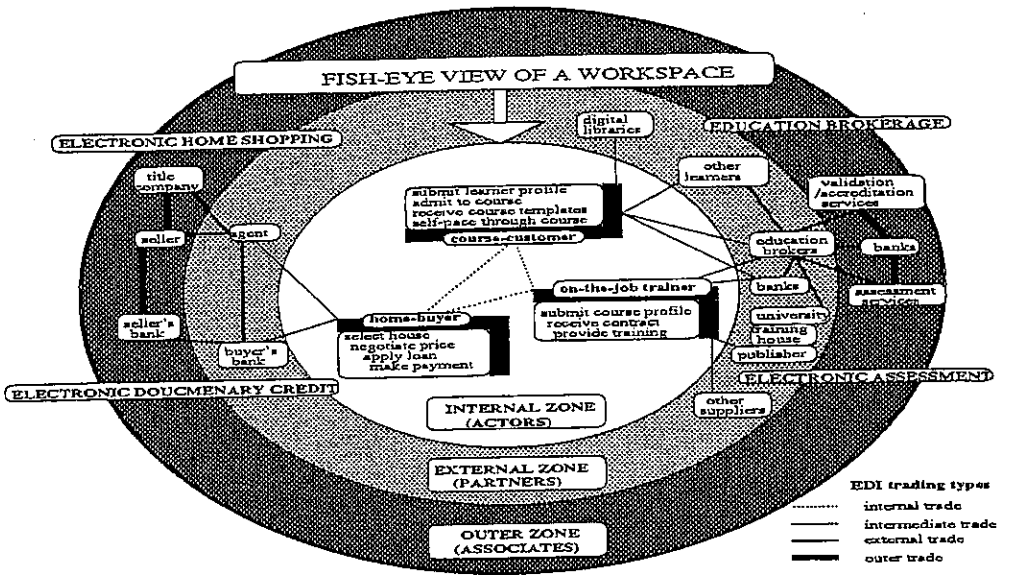


Figure 1. Electronic Documentary Dossiers of a workspace

et al., 1995). Another similar work, a CORBA-based distributed workflow has recently been introduced in (Miller et al., 1996). One common attribute in these proposals is how to use the persistent message queues to facilitate recovery in the presence of node failures. However, in their current states, neither of these models can be used to support those applications which do not have the overall workflow up-front but rather through a series of incremental trading steps.

The main principle of the Exotica/FMQM is that an integrated workflow that is centralised in one single node can be partitioned into smaller sub-nets which can then be stored and executed from a collection of other nodes. In this way, communication bottleneck can be avoided and made more resilient to node failure than in the case of the centralised workflow model. Our work is similar to the Exotica/FMQM model in the sense that the business data is defined and managed by Lotus Notes repositories and uses transactional PUT and GET (i.e. retrieve and update to message queues is only permanent after the transaction commits) of the message queues.

The CORBA-based TWF supports both the so called 'semi-distributed' and 'fully distributed' run-time architecture, while the FMQM system provides only the latter architecture. A fully distributed architecture refers to the distribution of both the workflow scheduler and task manager. A semi-distributed model refers to the distribution of only the task manager while keeping the tasks' scheduling algorithms in a centralised node. A workflow scheduler is used to coordinate the execution of tasks by enforcing the inter-task dependencies. A task manager is designed to start tasks and to supervise the forward recovery process. A comparison of these architectures is given in (Miller et al., 1996).

Another different between FMQM and CORBA-based TWF is that the FMQM model uses the transactional PUT and GET calls of the message queues to facilitate the recovery and asynchronous communication. On the other hand, the CORBA-based model uses the transaction and concurrency service, and the extended IDL of CORBA to support the transactional capability and asynchronous communication.

The major difference between our work and the above proposals would be in how the workflow is built and distributed. In the FMQM and CORBA-based distributed model, workflow is assumed to be available up front and distributed to other nodes (top-down approach). In our proposal, partial workflow is available in one of the documents. Step-by-step, an integrated workflow will be formulated after a series of successful trading steps. Our model exhibits the so called "arms-length" trade interoperability. Where trade procedures belonging to other nodes initially remain as black-boxes to all nodes. However, these black-boxes gradually become transparent to a selective few nodes which have gained special privilege and knowledge after a series of continuous tradings and negotiations.

The document-based workflow model is not new, in that a structured workflow based on a combination of Lotus Notes/release 4 (Lotus Notes, 1995) and the IBM/FlowMark (Leymann and Roller, 1994; IBM, 1995) has been recently introduced in (Reinwald and Mohan, 1996). However, this combination alone is not sufficient enough to support distributed workflow features.

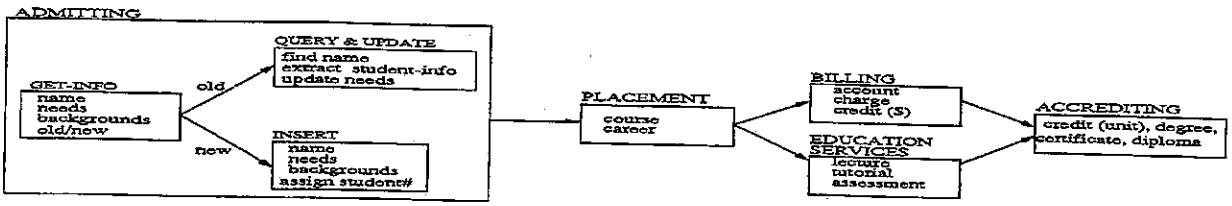


Figure 2. An electronic education (open/EDI) business process example

3 Background

This section provides a motivating example and background of distributed workflow application associated with open trading.

3.1 An open, one-to-many trading environment

To facilitate discussion on the organisation and representation of trade information in electronic commerce, Figure 1 shows a fish-eye view of person X's workspace. In this figure, person X may act as a course customer, a buyer, and a course provider depending on whether he/she is involved in the following trading tasks: taking a course from an education brokerage (Hämäläinen et al., 1996), buying a home, or giving instruction to a series of courses, respectively. Shown in the centre of Figure 1 are the 3 trade document folders associated with the above tasks.

In the conventional way (eg. paper-based), these tasks require documents (eg. learner profile, Letters-of-credit, course templates etc.) to be manually exchanged, processed, and stored in an appropriate manner. For example, to apply for a home loan or admission to a university, X probably needs to see the appropriate authority such as bank manager or university administrator, get an application form, fill out the necessary information, submit it, wait for the outcome, etc. This information can then be utilised for the purpose of recording and/or establishing other trades (eg. home buying information can later be used for filing income tax etc). The various related documents and personal reminders (eg. attached yellow pads with notes) can be organised and placed in the appropriate information drawers and folders.

This information processing and recording system is indeed simple, until it is extended to electronic processing, where all of the trade procedures and information pertaining to the trades need be computerised. Our aim is to formally represent the above trade procedures so that a WFMS can monitor, execute and manage the workflow without too much complication which can be caused by communication bottleneck, centralised node failures and the incremental trade development requirements.

