Establishing Virtual Enterprises by means of Mobile Agents

Panos K. Chrysanthis*, Taieb Znati
Computer Science Dept.
University of Pittsburgh
Pittsburgh, PA 15260, USA
{panos,znati}@cs.pitt.edu

Sujata Banerjee*
Info. Sci. & Telecom. Dept.
University of Pittsburgh
Pittsburgh, PA 15260, USA
sujata@tele.pitt.edu

Shi-Kuo Chang
Computer Science Dept.
University of Pittsburgh
Pittsburgh, PA 15260, USA
chang@cs.pitt.edu

Abstract

Electronic Commerce is expanding from the simple notion of Electronic Store to the notion of Virtual Enterprises (VE) where existing enterprises dynamically form temporary alliances, joining their business in order to share their costs, skills and resources in supporting certain activities. Two fundamental problems in VE are (1) how a VE is established and (2) how information is shared among the VE participants in a control and efficient manner. Currently, existing enterprises are using workflows to automate their operation integrating their information systems and human resources. Thus, in this paper, we view the establishment of a VE as a problem of dynamically expanding and integrating workflows in decentralized, autonomous and interacting workflow management systems. Its focus is on the idea of mobile agents called *adlets* and their use in establishing VEs that involves advertising, negotiating and exchanging control information and data as well as its management.

1. Introduction and Motivation

Electronic Commerce is expanding from the simple notion of Electronic Store to the notion of *Virtual Enterprises* (VEs) where existing enterprises dynamically form temporary alliances, joining their business in order to share their costs, skills and resources in supporting certain activities. An example of a VE in the context of the travel and tourism industry would be the collaboration of different travel agents, airliners, ground transportation services, hotels, restaurants and entertainment services in order to set up and manage a tourist tour. Two fundamental problems in a VE are (1) how a VE is established and (2) how information is shared among the VE participants in a controlled and efficient manner.

* This material is based upon work partially supported by NSF award IIS-9812532.

Currently, existing enterprises are using workflows to automate their operation integrating their information systems and human resources [15]. A workflow (also called business process) consists of a set of *activities* (also called *tasks*) that need to be executed in a particular controlled order over a combination of heterogeneous database systems and legacy systems. Within workflows, activities are performed cooperatively by either human or computational agents in accordance with their roles in the organizational hierarchy.

The challenge in facilitating the implementation of workflows has been in developing efficient *workflow management* systems. A workflow management system (also called workflow server, workflow engine or workflow enactment system) provides the necessary interfaces for coordination and communication among human and computational agents to execute the activities involved in a workflow and controls the execution orderings of activities as well as the flow of data and materials that these activities manipulate. (Materials are all the physical objects that typically exist outside of the computer system and require external support). Thus far, the research on the workflow management system has focused on techniques for correct and reliable specification, execution, and monitoring of workflows and the involved external support (e.g., [7, 5, 10]).

Very recently the idea of the use of workflows to support multi-organizational processes that form a virtual enterprise has attracted some attention [8]. Although, we share the same basic idea of integrating several heterogeneous workflow management systems with these efforts, in our work, we do not assume static, pre-negotiated component services from different enterprises that need to be controlled and monitored. In contrast, we view the establishment of a VE as a problem of dynamically expanding and integrating workflows in decentralized, autonomous and interacting workflow management systems. Thus, we are interested in supporting dynamic workflows both within individual enterprises to accommodate client preferences and requirements as well as across multiple enterprises to support *outsourcing* and form of an VE.

The focus of this paper is on the idea of information agents called *adlets* and their use in establishing VEs that involves advertising, negotiating and exchanging control information and data as well as its management. The basic features of the adlets are their ability to create personalized VE services, migrate from one location to another accumulating and integrating knowledge from clients and servers, and perform inferencing based on the acquired information.

