An Adaptable Workflow System Architecture on the Internet for Electronic Commerce Applications [*]

Ibrahim Cingil, Asuman Dogac, Nesime Tatbul, Sena Arpinar
Software Research and Development Center
Faculty of Engineering
Middle East Technical University (METU)
06531 Ankara Turkiye
asuman@srdc.metu.edu.tr

Abstract

An electronic commerce (EC) process is a business process and defining it as a workflow provides all the advantages that come with this technology. Yet electronic commerce processes place certain demands on the workflow technology like the distribution of the load of the workflow engine to multiple servers, dynamic modification of workflows for adaptability, openness and availability.

In this paper we propose a workflow system architecture to address these issues. The componentwise architecture of the system makes it possible to incorporate the functionality and thus the complexity only when it is actually needed. The infrastructure of the system is based on CORBA 2.0 where methods are invoked through XML. The clients of the system are coded as network transportable applets written in Java so that the end user can activate workflow components through the workflow domain manager over the network. The system provides high availability by replicating the component server repository and the workflow domain manager. We also discuss how this architecture can be used in building an electronic marketplace.

1. Introduction

There are many business models used in electronic commerce (EC) like e-shop, e-procurement, e-mall, electronic marketplace, virtual communities, value chain service providers, value chain integrators, collaboration platforms, and information brokerage [15]. In all of these models the business processes can be modeled as a set of steps that are ordered according to the control and data flow dependencies among them. This corresponds to a workflow process, where the coordination, control and communication of activities are automated, although the activities themselves can either be automated or performed by humans.

Yet the workflow systems to be used in electronic commerce should have specific features that are of critical importance for electronic commerce applications [11]:

- The mass-business characteristics of EC workflows requires a high-throughput workflow execution engine. Thus, load distribution across multiple workflow servers is necessary to ensure this kind of scalability. The EC workflow systems must also quickly adapt to network changes due to failed sites or due to load balancing.
- The EC workflows should easily adapt to different and changing requirements of the customers. So a workflow model is more likely to be a template that is dynamically enriched by introducing additional activities along with their control and data flow, and also possibly skipping the parts of the pre-specified workflow template. Also there could be changes in EC process execution flow triggered by collaborative decision points, or context-sensitive information updates or other internal or external events which necessitate dynamic modification of the workflow instance (e.g., cancelation of an order by the customer).
- Electronic commerce processes should be ubiquitous. To achieve this they should be able to run in environments with scarce resources, and they should also have an open architecture. That is the functionality of a workflow system should be tailorable to the needs

[*]This work is partially being supported by Middle East Technical University, the Graduate School of Natural and Applied Sciences, Project Number: AFP-97-07.02.08, by the Scientific and Technical Research Council of Turkey, Project Number: 197E038 and by the European Commission Project Number: INCO-DC 97-2496 MARIFlow.

and available resources of a customer and the system should run on the Internet and should be based on an open and interoperable infrastructure.

- Frequent failures and unavailability of EC workflow servers would immediately weaken the market position of the merchant. This requires efficient replication of workflow servers.

The current monolithic workflow engines on the other hand can not fulfill these needs. In the current systems even for very simple workflow processes, the full scale engine runs thus consuming resources unnecessarily. These systems also suffer from difficulty in load balancing, migration of workflow process instances, and efficient replication of servers.

We have designed a workflow system architecture based on Internet and CORBA [13, 14] to address these issues:

- Each process instance is a CORBA object that contains all the necessary data and control information as well as its execution history. This feature makes it possible to migrate the object in the network to provide load balancing. Furthermore it is possible to dynamically modify the process definition on the instance basis at run time. It should be noted that with this architecture, a site failure affects only the process instances running at that site.

- The system is designed to consist of functional components containing but not restricted to: Basic Enactment Service, User Worklist Manager, Workflow Monitoring Tool, Workflow History Manager, Dynamic Modification Tool, Process Definition Library Manager, Reliable Message Queue Manager, Workflow Domain Manager and Distributed Transaction Manager. This componentwise architecture makes it possible to incorporate the functionality and thus the complexity only when it is actually needed at run time by a process instance by downloading only the necessary components which results in effective usage of system and network resources. It is also possible to add new components or maintain and upgrade the existing components of the system incrementally without affecting the other parts of the system.