The work in (Wing and Colomb, 1997) specifies an architecture which supports the underlying principles and design of the electronic documentary dossiers, a compound framework containing computer process-able information such as trade procedures, different types of trades, and the EDI message protocols etc. This information structure supports the interoperability of information parcels in an open trade environment. In a sense, the dossier construct is essentially a documentary folder that is capable of storing and managing the various types of trades and associated commodities (eg. purchase order, letter of credit, invoice, etc.) which can be both formally represented and inter-operated across heterogeneous platforms. This paper is concerned with the characteristics of the dossier with regards to distributed workflow. For example, Figure 1 shows the X dossier (eg. information drawer associated with X's internal zone) which consists of other dossiers (eg. information folders) such as Home-buyer, course-customer, and course-supplier. Also shown in Figure 1 are the other traders' dossiers (eg. seller, digital libraries, education brokers, agent, and buyer's bank, etc). Our concern in this paper is to show how the structured trade information residing in each personal workspace can contribute to what we consider to be an approach to distributed workflow management.

To facilitate discussions on the distributed workflow, the following section provides an example of a business process.

3.2 Example of a business process.

Shown in Figure 2 is an example of a business process associated with an education brokerage, an open electronic commerce application to be operated on the Internet. The figure depicts a process of admitting a student into an institution, placing that student into a course, providing education services to that student, billing him/her for the services, and finally providing the student with an accredited credential. The admission process, a nested business

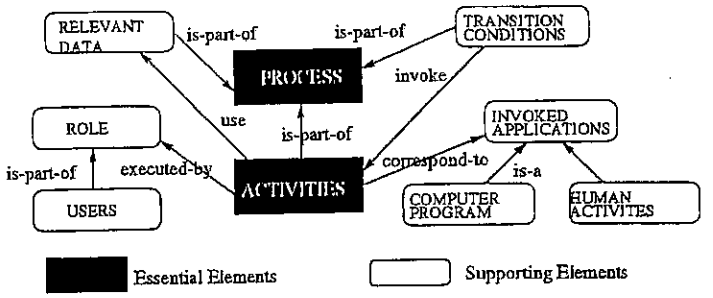


Figure 3. Elements of process-oriented model

process, comprises many other sub-processes: Get-Info, Query/Update, and Insert student information. Next, we describe how this business process can be modelled with the process-oriented workflow elements.

3.3 Workflow concepts in a process-oriented model

Today, many workflow products exist. In order to minimise the confusion of describing the various workflow concepts and products, we use, in this paper, the Workflow Management Coalition 's workflow reference model (WfMC, 1995) to describe the workflow notions.

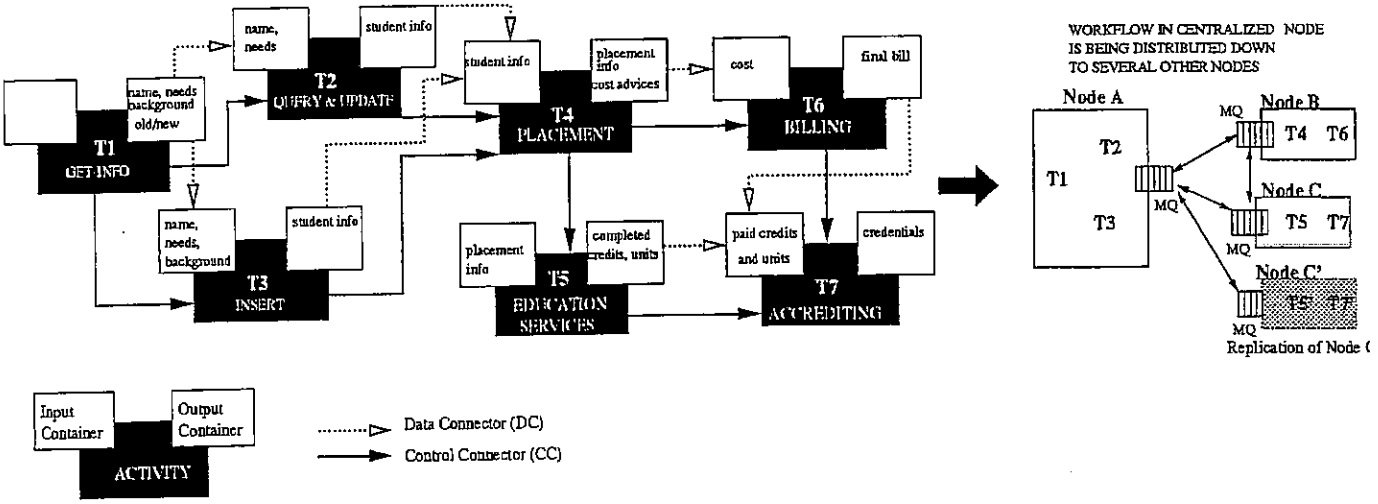


Figure 4. Formalising the example by using the process-oriented FDL (eg. FlowMark)

As depicted in Figure 3, underlying supporting elements of most WFMSs are centred around the process and its activities. The work steps within an organisation define the so called "business process". According to the workflow reference model, a business process is "a procedure where documents, information or tasks are passed between participants according to defined sets of rules to achieve, or contribute to, an overall business goal". The representation of this business process is hence referred to as "workflow".

In the process-oriented model, a workflow consists of two key elements: process and activity. Other notions such as relevant data, transition conditions, user, and role etc. are associated with the process and activity. These relationships are depicted in Figure 3.

Since the reference model does not specify a particular implementation language, we choose the IBM FlowMark's Flow Description Language (FDL) to describe the implementation of the above example. This decision is based on the fact that most concepts in the reference model can be implemented and supported by the IBM/FlowMark FDL.

The flow of the above business process is shown in the left portion of Figure 4. Often, the definitions and semantics of the workflow being monitored or executed are centralised in one single node. This decision has caused some serious problems which we will address and provide solutions for.

To begin, we need to explain what a top-down approach to distributed workflow is.

3.4 Top-down Distributed workflow steps

One approach to distributed workflow centres around the idea that the monitoring and execution of a workflow and its workloads can be divided into several nodes instead of one centralised node. In the context of this paper, a top-down workflow model refers to the characteristic that an overall workflow has been designed and defined in advance. Depending on the participants' roles and usages, the various portions of that workflow can be distributed and executed in several nodes. In general, this type of workflow modelling and support suits applications in which the business process can be easily captured and understood by all participants. An example of such an application is the common travel request example. In short, the top-down approach workflow centres around the availability of the business process and its activities. Due to this characteristic, we refer to the workflow used in the top-down approach as the process-oriented workflow. Next, we describe the distribution steps associated with the process-oriented style of workflow in term of process server and its instances.

The process server contains the process definitions and other relevant workflow information. The process instance can be executed on other nodes at run time. The following is a sample version of a distributed workflow model based on the top-down approach.

1. Users create a process
2. A process is then compiled in the server node
3. A process in the server node is decomposed into several process tables and distributed to the client nodes.
4. Upon receipt of its part of the process, a node then creates the process table (stores the static information about its part of the process), instance table (stores the run-time information and to keep track the status of the activities that are being executed), and the necessary process and activity threads to handle the execution of the instances.
5. Finally, in order to support recovery in the presence of crashes and asynchronous communication between applications that run at different points in time, trade messages between the nodes are to be handled by the transactional PUT and GET calls of the message queues (MQs). Message Queue is one of the mechanisms which can be used to place and retrieve trade messages without depending on the underlying supporting communication protocols. These message calls are executed within transactions so that their effects are not permanent until the transaction commits. Its usage has been demonstrated in (Alonso et al., 1995).

An example of the top-down distribution is shown in the right portion of Figure 4. In this figure, the activities (T1,...,T7) have been distributed and isolated into three nodes: A, B, and C. Also shown is that the network communication between these nodes is being handled through the message queues, depicted as MQ in Figure 4. As illustrated in this figure, high availability, adaptability, and flexibility of system configuration can be facilitated. As an example, let us assuming that education services are considered as the critical activities within the education brokerage domain. By allowing these activities to be distributed and replicated to different nodes (C and C'), system failure at node C will not disrupt the other nodes from executing their own part of the process.