Specifically, within our proposed framework, workflow activities and in particular outsourcing are implemented by adlets. One or more adlets, depending on the activity, are dynamically created to carry out that activity. In the case of outsourcing, adlets, which have the capability to advertise the profile (i.e., needs, skills, policies and capabilities) of their local server to a target set of remote servers, roam among a set of networked servers to discover new services and promising partnerships relevant to their local server, and engage in negotiation to achieve the best possible service for their clients. During the establishment of a VE, a distributed, multi-organizational workflow emerges from the dynamic growth and reconfiguration of workflows in the participating enterprises. Again using adlets, the VE can monitor and automatically keep track of the most relevant information for each workflow activity and allow component workflow management systems to share information and cooperate in an efficient manner. To ensure the scalability of the adlet framework, each adlet is associated with metadata that limits its roaming and advertising to a specific domain. Further, security concerns are handled by incorporating a credential authenticator in the adlet framework.

In the next section, we introduce our VE workflow model. In Section 3, by means of an example, we elaborate on the establishment of a VE using adlets and describe the major features of the proposed system. In Section 4 we provide a formal definition of adlets, discuss their basic operations and capabilities and describe the basic architecture to support these capabilities. In Section 5, we briefly discuss the implementation of our system using mobile agent technology. We conclude with a summary in Section 6.

2. VE Workflow Model

Current industries are utilizing workflows to automate their processes. A workflow specifies a set of activities that achieve a goal and their order of execution. A workflow management system (WFMS) provides a language and tools for the specification of workflow activities and controls the execution orderings of activities and the flow of data and materials among the activities.

A workflow activity is specified in terms of name, preconditions, actions, rules of exception handling, completion and temporal constraints. Every workflow specification formalism is built around three basic control flow relationships, which are *precedence*, *OR* and *AND* relationships [15]. OR and AND relationships are further refined into OR-split, AND-split, OR-join and AND-join. The first two relationships are used to specify branching decisions in a workflow whereas the remaining two specify points where activities converge to initiate the next activity within a workflow. An OR-join specifies alternatives whereas AND-join specifies required activities. Additional terminating constraints can capture more details, for example, *at-most-one* semantics.

In our work, we formally specify workflows and express the dependencies among their activities using ACTA [6], a first order predicate logic formalism with a precedence relation ($\rightarrow$). For example, a simple AND-join in a trip plan workflow can be expressed as an axiom:
$ReserveCar \wedge ReserveHotelRoom \wedge ReservePlaneSeat \Rightarrow MakeTripDecision$
whereas an OR-join with at-most-one semantics can be expressed by a pair of axioms:
1. $VisaCharge \vee MasterCharge \Rightarrow MakePayment$
2. $Commit(VisaCharge) \Leftrightarrow \neg Commit(MasterCharge)$.

A workflow can be graphically depicted with nodes (thick boxes in our figures) denoting activities and arrows denoting precedence. In figure 1 representing a vacation trip from [14], AND-splits (and AND-joins) are implicit when more than one arrow originate from (coincident to) a node. For example, *Get Input* represents an AND-split and Make Trip Decision represents an AND-join. OR-splits and OR-joins are depicted with arrows annotated with selection conditions. For example, *Make Payments* represent an OR-split with conditions **S** (Success) and **F** (Failure).

Any subgraph of a workflow graph defines a *segment* or a *view* of the workflow. Formally, a workflow view can be defined as a projection on the graph based on some criteria. Using the notion of a projection, then any workflow or workflow view can be decomposed into any number of workflow views whereas a node representing an activity is a trivial form of a view. For example, in figure 1, a shaded box can be viewed as a workflow view representing the subgraph within the corresponding rectangle.

In our VE workflow specification, we use the notion of views to express outsourcing. A workflow view can represent any activity performed by a *service provider* on behalf of a *service requester*. Consequently, workflow views can be used to express service requests. In our system, adlets advertise, request and negotiate workflow views. A requested workflow view can be potentially augmented during negotiation to match the service provider's workflow, reflecting opportunities, omitted activities and data. In this way, a VE is established between the service requester and service provider. During a negotiation an adlet may choose (based on its rules and discovered information) to negotiate

the entire advertised view or part of it by decomposing it into several views and seek other service providers for the other parts of the view. In this way, a single initial request may lead to the establishment of a VE comprising by multiple enterprises. A VE comprising by multiple enterprises can also be resulted when a service provider's view includes outsourcing.

In the next section, we elaborate on the vacation trip example to illustrate the establishment of a VE to support it.