- The componentwise architecture facilitates the replication to a great extend. Each site can download its own copy of a component server; also the Workflow Domain Manager can be replicated at each site as a Site Manager. This provides for availability and prevents network overhead.

- The clients of the system are coded as network-transportable applets written in Java so that the end user can acquire workflow components from the Workflow Domain Manager over the network. Thus it is not necessary to have the software pre-installed on the user machine. This promotes user mobility further as well as easy maintenance of the system components which can be upgraded transparently on the server side.

- The architecture proposed in this work uses CORBA as the distribution infrastructure. We have chosen to invoke methods in CORBA through XML [16]. In this way a CORBA compliant ORB is not necessary at the client side. The client can send the method invocation requests and its parameters expressed in XML through the ”post” method of the HTTP protocol. An HTTP server receiving this request can pass it to a CORBA server through CGI. CORBA server will invoke the method after parsing the XML document. If there is an ORB at the client side, the client sends the request directly to the CORBA server again in XML. In this way it also possible to replace the distribution infrastructure by a Web-native ORB in the future when the Web-native ORBs provide CORBA-like capabilities. All data exchanges are realized through Extensible Markup Language (XML) providing uniformity, simplicity and a highly open and interoperable architecture.

The paper is organized as follows: Section 2 describes our adaptable workflow system architecture and its advantages. In Section 3, an example electronic commerce application is given which uses the proposed workflow system architecture to realize an electronic marketplace. Related work is summarized in section 4. Finally, conclusions are given in Section 5.

2. Workflow system architecture

2.1. The architecture

The general system architecture is presented in Figure 1. In the following the basic components of the system are presented:

1. Component-Server Repository: The components of the system are implemented as CORBA objects that are invoked through Java applets. The Java applets are downloaded to the client machine when a user through a Web browser accesses the Workflow Domain Manager and asks for a specific service. Thereon the Java applets interact with the user and direct the user requests to the appropriate CORBA objects. The Component-Server Repository contains the following components:

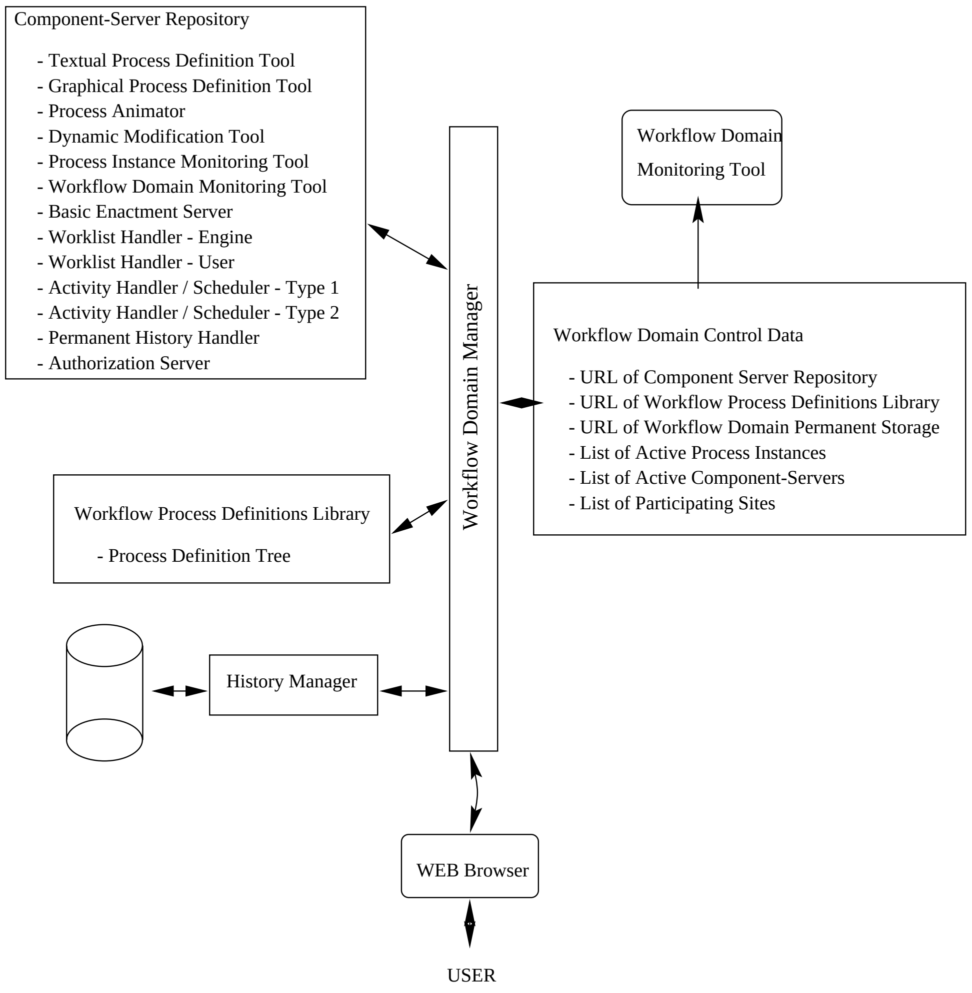
Component-Server Repository
- Textual Process Definition Tool
- Graphical Process Definition Tool
- Process Animator
- Dynamic Modification Tool
- Process Instance Monitoring Tool
- Workflow Domain Monitoring Tool
- Basic Enactment Server
- Worklist Handler - Engine
- Worklist Handler - User
- Activity Handler / Scheduler - Type 1
- Activity Handler / Scheduler - Type 2
- Permanent History Handler
- Authorization Server
Workflow Domain Monitoring Tool
Workflow Domain Manager
Workflow Domain Control Data
- URL of Component Server Repository
- URL of Workflow Process Definitions Library
- URL of Workflow Domain Permanent Storage
- List of Active Process Instances
- List of Active Component-Servers
- List of Participating Sites
Workflow Process Definitions Library
- Process Definition Tree
History Manager
WEB Browser
USER