To this point, what we have described the various concepts associated with the conventional workflow model, and what we considered to be a top-down approach to distributed workflow. The next section provides reasons why this top-down approach cannot be used to support the applications that we are hoping to support.

3.5 Problems & Objectives

From our observation, we believe that most commercial workflow products on the market today are geared toward the management of workflows which can be easily captured and readily distributed in a top-down fashion. This trend has successfully served, and continues to serve, a large percentage of workflow applications. However, the rapid maturity of the Internet, distributed object computing, electronic commerce, and groupware technologies, has introduced a new set of applications which require a new approach to distributed workflow. Because of the undetermined nature of the business processes associated with these new applications, a bottom-up approach to distributed workflow is required. As an example, one such application is the electronic education brokerage on the Internet (Hämäläinen et al., 1996). In this environment, an individual may want to participate either as a course customer or a course provider, based on the individual's needs and the ability to provide and receive trade information. Often, the information flow between the course customer and provider is not apparent up-front but rather through a series of trading steps. We envisage that the top-down approach, in its current state, is not suitable to support this kind of application. Instead, a new approach to distributed workflow is needed.

- **To support incremental trading:** The top-down approach may not be suitable for applications that do not have access to a well defined business process up-front. Rather, the business process is hidden from casual users

and nodes, and can only be available, gradually, as the trading continues. In the context of electronic commerce, we observe that business processes are often viewed as black-boxes. These black-boxes have been unfolded and used for trading after a series of elaborate negotiations and a high start-up cost. The problems associated with the so called 'close' trading are well recognised in the electronic document interchange (EDI) domain (Kimbrough and Moore, 1992; Knoppers, 1992). To justify the cost, only large volume and frequent trade activities can afford the luxury features provided and supported by the EDI technologies.

We hope, however, by encouraging open tradings in which trade activity that may happen only once (eg. buying a house), once in a while (eg. applying to a university), or on a regular basis (eg. obtaining education services), to support the next generation of electronic commerce applications where trades can be instantiated with a low start-up cost and variable trade volume.

- **To support critical applications:** The monitoring and execution of a workflow that is served from a centralised node can cause communication bottleneck and is not as resilient to crashes and trade failures. The centralised approach is less scalable and flexible to system configurations and thus cannot be used to support critical applications where high availability is a must. For example, to support the monitoring of exams and retrieval and updating of students' information during peak periods such as registration week, product release day etc., we need to provide duplication of services by replicating nodes which require high availability. Here, we hope to avoid the communication bottleneck situation similar to the release of Window95 in which during the first few weeks of its release, network communication at the release sites was congested and blocked up with hundreds of thousands of global requests.

The top-down distributed approach can be used to avoid some of the problems described above. However, due to the many-to-many nature of this approach, the coordination and management of the worklists and transition conditions are considered complex and difficult to manage. As an example, a worklist may contain a collection of work items (activities). A work item may appear in several other worklists. Any execution of the activity associated with these worklists need to update their status. This is due to the fact that the status of the work items (eg. starting and completing conditions) are generally stored at different nodes. To properly manage these work items, to coordinate and keep track the worklists' status, a one-to-many message protocol is required.

As an example, let us assume that there exist tasks T1,...,T7 in node A, B, and C as depicted in Figure 4. The following worklists: W_A(T1,T4), W_B(T1,T6), W_C(T1,T5,T7) belong to the different participants associated with A, B, and C, respectively. Note that activity T1 appears in all worklists W_A, W_B, W_C. In a sense, any one of the above three worklists can start to execute T1. If this happens, T1 needs to be removed from other worklists so that these worklists can re-schedule their work items. In order to do this, node A should contain information regarding which other worklists it has to inform about the execution of its activities (eg. T1). This example illustrates the one-to-many nature associated with the scheduling and management of the activities of a node.

In this paper, we use the term 'document-centric' to refer to the bottom-up, decentralised, step-by-step approach to workflow. In contrast, we use the term 'process-oriented' to refer to the centralised, top-down approach to workflow.

In summary, the centralised approach is vulnerable to communication bottleneck; it is neither very flexible nor scalable, and it cannot be used to isolate node failures from affecting other nodes. To avoid these problems, a top-down distributed approach can be implemented. However, this approach requires that the knowledge of a workflow be available up-front. Thus, it cannot be used to support applications in which the business process is undetermined in nature. Furthermore, due to the many-to-many relationships (eg. communication between distributed nodes) inherent in the process-oriented approach, the complex coordination and management of worklists and transition conditions may prevent an optimal solution.

Hoping to overcome these problems, the next section presents a bottom-up approach to the distributed workflow.

4 A bottom-up approach to distributed workflow

Contrary to the top-down approach, the bottom-up approach views the documents as key resources. With gradual integration steps, an integrated workflow can be fabricated.

The idea is to use the definitions and semantics embedded in the dossier to facilitate the routing of the dossier's components and constituent parts. In the context of the document-centric paradigm, every workflow application starts out with a document instead of an overall process.

Detailed description of the dossier, associated principles and their functionalities can be found in (Wing and Colomb, 1997). In this paper, we focus on the distributed features and implementation issues associated with the dossier in such a way that they can support the monitoring and execution of the workflow in a distributed manner.

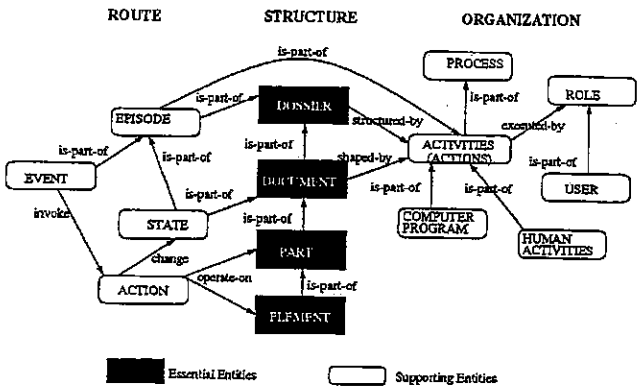


Figure 5. Elements of document-oriented model

4.1 Workflow concepts in the document-centric model

The core of the document-oriented workflow application is the documents themselves, an enclosed collection of documents form a dossier (eg. a folder contains university related documents). A document may be composed of many different parts (eg. a university application may consist of personal information, educational background, and needs). Similarly, each part may be composed of many different primitive elements (eg. personal information may consist of name, address, date of birth, etc.).

In the context of a workflow, as shown in Figure 5, a dossier can be formed by a collection of episodes and documents. An episode defines the complete semantic unit for executing a particular set of events in which various actions are to be employed onto a set of documents (eg. attach a solution header to a test document). Sometimes it is necessary for actions to be imposed on the documents' elements (eg. update the email address of a solution header). In a sense, the episodes can be used to capture the dynamic aspects of a business process. Related work (Bons et al., 1994; Wing and Colomb, 1997) uses these episodes to fabricate the so called Documentary Petri-Net (DPN). In the context of electronic commerce, the episode specifies the vital EDI messages which are necessary to carry out the collaborated activities. Each complete semantic unit of the episode is either executed completely or not at all. It is designed to support the commit, abort and rollback notions associated with trade documents.