3. Trip Reservation Scenario

Consider the (classical) example of an executive Jane Smith who is traveling from New York to Vienna to attend a meeting from October 11, 1998 to October 17, 1998. Jane wants to leave New York on October 11 and leave Vienna on October 17 and Jane must stay at hotel Maxim, the site of the conference. Jane prefers to fly on American, Delta, or United. Jane will not travel on any other airline. The car must be rented from Avis or National with which Jane's company has corporate accounts. If no flight or hotel is available, the whole trip is canceled. If a car cannot be rented, the trip can still proceed since Jane can take public transportation. Since Jane gets free frequent-flyer miles on Visa card charges, she prefers to charge as much as possible on her Visa card and uses American Express only if Visa is not accepted or Jane has exceeded her credit limit.

A workflow that can automate Jane's trip is shown in Figure 1. The workflow involves activities making flight and hotel reservations, optionally a car rental reservation and then charges the costs to a credit card. Each activity invokes database transactions one each for the possible candidates for a reservation until one of them succeeds. One way to arrange this trip is to first make the flight reservations based on some predefined order, and if they succeed then to try the hotel reservation, and finally, the car reservation. To reduce contention, each of these can commit the reservations independently, provided we can undo the reservation should the need arise, say if a hotel reservation is not possible. This calls for compensating actions corresponding to each of the three component activities. The above example illustrates a static workflow system.

To illustrate the major features of the adlet based dynamic workflow framework, we consider the same trip-planning scenario as above. Now in addition to the desired hotel accommodations and car reservations, Jane expresses a preference to fly on United, American or Delta in that order, her willingness to stay over a weekend if there is a significant fare saving and the wish to have the possibility to attend a music concert if the concert ticket prices are within a specified budget. These additional requirements are beyond the capabilities of a static workflow system and they point to four possible situations that need to be handled dynamically in a mission-critical fashion, hence leading to the establishment of a VE. First, given the flight preferences of Jane, the trip plan workflow activities at the travel agent need to be dynamically reorganized. Second, given Jane's music concert wish, the workflow need to be augmented with additional activities. Third, the travel agent may not be able to locally obtain any trip itinerary that suits the customer's preference list of budget limits, hotel chains, etc.. Fourth, the travel agent has no connections in the entertainment industry and cannot accommodate Jane's music concert requirements. In the last two cases, the travel agent needs to outsource these activities to other organizations (other travel agents, consolidators, etc.) potentially leading to further reorganization and expansion of the workflow. In our framework, whether outsourcing is for a personalized service or a longer term, it is automatically carried out with the help of adlets.

In our framework, for each workflow activity, an instance of an adlet is created with the necessary code and data and with credentials necessary for authentication to execute it. In our example, an adlet is created to perform the airline reservation. Such an adlet is equipped with the capability to interact with the airline reservation system and issue transactions to obtain available flights, seats and prices. It evaluates the obtained information against the preferences and budget specifications of the customer, selects the best match and issues a reservation pending final confirmation. Similarly, an adlet is created to perform a car reservation without outsourcing, i.e. requiring interaction with a new, external service provider.

However, the hotel reservation and the classical music concert booking in Vienna cannot be handled by local activities and an adlet is created to carry out the outsourcing. The first goal of such an adlet is to obtain relevant information about potential service providers from locally stored information and service advertisements in specialized databases, trading repositories, yellow pages etc. and match it with the travel agency's preferred type of service providers. (The advertisements of services and services providers' profiles are also performed by adlets.) Then one or more adlets may be created to explore all possible alternatives of service providers for Jane's hotel and concert reservations either separately or in conjunction as a two-node workflow view (see previous section).