Figure 1. The architecture of the proposed workflow system.

- Workflow Process Definition Tool: Authorized users are allowed to define new workflow processes. Graphical or textual specification interfaces can be implemented. The process definition is syntactically verified and permanently stored in the Workflow Process Definition Library.
- Dynamic Modification Tool: Authorized users are allowed to modify previously defined workflow processes that are stored in the Workflow Process Definition Library. It should be noted that template modification does not affect the already active instances of a workflow. Authorized users are also allowed to modify a particular workflow process instance to respond to external changes that cause variations in the prespecifed process definition. The modifications can be applied to executing instances selectively or to all instances of the same workflow process if required. The modifications can also be reflected to the template as well [9].
- Process Instance Monitoring Tool: Users are allowed to trace workflow process instances they have initiated and extract run-time information about the current execution status of an instance. Collecting and measuring process enactment data are needed to improve subsequent process enactment iterations as well as documenting what process actions actually occurred in what order. This feature provides data to improve optimization and evaluation of processes.
- Process Animator: Graphically simulating the reenactment of a process is needed to more readily observe process state transitions or to intuitively detect possible process enactment anomalies. Process visualization provides users with

graphical views of process templates or instances that can be viewed, navigationally traversed, and interactively edited and animated to convey process statistics and dynamics. Process visualizations enable intuitive analysis and discovery as well as being a key to user acceptability of the technology. Visualizing and replaying process enactment histories are well-received and support organizational drill-down when process anomalies are observed.

2. Workflow Process Definitions Library: Workflow definitions (i.e., the process templates), organizational role definitions, and participant-role assignments are durably stored in this library. Only workflow specification tool and workflow process template modification tool insert or update workflow process templates in this library. Accesses by all the other services are read-only. This library is maintained by the WFMS library manager.

   Different workflow schema versions have to be managed and different propagation strategies of workflow schema changes to their workflow instances have to be provided by a WFMS in order to flexibly support the migration from one business process to an improved one, to support alternative workflows for process variants, and to support adhoc changes of a workflow. When the workflow definition is modified permanently, the versions of workflow definitions are stored, since:

   - In some cases, it may be necessary to recover to the old workflow definition. For example, when it is observed that the new definition performs worse than the old definition.
   - It may be desired that more than one version of definitions are active at the same time. That is, some instances are created from one version, and some others from a different version of the definition. For example, a bank wants to change its *giveCredit* workflow process by adding more activities to search the financial backgrounds of the customers. However, for their well-known customers, they may still want to apply their old *giveCredit* process. By keeping versions of definitions, both process instances can be created.