In order to support the organisational aspects of the workflow, the process, activities, role, and user attributes can be part of the dossier's supporting elements. Their relationships with the dossier and its constituent parts are illustrated in Figure 5. Illustrated in this figure are the underlying supporting elements of our proposed workflow management system which are centred around the dossier and its documents. This is what the document-centric paradigm is based on.

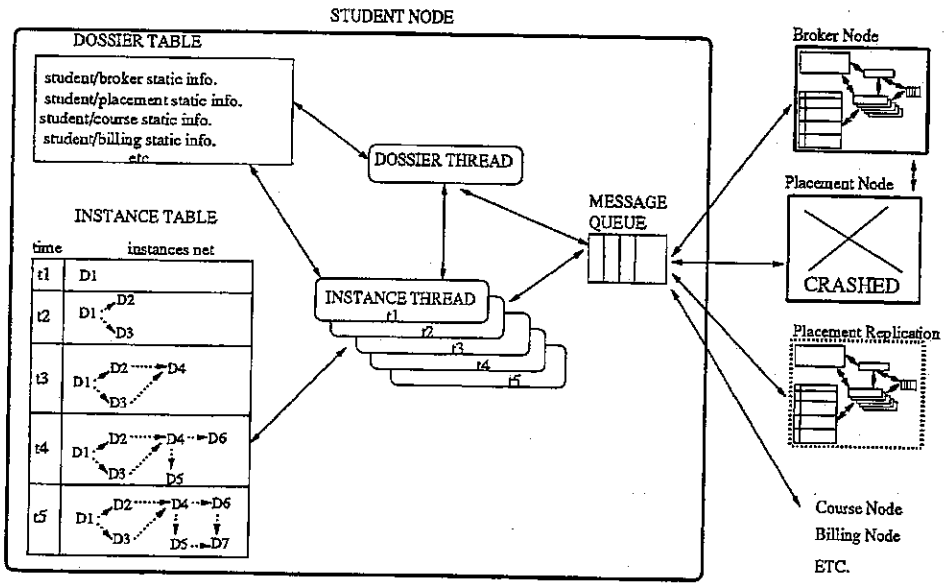


Figure 6. Execution components in the one-to-many, bottom-up distributed architecture

In short, we can view the document-centric model as if it has 3 distinct underlying supporting features: 1) The ROUTE feature comprised of all components shown in the left portion of Figure 5; 2) the STRUCTURE feature comprised of all elements shown in the centre column, and; 3) The ORGANISATION feature comprised of all components shown in the right portion of Figure 5.

We have introduced the basic workflow concepts associated with the document-centric model. Next, we examine the distributed feature associated with the document-centric model.

4.2 Distributed workflow: a bottom-up view

In the document-centric model, the monitoring and execution of a workflow does not need to be decomposed and distributed as it does in the top-down approach. Each dossier contains the necessary routing definitions to direct the documents to the desired places, and the necessary semantics to invoke the required actions to be employed in its constituent parts. By definition, each dossier by itself, is a distributed component of a workflow. Each dossier is thus in charge of its own part of the process. To exhibit the one-to-many relationship, it acts as the node manager to direct the control and data flow associated with other dossiers.

A node may contain one or more dossiers. To give an illustration, Figure 6 depicts the different components associated with the execution of a dossier. During the execution of a dossier, each dossier creates a dossier table which contains the static information describing the various relationships between that dossier and other related dossiers (residing in other nodes). A dossier table (depicted in Figure 7) describes the static information about that dossier's structure, route and organisational aspects (meta data) associated with a node.

In general, the meta data can be imported from a dossier repository of a particular type of trade (eg. education brokerage). Based on the node's role and usage, a particular type of dossier can then be copied down to the node environment. For example, a participant may make a request to the education brokerage repository for a particular set of dossiers which can support the multiple role types. As illustrated in Figure 1, the depicted workspace may support 3 different roles: course customer and course provider roles associated with the education brokerage and home-buyer role associated with the electronic-home-shopping trading environment.

Associated with each dossier table is a dossier thread. The dossier thread is used to manage the execution of a dossier's instances. The dossier thread uses the information derived from the dossier table to communicate with other nodes. As previously mentioned, a bottom-up approach to distributed workflow can be formulated after a series of trading steps. This can be supported by the instance table shown in Figure 6. Stored in this instance table are the different trading steps (eg. t1, t2, ..., t5), and the run-time information (eg. initial state, final state, list of documents, and actions etc.) associated with the dossier's event. In order to simplify the illustration, details of the instance table are graphically depicted as a series of instance nets which stores the workflow information at a particular point in time.

Shown in Figure 6, corresponding to each trading step (of the instance table) are the associated dossiers and their instance threads. The instance threads are used to manage the execution of the dossier instances. Note that the execution of a dossier does not require the existence of the overall business process. However, an audit trail can be formulated based on the information available from the instance table.

Also shown in Figure 6 is that, the dossier and instance threads communicate (eg. issue PUT/GET calls) with other nodes via the message queue. In this way, a node failure relating to any particular instance can be recovered by the persistent feature of the message queue. Static information of the dossier table does not have to be stored in persistent storage since it can be again downloaded from the dossier repository. We want to illustrate that in order to be resilient to node failure, we use the transactional PUT and GET calls of the message queue to recover the messages that were stored in the queue during the crashes.

Hoping to provide high availability of the process execution to the critical nodes, we can replicate the dossier with identical functionality and distribute it to several different nodes. In this way, communication bottleneck or crashes associated with the critical nodes can be minimised due to the additional supports of the replicated nodes. For example, during the first week of a registration period, several more placement service nodes can be replicated in order to provide sufficient advice service to the new students before they can be assigned to a course.

Information stored in the instance table can be used to isolate node failures from affecting other nodes. For example, if a crash occurred in a placement node, then all other nodes (eg. broker, course etc.) can still be executed. This is due to the following two factors: First, the execution of an instance thread is independent from other instance threads. Second, the data and control flow associated with the dossier of the crashed node can either be diverted (to its replication) or suspended until the node is later started and recovered. As an example, Figure 6 illustrates that if a crash occurs at the placement node, the functionality of the placement service can still be carried out in the replicated node(s).

One important aspect of the bottom-up approach is how a workflow can be incrementally updated. In order to do this we must know when an updated workflow of a trade domain can be used by another trade domain. To

do so we need to know how to integrate a new term t_1 from a source vocabulary V_1 to a destination vocabulary V_2. (Mineau, 1992) has shown that a term from one knowledge domain can be integrated into another knowledge domain by using the conceptual graph formalism (Sowa, 1984). Incorporating this formalism into the context of our work, the vocabulary of a trade domain describing the standard trade messages and procedures can be defined by the concept, conceptual relation, and part-of hierachies. The concept hierarchy describes the events and states associated with the trades. The conceptual relation hierarchy describes the associations between the trades' states and events. Finally, the part-of hierarchy describes the vocabulary that makes up the terms and concepts relevant to the trades.

Basically, a workflow can be incrementally updated when each of its subsequently introduced terms (eg. $t_1, ..., t_n$), defined by V_1, can be gradually integrated, step-by-step into V_2 by using the terms and concepts that already defined in V_2. As latter shown in section 6, an overall process-oriented FDL graph can be generated by gradually integrating the partially formed DPNs.

In our implementation, the data integrity associated with the document update can be enforced by using the document links. For example, a student's address may be changed at any time and other documents can refer to this new information via the documents' pointers.

In this section, we have described the basic elements and components associated with the execution of the bottom-up approach. In particular, we have presented a way to facilitate the so called incremental approach to distributed workflow. In this approach, crash failures can be isolated; replication can be used to support critical applications; process execution can be flexibly partitioned into many different clusters of configuration; and in the presence of node failures, transactional recovery can also be supported. The next section examines how these features may be implemented.