An adlet is engaged in a two-phase interaction with a service provider. During the first phase, the adlet negotiates the service in accordance with the policies of the service requester. This first phase is completed with the establishment of a service *contract* that is validated by both service requester and service provider's *Authentication and Policy Manager (APM)*. We assume that every site has an APM that uses encryption to ensure that an adlet is only executed at the intended service provider during the second

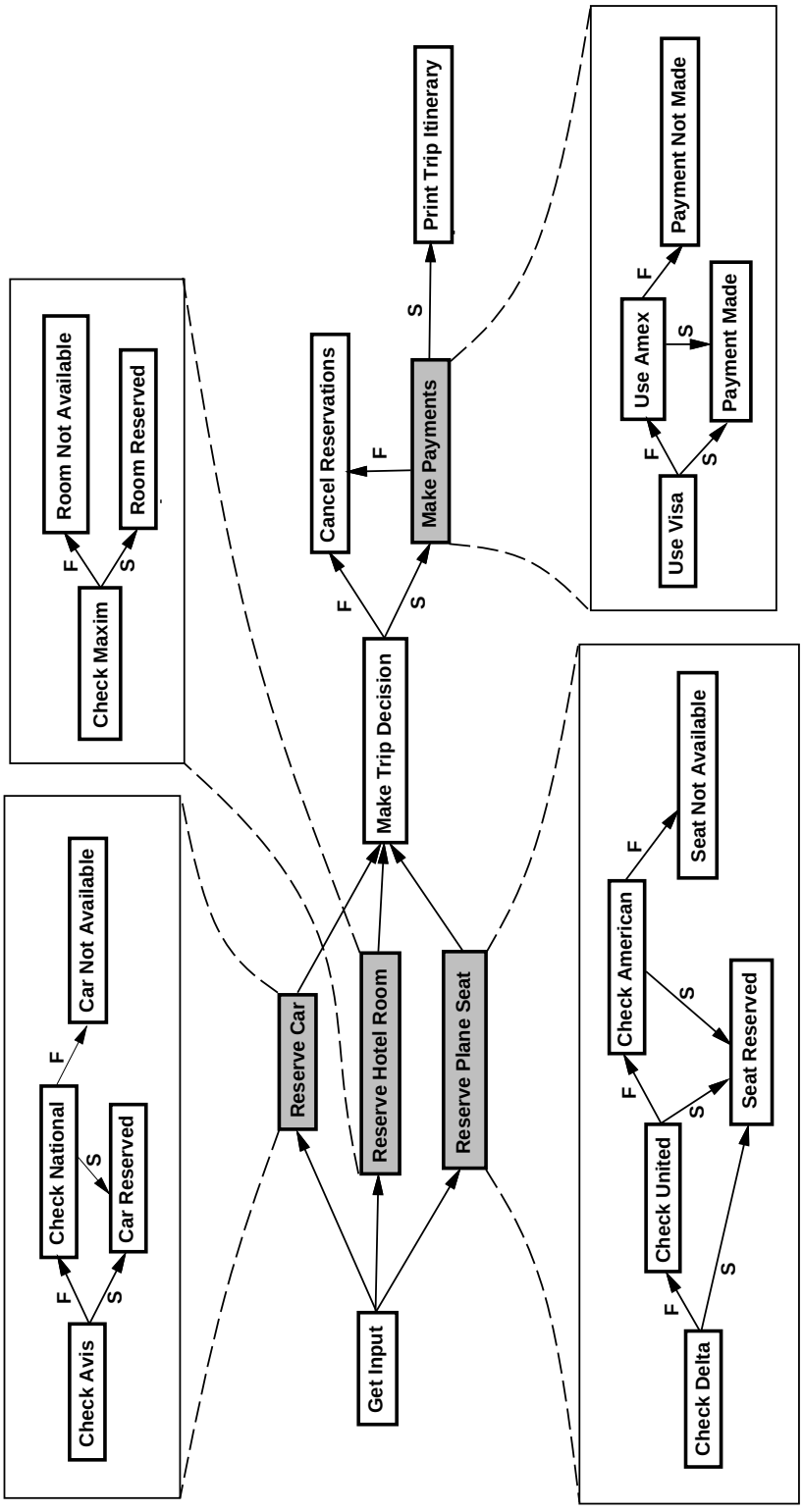

Figure 1. Trip Plan Workflow

phase during which the workflow activities are actually performed. During the second phase, the actual reservations proceed as described above, with the different adlets making tentative reservations, considering value-added options and coordinating among themselves to obtain the reservations that best match Jane's preferences. Further, adlets use the rules and semantics specified by the service requester to collect any potential information that may be relevant to future workflows involving traveling customers. When all potential service providers are visited, the best reservations are confirmed and the workflow is completed.