   In the system, to handle the versioning of definitions, a *definition tree* is kept to provide the administrator the flexibility of modifying a definition several times. During the modification, the administrator selects one version, default being the last one. Thus new instances are created from the default definition, if the version number of workflow definition is not identified explicitly during the instance creation.

3. History Manager: The History Manager handles the database that stores the information about workflow process instances which have been enacted to completion to provide history related information to its clients (e.g. for data mining purposes). It should be noted that the history of active process instances are stored in the process instance object.

4. Workflow Domain Manager: The domain manager is the Web server of the system. All clients access to the domain manager via their Web browsers and in response to their authorized service requests the domain manager downloads appropriate Java applets to the client which then handle subsequent requests of the same client for that particular service which is provided by a component server. If the client needs a different WFMS service, the domain manager is then accessed again via the Web browser and another Java applet is downloaded. The domain manager keeps runtime information such as list of active process instances, active component servers, list of participating sites, etc. for domain monitoring purposes.

5. Workflow Process Instance Object (WPIO): When a user wants to initiate a new instance of a prespecified workflow, the Domain Manager creates a new Workflow Process Instance Object (WPIO). The main method of this object is Basic Enactment Method (BES) which is activated by the Domain Manager on behalf of the client. The Workflow Process Instance Object (WPIO) contains all the data (such as workflow process definition, workflow relevant data, enactment history of that instance up to the current execution status, etc.) required to complete the execution of the process instance. When a WPIO completes its execution it is destroyed after being durably stored. That is, WPIOs exist only for active process instances. A WPIO contains all the run-time information about its own process instance which might be needed to migrate the process instance from one site to another or to rescue an instance in case of failures. Since a WPIO contains the workflow process definition and all the run-time information about its own process instance, the dynamic modification of the workflow process definition contained in that WPIO is simply enabled by dynamically modifying the WPIO. The dynamic modification of a WPIO may be initiated in two ways: either by a user or by means of a special activity specified in the process definition as explained in the following:

   - A user via his Web browser may access the

Workflow Domain Manager and download the Dynamic Modification tool. The user is supposed to provide the WPIO-identifier which otherwise may be obtained by the Dynamic Modification tool from the List of Active Process Instances contained in the Workflow Domain Control Data. Once the WPIO to be modified is identified enactment service of the WPIO is paused by sending a proper message to its BES and from thereon the Dynamic Modification tool has control of the WPIO. The user modifies the workflow process definition contained in the WPIO as s/he wishes. If the modification requires undoing some activities that have been already executed, corresponding compensation activities are included as part of the modification such that when the enactment service resumes, those already executed activities are compensated first. After the user indicates explicitly that the modification is completed the enactment service of the WPIO resumes by sending another message to its BES. The user has at his will the choice of applying this modification to the other instances of the same workflow process or even to update the process template stored in the Workflow Process Definitions Library.

- Workflow process definition contains a special activity called Workflow Process Modification Activity (WPMA) that when executed automatically invokes the Dynamic Modification tool on behalf of a user so that the user can modify the WPIO. The WPMA handles instance-specific differences of the process definition when necessary. Each specification of the WPMA activity results in a separate modification of the WPIO. A WPMA initiated modification may not affect other instances of the same workflow process or the Workflow Process Definitions Library.
- Dynamic Workflows which have no pre-specified process definition are handled with another special activity called Dynamic Workflow Special Activity (DWSA) that automatically invokes the Dynamic Modification tool on behalf of a user so that the user can specify the next activity to be executed. A dynamic workflow process definition initially includes only one activity: the DWSA. When this process is initiated, the DWSA invokes the Dynamic Modification tool and awaits the user to specify activities to be executed. When the user specifies the next activity or activities, another DWSA is appended automatically such that after the user-specified next activity(s) is executed, the DWSA will be invoked again. The DWSA will not be appended if the user explicitly indicates that no more activities are to be specified in which case the termination of DWSA will indicate the termination of the WPIO. In this way a workflow process can interactively be defined on-the-fly by a user and it is saved in the Workflow Process Definition Library if the user specifies so when terminating DWSA.