5 Implementation issues

Since the workflow reference model does not provide the implementation detail, we use the IBM/FlowMark FDL to implement the workflow of the dossiers, and use the Lotus/Notes as an implementation to the dossier parts and primitive elements.

5.1 Workflow concepts in Notes

A Notes application consists of: forms, views, and documents. Documents are stored in Notes databases. Views are used to list the documents. Forms can be used by the user to edit the documents. Associated with each document are the various agents stored in the databases. There is a similar trigger mechanism as that in active database systems (Chakravarthy et al., 1994; Dayal et al., 1988). Each agent consists of name, EVENT (to tell when to start and stop), LIST (to tell which documents are to be acted on), ACTION (to tell what actions are to be taken). These are notions that are employed to implement the routing of workflows, and are used to send and perform periodic and non-periodic notices to places and tasks.

Lotus actions can be either simple, formula, or lotus scripts type. User defined event is supported in the producer/consumer fashion. Event producer puts an event in an event queue whereas event consumer removes an event from an event queue and acts on it appropriately. This is similar to the PUT and GET calls associated with the message queues.

Below are some of the main features that are supported by the Note/Release 4 version.

- Call back routine which can be used to activate before and/or after an internal operation.
- Notes documents are collapsible in forms depending on the workflow participant and states.
- Workflow data created by any OLE2 enabled applications are stored in Notes documents and databases.
- Notes can support authorisation/authentication, RSA-based encryption and digital signature.
- Notes can support replication for mobile computing.
- Notes can provide links to databases and documents. In other words, container can send the actual document, its replication, or the doc's link that points to it.

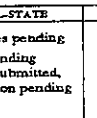
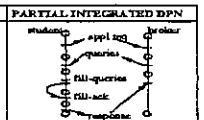
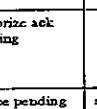
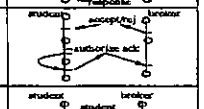
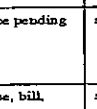
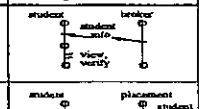
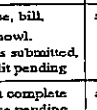
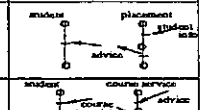
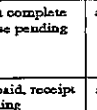
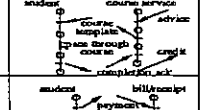
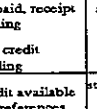
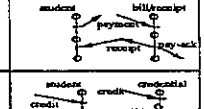
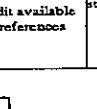
There are some problems associated with Notes that must be considered: 1) Notes does not support nested type; 2) Notes does not support hierarchical organisational information or role information. To support this it is required that an additional organisational database be available at runtime. For this reason, FlowMark FDL organisational capabilities (eg. role, levels of expertise etc.) can enhance the Note address book's basic functionality.

One of the important features of Notes which can help us to implement the transactional features is that the concurrent update conflict associated with Notes documents can be detected. A Note document is flagged if it is involved in some kind of update or replication conflict.

5.2 Modelling workflow with FlowMark's FDL

The navigation and execution of FlowMark can be described as follows:

- Flow of control (CC) corresponding to transition condition.
- Input container.
- Output container.
- Flow of data (DC) corresponding to relevant data.
- Start condition corresponding to the pre-condition of activity.
- Exit condition corresponding to the post-condition of activity.

DOSSIER	PART	EVENT	INIT-STATE	ACTION	WHICH PART	FINAL-STATE	RELATIONSHIP	DOSSIER GRAPH	PARTIAL INTEGRATED DPN
university-application D1	request-ltr queries response fill-ack	queries arr. fill ack.	queries pending fill pending	send fill send populate	request-ltr application response decision ltr student info	queries pending fill pending appl submitted, decision pending	student/broker		
Notification D2	accept msg reject msg	accept/rej msg arrive authorize ack arrive	decision pending authorize ack pending	prepare send populate	authorize ack advice sheet	authorize ack pending	student/broker		
student info D3	name, addr, background, needs, etc.	student-info arr.	student-info pending	view & verify populate	student-info advice-sheet	advice pending	student/broker		
advice sheet D4	advice info	advice-info arrive	advice-info pending	forward populate	advice-info course template bill/receipt	course, bill, acknowl. solns submitted, credit pending	student/placement service		
course/test template D5	course, assignment, test, tutorial	course component arrive	course pending	pass through populate	course/test template credential	unpln complete course pending	student/course service		
bill & receipt D6	bill, receipt, payment	bill/receipt arrive	bill/receipt pending	send forward populate	payment receipt credential	bill paid, receipt pending paid credit pending	student/billing service		
credential D7	credential, receipt paid credit	credit arr. paid credit arrive	credit pending paid credit pending	verify market	unpaid credit paid credit	credit available for references	student/accreditation service		

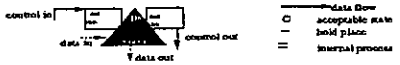


Figure 7. Dossier table

Work described in (Reinwald and Mohan, 1996) has shown that FDL definitions in FlowMark can be mapped down to Notes's native workflow objects. The modelled workflow defined in FlowMark can be compiled into FlowMark run-time objects. These runtime objects can then be mapped into Notes run-time objects. In our dossier model, the business process embedded in a group of dossiers can be mapped back in the reverse direction.

There are some minor problems associated with FlowMark that we must also consider and find solutions for: FlowMark FDL does not support external events, ad-hoc routing and loop.

We have shown how the document-centric, a bottom-up approach to distributed workflow can be implemented. In the next section, we describe how the proposed model can be used.

6 An example of bottom-up distributed workflow

This section provides a bottom-up example and discussions related to the business process shown in Figure 2.

1. Before any trade can happen, user groups of an trade organisation define the relevant trade documents and their dossiers.

2. Based on the functionality of the organisations (eg. education broker), their service types (placement service), and their trading purposes (eg. generating advice), a collection of documents can be classified into a group of dossier types, ready for distribution.

3. Upon retrieval of the dossiers and their constituent parts, a user (eg. a student) can freely use them to instantiate trades. This process can be as simple as sending a message (eg. university application request) in the conventional electronic mail.

4. On the receiving end of this trade, depending on the receiver's (eg. university broker) capability and functionality (this knowledge is embedded in the broker's dossiers), the request message can be used to instantiate and generate other related documents (eg. application queries, accept/reject messages, etc.) or to be returned to the sender in case of a dead path situation. A dead path situation (eg. a reversal of dead path elimination process

of the top-down model) occurs when the request message cannot possibly instantiate any other related documents due to the lack of knowledge in understanding each other's trading capability. The dossier D1 and D2 of Figure 7 show the components associated with the admission process of the student/broker relationship. As shown in this figure, is a partially integrated petri-net (DPN) associated with each dossier, used to model the workflow of that part of the process.