The above scenario illustrates the main features and capabilities of the adlet based workflow framework. In particular, it demonstrates:

- the ability of the framework to support dynamic execution of workflow steps,
- the ability of the workflow to create adlets to execute specific tasks asynchronously and remotely,
- the ability to enforce various level of security for local and remote protection, using potentially different security schemes and policies,
- the ability of the adlets to discover and accumulate new knowledge for potential use in future transactions, and
- the ability of the server to advertise its services.

4. Adlets: Mobile Workflow Agents

In the environment described so far, useful information and services necessary to achieve a business goal reside at various servers of different enterprises that need to be dynamically linked to form a VE. In our framework, the task of linking the sites and hence their relevant services forming a multi-organizational workflow is accomplished by mobile agents called *workflow adlets*. Adlets can be dynamically dispatched and can roam from one server to another based on their mission and acquire new knowledge from the visited servers. The information carried in the adlets may in turn spawn other adlets or terminate previously created adlets. In this section, we describe how adlets identify the most appropriate sites, glean the relevant information from the huge data repositories at these sites and employ the services available at these sites to complete the objectives of the dynamic workflow in a resource efficient and cost effective manner.

Each site in the environment can be both a service requester and a service provider and maintains a service or information repository (Figure 2). A service repository contains descriptions of (1) offered services by its site and (2) of other service providers and their services. We also assume the existence of trading serves which in addition to offered services, they store requested services in their repositories. Both provided and requested services are specified as workflow views. In accordance to their authentication, adlets initiated by a site or by another site may use, modify and delete service descriptions as well as insert new ones, for example, as a result of advertisement or the establishment of a new partnership.

To achieve its functionalities, an adlet is associated with a set of metadata that describe its mission and roaming and advertising strategies, its scope and authentication credentials. Depending on its type, the metadata elements may be specified by the user or derived from the application and possibly constructed as a set of conceptual relations. The user, for example, must restrict the roaming and advertising strategy to a specific domain. Similarly, the nature of the application may impose strict authentication and filtering policies to achieve stronger security and guard against potential intrusions and fake advertisement. Formally, an adlet can be defined as follows:

adlet = (mission, profile, target, non-target, strategy)

The $mission$ $(< type, \{ads\} >)$ specifies the type of adlet (service provider or service requester) and service advertisements ads. An ad is specified as a workflow view representing a service along with the associated set of preconditions and constraints. An adlet can dynamically change its mission by decomposing a workflow view into several smaller views and delegating some of them to another adlet or dropping them, or by expanding a workflow view incorporating opportunities, omitted activities and data. Whether or not an adlet is able to change its mission depends on the profile of the adlet, described next.

The *profile* specifies the behavior of an adlet. First it describes its authentication credentials and identify its site and application of origination. Second it describes its itinerary and rules for identifying servers and for modifying its itinerary. These rules also include termination criteria. Third, it describes how the workflow views in the mission are manipulated. In the case of a service provider, these rules include dissemination criteria whereas in the case of service requester, these rules include acquisition criteria. Fourth, it describes cost comparison strategies to aid negotiation and rules that define negotiation requirements and contract establishment. For example, negotiation requirements include the minimum security levels expected from any sites that are part of any negotiation. A cost function may have value-adding and value reducing properties. For example, given a workflow view representing a service, the involved activities can be classified as *vital* or *non-vital*. The projection on the vital activities results in a vital workflow (sub)view that defines the essential service to be provided (or requested). A contract can only be established if an essential service can be agreed between a requester

and one or more providers. That is, for example, a vital view in an adlet cannot be negotiated and must be matched exactly to a service available at a site or decomposed into smaller views that are matched at different sites. In general, multiple sites may provide a match, but the contract is established with the service provider that is able to satisfy as many of the non-vital views possible and which augments the vital view with as little as possible non-vital and expensive views. Thus, non-vital views may act as incentives for service providers to win a contract.