2.2. Advantages of the proposed architecture

The run-time system despite having a central control on a process instance basis brings out all the benefits of highly distributed environments. Each WPIO instance of a workflow process may execute at a different site. The Component-Server Repository, Workflow Definition Library, Workflow Domain Control Data and the Workflow Domain Manager may all be replicated for better performance and availability. Each participating site may have its own replication of Workflow Domain Manager as the Site Manager. Since no prior installation of any WFMS software is required on the client side, the system is highly dynamic and thus any component-server implementation may be upgraded at the server side without necessitating any changes on the client side. In addition a site failure can be handled simply by migrating the WPIOs to be executed on that site to another site/other sites.

3. An example electronic commerce application

In this section, an example application describing the use of the proposed workflow system architecture in electronic commerce is presented. An electronic marketplace architecture is designed as an application. In this marketplace, electronic commerce is realized through the adaptable workflow templates provided by the marketplace to its users.

3.1. Requirements of electronic marketplaces

In addition to the general requirements of electronic commerce discussed in Section 1, electronic marketplaces also have some specific requirements to be met. The marketplace is designed with the following issues under consideration:

1. Electronic commerce processes in the marketplace are modeled as workflow processes.

   Many researchers as well as commercial companies have created systems that support various aspects of electronic commerce such as online shopping, virtual catalogs or electronic marketplaces. While these systems provide interesting shopping experiences, they fall short in fully exploiting the capabilities offered by the electronic medium [2].

These systems can not handle diversity of customer needs. As an example, a customer may want to buy more than one related item and there can be dependencies among the items and also compatibility requirements that stem from the nature of these items. For example, s/he may want to buy a printer together with a personal computer. This creates a dependency between the computer and the printer. Also, for the specific software that s/he considers there could be a certain amount of memory requirement. This illustrates a compatibility requirement. In contrast, current systems are mostly designed to handle one request at a time. For instance, a customer may buy an item by searching several shops/stores but can not make several inquiries in one step to buy several compatible and related items and/or services. Shopping carts support several inquiries of a buyer however the buyer cannot give dependencies among the items or the specific order of the related purchases. Also, no support is provided for compatibility requirements. In our approach, the dependencies expressed by the user are represented through control flow dependencies. For expressing the compatibility requirements among items, there is a need for a knowledge base to store the rules.

More importantly, most of the systems developed do not have enough facilities to automate the business processes conducted by the user/customer. In this respect, we propose to organize the electronic commerce processes into workflow templates adaptable to user needs. Workflow based approach allows involved parties to define their own tasks and to invoke already existing applications within the workflow and to restructure the control and data-flow among the tasks, in other words, to automatically create a custom built workflow from the workflow template. The higher level of abstraction provided by the workflow technology makes this customization of processes for different users possible. In addition, the workflow definition makes it possible to invoke any number of activities in parallel to provide efficiency and to dynamically reengineer the commerce processes not only to the user needs but also to balance the system workload. For example, the search and purchase of related items from different stores can be activated in parallel. Furthermore the recovery functionality of a workflow system allows to automatically rollback the necessary activities if a dependent activity fails, eg., a desk purchase request of a user executing in parallel with his computer purchase request can be automatically rolled back if the computer purchase request fails.

2. The communication infrastructure of the underlying workflow system is CORBA. All data exchanges are realized through XML (Extensible Markup Language).

   For electronic commerce to become really ubiquitous, electronic commerce architectures should be open, that is, they should be based on infrastructures providing for semantic interoperability. The most promising proposal in providing an open and interoperable electronic commerce architecture seems to be the efforts of World Wide Web Consortium (W3C) in providing data exchange and data semantic standards like XML, RDF (Resource Description Framework) and CommerceNet's efforts on developing an open electronic commerce framework based on Common Business Library (CBL).

   The Extensible Markup Language (XML) is a data format for structured document interchange on the Web. It provides a framework for tagging structured data by allowing developers to define an unlimited set of tags bringing great flexibility. XML resembles and complements HTML. XML describes data such as city name, temperature, and HTML defines tags that describe how the data should be displayed such as with a bulleted list or a table. Document Type Definitions (DTDs) may accompany an XML document, essentially defining the rules of document, such as which elements are present and the structural relationships between the elements. DTDs help to validate the data. XML brings so much power and flexibility to Web-based applications that it provides a number of benefits to developers such as being able to do more meaningful searches.