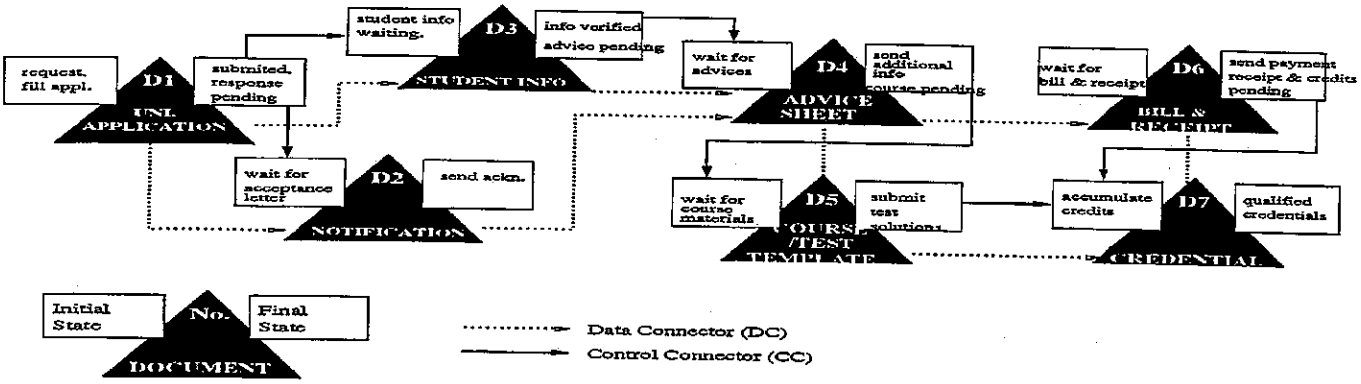


Figure 8. Formalising the example by using the document-oriented FDL

5. Step-by-step, documents are continuously exchanged between the dossiers until one of the following two possibilities occurs: First, if a dead path situation is reached, an audit trail can be built and analysed in order to determine the next necessary actions involved in responding to the received documents. Second, if one of the participating nodes crashes, the other remaining nodes will have to isolate their trades from the crashing node. This can be done by simply not sending/receiving documents to/from the crashing node. Recovery operation can be started as soon as the crashing node is up and running again. The documents and messages involved in the crash can be easily recovered. This can be supported by the persistent message queues as described in section 3.4. In this manner, a node failure cannot affect the entire workflow.

6. Since each dossier of different nodes minds its own part of the workflow, network communication is no longer imposed onto any one particular centralised node, but is rather distributed among all of the participating nodes. The communication bottleneck of the centralised model can thus be avoided. Furthermore, each node by deciding on which types of the dossiers it willing to support can now have a choice in specifying the communication load which it can handle.

7. To illustrate what we call non-definitive trading, let us assume that the student's admission is accepted with a provision that he/she must sign up for a certain required courses. This can be done by instantiating a new document to specify the courses. This document can then be attached to the advice sheet as one of its contingencies. Dossier 3,4, and 5 of Figure 7 show the relevant documents and components associated with the placement and course signing up process.

8. The above steps suggest that arms-length trade interoperability can be done if the trade definitions and semantics can be gradually revealed, step-by-step. With the support of the traded documents, business processes can be formulated in a bottom-up fashion. For example, after a series of continuous trading steps involving all of the dossiers shown in Figure 7, an integrated dossier graph can finally be formulated (as shown in Figure 8). The dossier graph is a graphical representation of a document-oriented flow definition language (FDL). This graph illustrates the detailed structure of the 'course-customer' dossier shown in Figure 1. In general, the document-oriented FDL graph reflects the perspective of an end user (eg. an course customer) whereas the process-oriented FDL graph, as shown in Figure 4, reflects the perspective of a workflow designer.

9. As an example of incremental trading, a combination of all of the partially integrated DPNs shown in Figure 7 will produce an overall workflow. The integrated DPN shown in Figure 9 is equivalent to the process-oriented FDL graph shown in Figure 4. We have shown that an integrated workflow can be formulated from the step-by-step integration of smaller sub-nets, thus illustrating the bottom-up approach to distributed workflow.

The modelled FlowMark workflow can be compiled into FlowMark run-time objects. These runtime objects can then be mapped into the Notes run-time objects.

In the reverse direction, in the dossier construct, the business process defined in Note native FDL embedded in a dossier can be mapped back onto FlowMark FDL. This is because the workflow defined in Notes can be compiled into Notes run-time objects. These objects then can be mapped into FlowMark run-time objects as if they have been originally specified in the FlowMark native FDL. Shown below is a sample of what FlowMark FDL may look like in describing the business process shown in Figure 4.