The parameter *target* and *non-target* specify the scope and the focus of the adlet's itinerary, while the non-target specifies what is out of the scope. The scope of the itinerary may be *global, restricted or local*. A global scope extends over the entire network. A restricted scope involves a selected set of domains within which the adlet is allowed to roam. A local scope confines the adlet to within the boundaries of the local network. Within a target the **focus** of the roaming adlet may be *inclusive* or *selective*. A select focus allows the adlet to interact with all servers within the target, while a selective focus restricts the adlet to a specific set of servers characterized by a set of specific properties as described in the profile of the adlet. Further, the restricted scope could be specified as the number of levels of sources to traverse beyond the primary sources. The *non-target* parameter is a list of specific sites within the target domain not to visit. Both the $target$ and *non-target* parameters may be revised to expand or limit the current scope further. In fact, one of the adlet missions may be to update the scope as more up-to-date information is obtained.

Adlets can be categorized into two broad types based on the $strategy$ parameter. These are *pro-active* versus *reactive* adlets. Pro-active adlets conduct groundwork in anticipation of the requirements of workflows that will be invoked in the future. For instance these adlets could conduct a systematic search for secondary and tertiary sources or service providers. These information sources/service providers could be of different types (e.g., car rental agencies, hotel chains, concert ticket offices, etc.). Note that pro-active adlets have the potential of providing fast workflow execution if appropriate groundwork has been completed before the invocation of the workflow. Further, the pro-active adlets may be programmed to be automatically re-launch their actions periodically to obtain up-to-date information about service availability, and log changes in the environment. The periodicity of these actions depend on the availability of computing and networking resources. Reactive adlets on the other hand, are invoked only when a specific workflow is initiated. At any given time, the system could have a mix of reactive and pro-active adlets depending on the availability of resources.

Adlets may be created dynamically by a workflow and invoked, possibly repeatedly, during the execution of the workflow. In the following section, we describe the overall system architecture to support the adlet-based framework.

4.1. Adlet System Architecture

The system architecture to support adlet-based dynamic workflows should address the basic issues of how to organize the resource space flexibly and dynamically to make the implementation of a large-scaled infrastructure feasible. The architecture must:

- allow the system's structure and interactions between adlets and servers to evolve over time in accordance with usage patterns, service advertising and adlet roaming strategies,
- capture the interactions between adlets and servers and promote service advertising, filtering, and information archiving in a user and context sensitive subgroupings so as to reduce the amount of overhead required to locate a server and limit the interaction to particular groups of servers which are most likely to assist effectively in the execution of the workflow. Capabilities must be provided to allow the adlets to plan their roaming itinerary in the most effective way by continuously seeking to construct virtual links among themselves and potential servers based on the metadata that describe the content and type of their mission, the advertising and recruiting strategies specified by the profile, and the context within which the workflow task is being undertaken,
- support negotiation between adlets and servers so that their collaboration is achieved in the most effective way.

To achieve these objectives, the proposed architecture naturally draws upon existing components of an internetwork, including name servers, authentication servers and routers, and adds new components, namely *Adlet Managers*, *Adlet Credential Authenticators*, and *Adlet Communication Servers*.

Adlet Managers, at different networks, cooperate to provide the basic mechanisms and capabilities to support the interaction between servers and adlets. Adlet Authenticators verify and validate the credentials of the adlets, such as the identity of the adlet originator, the network address where the originator resides and possibly the name and address of author authorities sanctioning entities. The Adlet Communication Server handles communication and group formation among adlets and servers to facilitate their cooperation toward efficiently executing the steps of the workflow. The overall structure and functional level decomposition of the proposed architecture is depicted in Figure 2. In

the following sections, we briefly discuss the functionalities and services provided by each of these components.