   XML has gained a great momentum and is emerging as the standard for self-describing data exchange on the Internet. Its power lies in its extensibility and ubiquity. Anyone can invent new tags for particular subject areas and they define what they mean in document type definitions. Content oriented tagging enables a computer to understand the meaning of data. But if every business uses its own XML definition for describing its data, it is not possible to achieve interoperability. The tags need to be semantically consistent across merchant boundaries. For this reason, CBL which consists of product taxonomies and message formats as XML DTDs, has been developed [10] and the baseline version (1.1) is available from [17]. CBL contains a set of building blocks common to many business domains such as address (location.dtd), price (value.dtd), purchase order (order.dtd) and standard measurements (measures.dtd), and thus provides the much needed basis to ensure interoperability among XML applications. When this is complemented by a set of DTDs common for specific industries, that is for vertical domains, the open electronic commerce infrastructure will be achieved.

3.2. The electronic marketplace architecture

Having explained our workflow management system architecture, we now discuss how this architecture can be used in building an electronic marketplace where sellers and buyers meet and negotiate to make the best deal. Figure 2 shows the general outlook of the marketplace architecture. In our marketplace, electronic commerce processes are realized through the adaptable workflow templates provided by the marketplace to its users. The templates contain agents (both buying and selling) as well as existing applications invoked by the workflows and are stored in the Intelligent Directory Service (IDS) of the marketplace. IDS constitutes the core of the marketplace architecture and provides document type definitions (DTDs), a dictionary of synonyms, repositories for agents, workflow templates, a knowledge base (KB) for item compatibilities and some library modules for agents' use. IDS also contains a match making mechanism for agents to find out about each other and the related Document Type Definitions (DTDs) to eliminate the need for an ontology in agent communication.

When a customer wants to buy a service or an item from the marketplace, s/he contacts a pre-specified URL, and an applet is downloaded which presents a form to be filled in. The user fills in the fields (customer info and the name of the item). This in turn activates a workflow template on the server. The first activity invoked registers the buyer to the IDS. The next activity checks the related DTDs in IDS with the item name specified by the buyer. Since the buyer may not know the right term (used in DTD) to use for the item, an intelligent dictionary of synonyms is used. For example, consider a computer shop using a computer DTD in describing its service. If a customer wants to buy a CPU and uses the term ”Processor” and if ”CPU” is the term used in DTD, then dictionary of synonyms is used to match the word ”Processor” with ”CPU”. The names and types of the properties obtained from the related DTD by the activity are passed to the buying agent activated. Obtaining the names and types of properties from DTDs is necessary since the buyer may not know in advance all the properties of the item.

The buying agent which presents the user a form containing the values or ranges for the properties of the item along with the criteria that the customer wishes to be optimized in the negotiation phase and the required parameters. Note that the choices made by the user on this form affects how the rest of the workflow process is to be formed. The buying agent negotiates with the related selling agents to realize the deal.

When a seller wants to sell some service or an item through the marketplace, s/he contacts a pre-specified URL, and an applet is downloaded which presents a form to be filled in. The form asks for the type of the service given and under what conditions. When the form is submitted, a workflow template is activated on the server. The first activity of this workflow registers the seller to the marketplace directory as in the buyer's case.

Note that the workflow templates (buyer and seller workflow templates) stored in the marketplace directory only contain the basic steps of the commerce process like registering to the marketplace, matchmaking, iniating a buying/selling agent to realize the actual negotiation step. These templates can be modified at run time according to the specific requirements of the users. This modification phase is realized through the dynamic modification tool of the underlying workflow system. This facility provides the buyer the opportunity to buy more than one, related items with dependencies among them. The item dependencies are embedded into the buyer workflow template. The compatibility constraints can also be checked from the IDS at this step.

4. Related work

In [5, 4], we present some initial ideas on a workflow based electronic marketplace on the Web.

In [7, 8], we present a workflow system, namely MARIFlow, through cooperating agents for document flow over the Internet. In that work we describe an architecture that provides for automating and monitoring the flow of control and data over the Internet among different organizations, thereby creating a platform necessary to describe higher order processes involving several organizations and companies. A workflow process is executed through cooperating agents, called MARCAs (MARIFlow Cooperating Agents) that are automatically initialized at each site that the process executes. MARCAs handle the activities at their site, provide for coordination with other MARCAs in the system by routing the documents in electronic form according to the process description, keeping track of process information, and providing for the security and authentication of documents as well as comprehensive monitoring facilities. Although the application domain of the MARIFlow system is maritime industry, the workflow system architecture is general enough to be applied to any business practice. While developing the architecure proposed in this paper, we have made use of some of the ideas given in [7, 8].