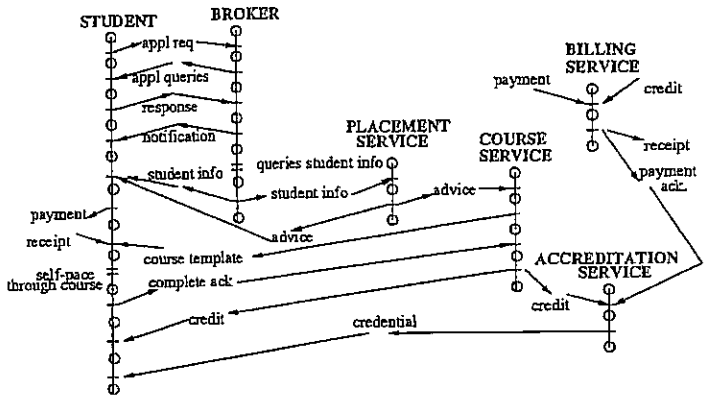


Figure 9. Integrated DPN formulated from the dossier tables shown in Figure 7

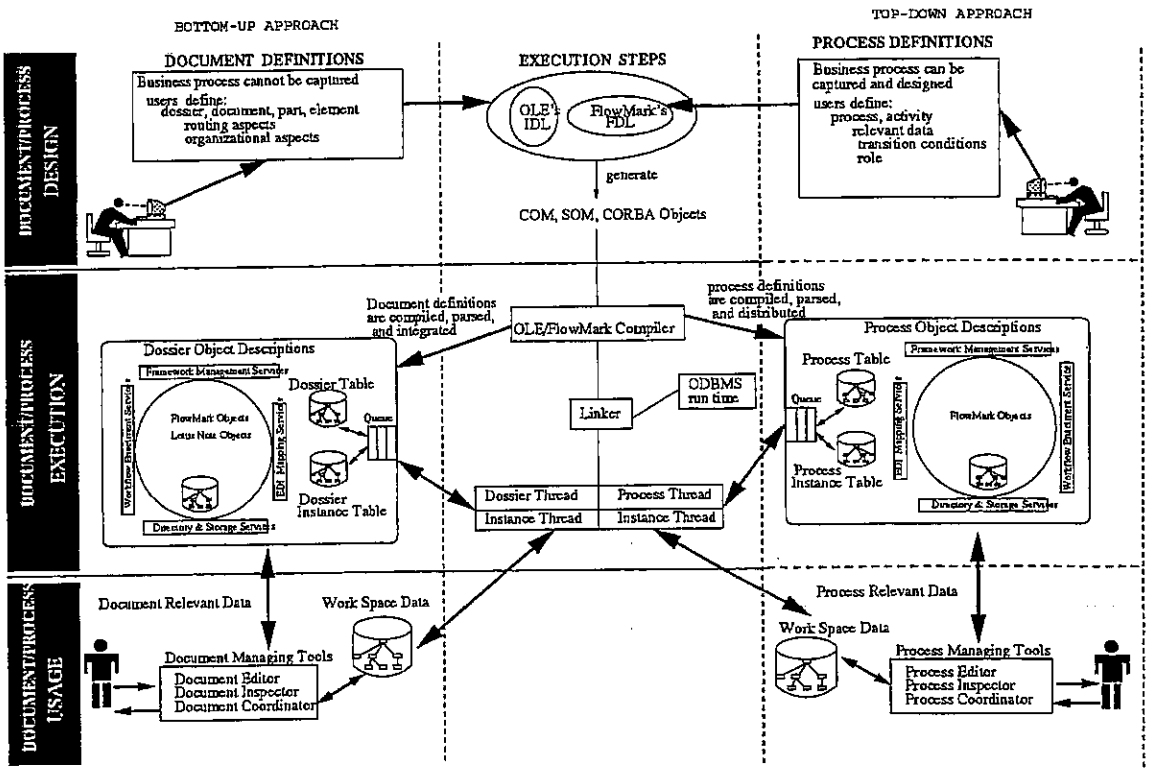


Figure 10. Building an open trade application

```
STRUCTURE 'Education-on-WWW'
    'Learner'     : 'CustomerInfo';
    'Instructor'  : 'ProviderInfo';
    'Course'      : 'CourseTemplates';
    'Old/New'     : STRING;
    'Accept'      : STRING;
    ...
END 'Education-on-WWW'

STRUCTURE 'CustomerInfo';
  ...
END 'CustomerInfo'

      ...

ROLE 'COUNSELOR'
  ...
END 'COUNSELOR'

ROLE 'ADMINISTRATOR'
  ...
END 'ADMINISTRATOR'

PROGRAM 'ApplicationFormEditor' ('Education-on-WWW', 'Education-on-WWW')
  ...
END 'ApplicationFormEditor'

      ...

PROCESS 'Education-on-WWW'
PROGRAM_ACTIVITY 'AdmissionApproval'('CustomerInfo','AdmissionInfo')
PROGRAM 'ApplicationFormEditor'
  ...
END 'AdmissionApproval'
  ...
CONTROL NAME 'UpdateCustomerInfo' FROM 'GetInfo' TO 'Insert'
    WHEN 'old/new' = "new"
CONTROL NAME 'RetUpdateCustomerInfo' FROM 'GetInfo' TO 'QueryUpdate'
    WHEN 'old/new' = "old"
CONTROL NAME 'counselling' FROM 'Admitting' TO 'Placement'
    WHEN 'Accept' = "yes"
CONTROL NAME 'PayingFees' FROM 'Placement' TO 'Billing'
CONTROL NAME 'AssignToCourse' FROM 'Placement' TO 'EducationService'
CONTROL NAME 'ValidateCredits' FROM 'Billing' TO 'Accrediting'
CONTROL NAME 'ReportCredits' FROM 'EducationService' TO 'Accrediting'
  ...
DATA FROM 'GetInfo' TO 'Query&Update'
DATA FROM 'GetInfo' TO 'Insert'
DATA FROM 'Admitting' TO 'Placement'
DATA FROM 'EducationServices' TO 'Accrediting'
  ...

END 'Education-on-WWW'
```

Building trade application

The following steps describe how an open trade application can be built. Depending on the nature of the business, users can build a trade application based on either the bottom-up or top-down approach. The first approach is used when the business process is more obscured and undetermined, otherwise the second approach can be used.

Step 1: Document/Process design

Trade documents can be designed and defined with either the top-down or bottom-up approach. The top-down approach favours a close-knit group of traders (eg. committee of stock brokers) where the overall information and central policy can be carefully designed and governed. In contrast, the bottom-up approach is designed for a situation in which the individual trader is able to participate in a trade, with other partners, without being required to engage in an elaborate negotiation in advance.

In either case, a trade designer or a programmer who is familiar with one of the object-oriented programming language such as C++ or Smalltalk can design and specify the trade objects based on the FlowMark's FDL, OLE's IDL or a combination of both. These trade objects specify the document's declarations, process workflow definitions, and EDI class hierarchies. Figure 10 shows how the data definitions and execution flow can be carried out in accordance with this proposal.

Step 2: Document/Process execution

The Lotus Notes, OLE definitions or FlowMark classes can then be compiled and parsed by the appropriate (eg. Note or FlowMark) compiler to generate inter-operable entities such as FlowMark and Note objects. The architectural components such as workflow enactment services, framework management services, directory and storage services, and EDI mapping services, interact with these objects, workflow definitions in the Dossier table, and the run-time information in the Instance table to provide an array of useful services to the end users (eg. trade participant and workflow designer).

Step 3: Document/Process usage

The end users of a trade document (or process) are the various role players (eg. home buyer, real-estate agent, bank manager etc.) associated with the trade. These users can interact with the trade documents by activating the various document managing tools such as document's editor, inspector and coordinator.

7 Pros and Cons

In this section we provide some comparison and discussions about the above described approaches to distributed workflow.

Document-centric workflow model allows incremental trade development: The document-centric type of programming shifts the controls from the application programs to the documents themselves. A document contains all of the necessary information about itself and about what types of actors it can serve and how it can be operated. Consequently, the document's behaviour can be adapted to each of the associated roles and actions.

The solution here is to allow a document to be composed of the various constructs such as Frameworks, Parts, and atomic Elements. For example, the documents (including the flows, activities, roles etc.) associated with an education brokerage can be formulated as a framework. Course admission, participation, assessment, and accrediting tasks can be formulated as the framework's constituent parts. Consequently, the course profile, question templates and annotated solutions can be formulated as the atomic elements of the assessment part. Since the overall controls and underlying taxonomy are now embedded in the dossier's constituent parts and documents, new updates can be accommodated without changing the application codes.

Listed below are some of the pros (depicted as +) and cons (depicted as -) associated with the process-oriented model. Factors that are not so obviously pros or cons are depicted as *.

Top-down (process-oriented)

- If control and diverse information is contained in the application codes (true for earlier workflow systems), applications need to be rewritten, recompiled, and linked in order to accommodate changes.
- It is more vulnerable to communication bottlenecks.
- It is not resilient to node failure (if it is non-distributed).
- It has low scalability therefore it cannot be used to support critical applications which require high availability.
- It must support a large volume of trades in order to justify the high start up cost and elaborate negotiations.
- It is harder to support incremental trading (trading on the fly).

* It is quite simple to support business processes that can be clearly defined in advance.

* It is less natural to the end users (trade participants); however it is more natural to the process designer (eg. workflow programmers, and managers).

* If control and diverse information is contained in the processes' definitions (true for modern workflow systems), changes can be accommodated without many problems. However, it cannot provide a finer granularity of control as in the case of the document-centric model.

+ It can support distributed features.

+ The process is well understood hence it is easier to monitor and test.

Bottom-up (document-centric)

- Since the business process is fuzzy, knowledge of the workflow is harder to capture and represent.
- Without the overall workflow, it is more difficult to simulate and test.

* It is more natural to end users; however it is less natural to the process designer.

* The initial start up cost and negotiation must be small enough to justify infrequent trading (eg. buying a house may only happen once).

+ It has high scalability and availability (to support critical application).

+ It is more convenient to adapt to changes.

+ Control and diverse information is contained in the documents' definitions hence providing a finer granularity in updating and control.

+ It can support incremental trading.

Definitive vs Indefinitive trade information

In a definitive trade environment, FWMS which is based on the process-oriented model is considered the best candidate to support a closed trading, since the workflow can be more or less definite and well tested beforehand. As an example, a travel request process is considered to be well understood within an organisation, and similarly an order/purchasing process should be well established before a purchase order can be submitted.

In general, we can assume that a 'reasonable' travel request will have a better chance for approval, and that an 'unreasonable' travel request will not be approved. Judgement on whether or not a travel request is reasonable very much depends on how much knowledge about the travel funding criteria can be known in advance. In the top-down approach, this definitive knowledge is a must in order to generate rules which enforce the process flow.