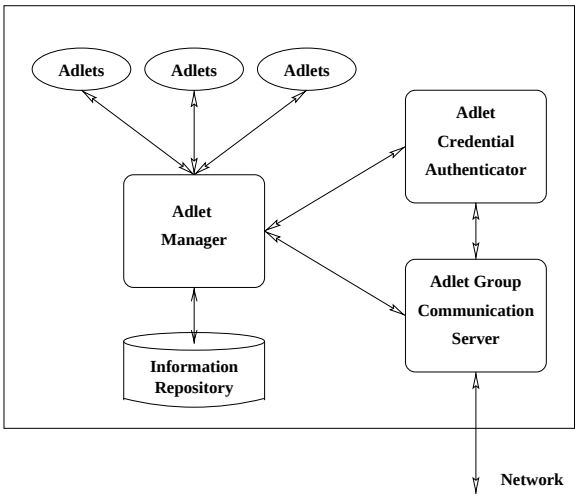

Figure 2. System architecture to support active document advertising.

4.1.1 Adlet Manager

The main purpose of an adlet manager is to facilitate the establishment of logical structures to support collaborative sessions between adlets and servers. Adlet managers form a multicast group and communicate on a common information channel to advertise the profiles of their associated servers and assist adlets in identifying suitable servers for the execution of a task.

When an adlet manager receives an adlet advertising the service profile of a server, it first authenticates the adlet and its associated server. The adlet manager then extracts the server's profile information in order to determines the "usefulness" of the advertised service based on the profiles of the local adlets and the recent execution contexts of workflows. If it determines that the advertised information is potentially useful for the execution of current or future workflows, the adlet manager summarizes the information in various type-specific ways to generate structured indexing information that can be used to help workflow adlets identify suitable remote servers that can assist in the execution of the workflow.

Similarly, when a workflow step requires the assistance of a remote server or the retrieval of new information, an adlet is created and sent to the the adlet manager. When the adlet arrives, the adlet manager, first extracts the metadata and profile information associated with the adlet. It then performs similarity tests, using the query specified in the adlet metadata and the advertised service profile, to determine if the service sought by the adlet matches profiles previously advertised by remote servers. If such a match exists, the adlet is provided with a list of potentially suitable servers. This list is used to select a set of servers that meet the criteria specified by the underlying application.

Armed with its credentials and the minimal information necessary to accomplish its task, the adlet first resolves the names of these servers into their network addresses and proceed to visit each server to negotiate their cooperation in completing the workflow task. The decision of the adlet to visit all the servers in the list depends on the underlying strategy as specified in the metadata. If the adlet is seeking "any server" which can provide the required service, the planned itinerary may be interrupted immediately after such a server is discovered.

On the other hand, if the adlet manager determines that no match exists between the adlet query and currently available service profiles, or if the number of selected servers does not meet the minimum number of servers required by the underlying application, the adlet initiates a procedure to seek new servers. To achieve this the adlet progressively visits the neighboring adlet managers, the closest neighbor first and the neighbors of the neighbors, etc. During this trip, the adlet itself may carry advertising information about its local servers to neighboring adlet managers.

4.1.2 Adlet Group Communication Server

Depending on the underlying application, the interaction between adlets and servers may require provision for multicast capabilities and efficient management of multicast group addresses. These groups are short lived and usually last for the duration of the workflow execution. Consequently, an efficient scheme to manage dynamic allocation of multicast addresses in the adlet framework is an important design issue that need to be addressed. While the Internet protocol suite provides a number of proposals for multicast protocols, (e.g., CBT [2] and MOSPF [12]), but these do not address the issue of dynamic allocation of group addresses. The main objective of the adlet group communication server is the dynamic allocation of multicast addresses in support of group communication among adlets, adlet managers and servers.

5. Preliminary Implementation Plan

We are planning on building a prototype of a dynamic workflow system that integrates workflows with mobile agents implementing our notion of adlets. As a first step, we plan on using a mobile agent technology called *aglets* (agile applets) developed by IBM Japan [11] that secure communication.

An *Aglet* (agile applet) is a lightweight mobile Java object (approximately 2 kilobytes). If the aglets are equipped with database capabilities and Java Database Connectivity (JDBC) Drivers, they become *DBMS-aglets*, that can connect to a remote data server and collect data and performed updates [13]. An aglet carries along its program code, state, unique identification, and query trip plan as it moves from host to host. If there are any communication problems, such as a host failure, the trip plan allows the aglet to try alternate hosts and solutions. A host interacts with an aglet by utilizing an aglet server program. This Java program listens for incoming aglets and provides a context in which they can resume their suspended execution. Aglets can be created, talk to one another, move from host to host, and be disposed of at any time.