The current marketplaces can be investigated in the following categories:

- In product brokering marketing systems the users determine or identify the product that they want to

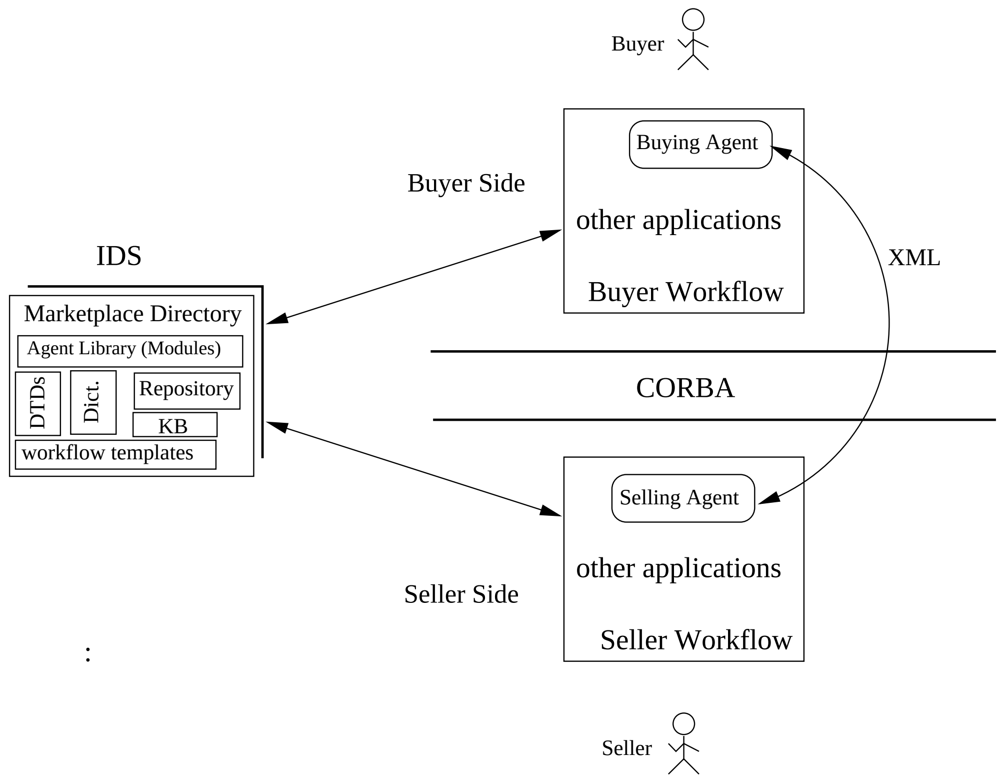

Figure 2. The general architecture of the electronic marketplace.

buy. PersonaLogic (www.personalogic.com) guides the user through selection among various products by presenting a large feature space. The system filters out unwanted products according to the user-specified constraints yielding the most appropriate ones to the consumer. FireFly (www.firefly.com) uses the opinions of like minded people to offer recommendations of such commodity products as music and books, as well as more difficult to characterize products such as Web pages and restaurants.

- Another category is where the user selects the merchant to buy from, according to merchant value-added services like price, delivery time, warranty etc. Andersen Consulting's BargainFinder (bf.cstar.ac.com), the first intelligent agent to provide comparison shopping, allows users to type in the name of a CD or music group and then search on-line stores for the lowest prices available. A more advanced system, Excite's Jango (www.jango.com) has more product features and shopping categories to search across however other issues for selection like delivery time and warranty are still amiss. The MIT Media Lab's Kasbah (kasbah.media.mit.edu) is a multi-agent system where the user gives the agent s/he has created, some strategic directions and the agents communicate with each other in a centralized agent marketplace based on the determined strategy. The new e-marketplace of MIT Media Lab, Market Maker (maker.media.mit.edu) improves upon Kasbah's agent based transaction concept, further extending agent capabilities with dynamic marketplace concept where new services and products can be added dynamically. AuctionBot (auction.eecs.umich.edu) is a general purpose auction server developed and operating at University of Michigan. It collects bids, determines a resulting allocation as entailed by a well-defined set of auction rules, and notifies the participants. Users can also generate their own agents based on the API supplied as a part of the project, and use them in auctions in the system. In Tete-a-Tete (ecommerce.media.mit.edu/Tete-a-Tete), a project within MIT Media Lab's Agent-mediated Electronic Commerce Initiative, agents cooperatively negotiate across multiple terms of a transaction like delivery time, warrancy etc. Negotiation criteria are determined by customer taste that is obtained through a multi-attribute utility ranking of the merchant offerings.