In contrast, open trading often leads to obscure knowledge about the trade information since definitions and the semantics of the workflow may not be available up-front. As an example, a student may submit his personal profile, education needs and background to an education placement service without really knowing what kind of advice (or contingencies) he/she may get in return. This is due to the fact that different institutions may have different placement policies and internal factors (eg. lack of courses) which may affect the outcomes. Often,

these policies and factors are 'black-boxes' to the applicants. The point is that in the document-oriented model, it is natural to model the exceptions related to trades. Where the rules to represent the contingencies may not be known in advance, the top-down approach is not very useful. However, the bottom-up approach is more suitable for supporting the incremental trade applications.

8 Conclusion

We have attempted to present a bottom-up approach to distributed workflow. We reason that the conventional top-down approach may not be feasible in supporting the so called 'arms-length' trade interoperability associated with open electronic commerce. This is due to the fact that most business processes are viewed as black-boxes by all interchanges, at least in the beginning. However, with gradual tradings and negotiation steps, the business processes should be more relevant, and transparent, to the monitoring, execution, and management tools. In a sense, we hope through an incremental trading and negotiation steps, to provide a baby-stepping approach which facilitates trading in which the overall business process is not so well understood up front.

In order to avoid communication bottlenecks associated with the centralised workflow, we use a collection of persistent document-oriented dossiers which can be stored and executed at the different nodes. In doing so, node failures can be isolated from affecting others from performing their work.

Finally, by using the persistent message queues to facilitate communication between dossiers which reside in different nodes, asynchronous communication between the applications that run at different points in time can be supported. Furthermore, in the presence of node failures, recovery can also be supported due to the transactional PUT/GET calls and the persistent feature of the message queues embedded in the dossier constructs.

Our future work is to extend the CORBA's IDL and its transaction and concurrency services to accommodate the transactional capability and asynchronous communication associated with the document-centric, bottom-up approach.

References

Alonso, G., Agrawal, D., Abbadi, A., Kamath, K., Günthör, R., and Mohan, C. (1995). Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. In *Proc. IFIP Working Conference on Information System for Decentralized Organizations.*

Bons, R., Lee, R., Wagenaar, R., and Wrigley, C. (1994). Modelling Inter-organizational Trade Procedures Using Documentary Petri Nets. In *The Hawaii International Conference on System Sciences.* about DPN.

Breitbart, Y., Deacon, A., Schek, H.-J., Sheth, A., and Weikum, G. (1993). Merging application-centric and data-centric approaches to support transaction-oriented multi-system workflows. *SIGMOD Records*, 22(3).

Chakravarthy, S., Answar, E., and Maugis, L. (1994). Design of sentinel: an object-oriented dbms with event-based rules. *Information and Software Technology*, 36(9):555–568.

Dayal, U., Buchmann, A., and McCarthy, D. (1988). Rules are objects too: A knowledge model for an active, object-oriented database system. In Dittrich, K., editor, *2nd Int. Workshop on OODBS*, pages 129–143.

Elmagarmid, A., editor (1992). *Database Transaction Models For Advanced Applications.* Morgan Kaufmann.

Elmagarmid, A. K., Leu, Y., Litwin, W., and Rusinkiewicz, M. (1990). A multidatabase transaction model for interbase. In *Proc. of the 16th VLDB Conference.*

Garcia-Molina, H. and Salem, K. (1987). Sagas. In *Proc. 1987 SIGMOD International Conference on Management of Data.*

Georgakopoulos, D. and Hornick, M. (1994). A framework for enforceable specification of extended transaction models and transactional workflows. *International Journal of Intellegent and Cooperative Information Systems*, 3(3):599–617.

Georgakopoulos, D., Hornick, M., and Sheth, A. (1995). An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153.

Goldberg, Y., Safran, M., and Shapiro, E. (1992). Active Mail - A Framework for Implementing Groupware. In *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW), Toronto, Canada.*

Hämäläinen, M., Whinston, A. B., and Vishik, S. (1996). Electronic Markets for Learning: Education Brokerages on the Internet. *Communications of the ACM*, 39(6).

Hsu, M. (1993). Special Issue on workflow and extended transaction systems. *Bulletin of TC on Data Engineering*, 16(2).

IBM (1993). *Message Queue Interface: Technical Reference*. IBM Document No. SC33-0850-01.

IBM (1995). *FlowMark - Modeling Workflow, Version 2.1*. IBM Document No. SH19-8241-00.

Kimbrough, O. S. and Moore, A. S. (1992). Message Management Systems: Concepts and Motivations. In *Proceedings of the Hawaii International Conference on System Science*.

Knoppers, J. (1992). Important of the "OPEN-EDI" reference model from a user and business perspective. In *Proceedings of the Interorganizational systems in the global environment, Bled.*

Kuo, D., Lawley, M., Liu, C., and Orlowska, M. (1996). A General Model for Nested Transactional Workflows. In *Proc. of the International Workshop on Advanced Transaction Models and Architectures.*

Leymann, F. and Roller, D. (1994). Business processes management with flowmark. In *Proc. 39th IEEE Computer Society International Conference (CompCon), Digest of Papers, San Francisco, California, IEEE*, pages 230–233.

Lotus Notes, Cambridge, M. (1995). *Lotus Notes Developer's Guide, Version 4.0*. Lotus.

McCarthy, D. and Sarin, S. (1993). Workflow and Transactions in InConcert. *Bulletin of TC on Data Engineering, Special Issue on workflow and extended transaction systems*, 16(2).

Medina-Mora, R., Winograd, T., Flores, R., and Flores, F. (1992). The action workflow approach to workflow management technology. In *ACM 1992 Conference on Computer-Supported Cooperative Work: Sharing Perspectives.*

Miller, J., Sheth, A., Kochut, K., and Wang, X. (1996). CORBA-Based Run-Time Architectures for Workflow Management Systems. *Journal of Database Management, Special issue on Multidatabases*, 7(1).

Mineau, G. W. (1992). Sharing Knowledge: Starting with the Integration of Vocabularies. In Pfeiffer, H. D. and Nagle, T. E., editors, *Conceptual Structures: Theory and Implementation, the 7th Annual Workshop.*

Reinwald, B. and Mohan, C. (1996). Structured Workflow Management with Lotus Notes Release 4. In *Proc. 4th IEEE Computer Society International Conference (CompCon), digest of papers.* available in http://www.almaden.ibm.com/cs/exotica.

Searle, J. (1969). Speech Acts: An Essay in the Philosophy of Language.

Sheth, A. and Rusinkiewicz, M. (1993). On transactional workflows. *IEEE Data Engineering Bulletin*, 16(2).

Sheth, A. and Rusinkiewicz, M. (1995). Specification and execution of transactional workflows. In Kim, W., editor, *Modern Database Systems: The Object Model, Interoperability, and Beyond.* AMC Press.

Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley Publishing Co.

Tsichritzis, D. (1982). Form Management. *Communications of the ACM*, 16(2).

Waechter, H. and Reuter, A. (1992). The ConTract Model. In Elmagarmid, E., editor, *Database Transaction Models for Advanced Applications.* Morgan Kaufmann.

WfMC (1995). *Workflow Reference Model Specification.* Workflow Management Coalition. http://www/aia.ed.ac.uk.wfmc.

Wing, H. and Colomb, R. M. (1997). An Architecture to Support Distributed Trade Documents. In *The Third International Symposium on Autonomous Decentralized Systems (ISADS97), Berlin, Germany.* IEEE Computer Society Press. To appear.