Our preliminary implementation plans consist of developing the adlet prototype over aglets, incorporating the advertising strategies and dynamic selection of workflow segments and adding efficiency-enhancing features such as multicasting. This will prove to be relatively easier since several features of adlets (such as migration between hosts) are already available in the aglet technology. Furthermore we can concentrate our efforts on the interactions between the workflow system and adlets.

6. Conclusions and Future Work

In this paper, we have described a framework to enable the efficient operation of a virtual enterprise (VE) composed of multiple distributed workflows. The major goal of this paper was to develop a framework that enables the automatic establishment of a VE and maximizes the sharing of useful information between the various components of the VE without human intervention as done in previous strategies. The core of our approach is based on mobile agents referred to as adlets. Adlets are dynamically created on a need-to basis and roam a specified relevant area of the workspace to gather new information and facilitate connections between related workflows.

Our work is related to the work on adaptive, evolving and migrating workflows (e.g., [9, 3, 4]). In particular, migrating workflows [4] share the same underlying idea with our work, treating a static workflow specification as incomplete and attempt to dynamically expand it by migrating the execution of the workflow from one service site to another until it achieves its goal. They also suggest the use of mobile agents as a potential implementation infrastructure.

Currently, we are investigating different negotiation strategies and contract establishment methods as part of the refinement of our framework. Some implementation of an adlet prototype has already begun. Future work is planned on implementing a prototype of the dynamic workflow environment that interacts with the adlet component.

References

[1] Alonso G., D. Agrawal, A. El Abbadi and C. Mohan. Functionalities and Limitations of Current Workflow Systems. *IEEE Expert*, 12(5), 1997.

[2] Ballardie T., P. Francis and J. Crowcroft. Core Based Trees (CBT): An Architecture for Scalable Interdomain Multicast Routing. *Proc. of the ACM SIGCOMM Symp.*, pp. 85–95, 1993.

[3] Casati F., S. Ceri, B. Pernici and G. Pozzi. Workflow Evolution, *Data & Knowledge Engineering*, 24(3): 211-238, 1998.

[4] Cichocki A. and M. Rusinkiewicz. Migrating Workflows. *Workflow Management Systems and Interoperability*, A.Dogac et al. (Eds) Springer Verlag, Series F, Vol 164, pp. 339–355, 1998.

[5] Chrysanthis P. K.. Guest Editor's Introduction to Special Issue on Workflow Systems. *Distributed Systems Engineering*, 3(4):211–212, 1996.

[6] Chrysanthis P. K. and K. Ramamritham. Synthesis of Extended Transaction Models using ACTA. *ACM Trans. on Database Systems*, 19(3):450–491, 1994.

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

[8] Georgakopoulos D., H. Sinha, K. Huff and B. Hurwitz. "Monitoring Multi-organizational Processes." *Proc. of the 11th Int'l Conf. on Parallel and Distributed Computing Systems*, pp. 75–80, 1998.

[9] Han Y. and A. Sheth. On Adaptive Workflow Modeling. *Proc. of the 4th Int'l Conf. on Information Systems Analysis and Synthesis*, pp. 108-116, 1998

[10] Jablonski S. et al. "External and Internal Support Services in Workflow Management Systems." *Proc. of the 11th Int'l Conf. on Parallel and Distributed Computing Systems*, pp. 81–86, 1998.

[11] Lange D. and Oshima M. *Programming and Deploying Java Mobile Agents with Aglets*, Addison Wesley, 1998

[12] Moy J.. Multicast Routing Extensions to OSPF. *Communications of the ACM*, Aug. 1994.

[13] Papastavrou S., E. Pitoura, and G. Samaras. Mobile Agents for WWW Distribued Database Access. *Proc. of the 15th Int'l Conf. on Data Engg.*, 1999.

[14] Ramamritham K. and P. K. Chrysanthis. *Advances in Concurrency Control and Transaction Processing*, IEEE Computer Society Press, 1997.

[15] Workflow Management Coalition, Technology & Glossary, Document Number WFMC-TC-1011, June 1996.