- Apart from these systems there are many catalog based marketing systems where the users select the items they want using a search engine. Amazon.com, BuyBooks.com, BookFinder.com are the specialized ones for buying and selling books while PriceWatch is for computer products and Carpoint.msn is for automobiles. RoboShopper (www.roboshopper.com) and Netmarket (www.netmarket.com) are marketplaces that comprise various products and are based on textual search engines. RoboShopper tries to help the users in selection of the merchant. It is a comparative shopping marketplace price being the comparison criteria. The comparison engine used belongs to PriceSCAN (www.pricescan.com). eBay (www.ebay.com) is another successful marketplace selling refurbished and second hand products through a choice of auction protocols. eBay provides over a half million new auctions every day from which users may choose from more than 1,000 categories and participate in auctions or create new auctions for their own goods. Commerce One MarketSite 3.0 Open Marketplace Platform (www.commerceone.com) provides an open solution for market makers to deploy interoperable marketplaces. It is comprised of software and commerce services that enable trading partners to seamlessly exchange business information and provide access to value-added services that are key to efficiently conducting commerce. Commerce One MarketSite.net (www.marketsite.net), built on MarketSite 3.0, is an open business to business marketplace for electronic procurement providing unprecedented commerce services and the ability to interoperate with numerous buying and selling applications.

- Many of the marketplaces today are auction based and the auction mechanisms used imitate their real life counter parts. They can be classified as follows: single-sided auctions and double-sided auctions. In a single-sided auction there is a single buyer and many sellers or a single seller and many buyers. Examples of such auctions include the English open outcry auction in which the auctioneer calls a price and the bidders indicate their acceptance; the Dutch auction, in which the auctioneer starts at a high price and lowers it until someone accepts; or the Vickrey auction where bids are sealed and the highest bidder wins at the second bidder's price. In a double-sided auction there are many buyers and many sellers, and every buyer can also be a seller. Examples include the continuous double auction, in which bids and asks are matched in the order received, and the sealed (clearinghouse) double auction, where bids and asks are collected for a predetermined time interval and are matched at the end of the interval according to price and arrival order. Auction based marketplaces require consumers to manage their own negotiation strategies over an extended period of time and this is where the agent technologies come in. When agents are involved negotiation schemes can be automated relieving the user from this task. Many attributes such as price, deadline, delivery time, etc. can be negotiated among many agents. Therefore, a multi-issue, multi-party negotiation algorithm is necessary. Negotiation model can be based on Raiffa's bilateral (two parties, many issue) negotiation. This bilateral negotiation model can be transformed into multilateral (many parties, many issues) negotiation model by using a set of mutually influencing two parties, many issues negotiations. This type of negotiation is also called as integrative bargaining. Buying and selling agents which realize the negotiation process in our electronic marketplace makes use of this multi-issue, multi-party negotiation algorithm.

5. Conclusions

Electronic commerce is one of the most exciting and fast moving fields of today with a high demand for innovative new technologies [1, 3, 6]. In this paper we describe a workflow architecture specifically designed for Internet electronic commerce applications and describe its usage through an electronic marketplace scenario.

The proposed workflow architecture provides for the requirements of the marketplace application described as follows:

- It is possible to model the whole commerce process through workflow templates.
- It is possible to modify the templates according to customer requirements by using the dynamic modification tools provided by the system. For example if a customer does not prefer automatic payment that activity will not be included in the template definition.
- A customer only with a browser can access the system.
- The system is higly available since it is possible to replicate the component repositories and the domain monitoring tools through the proposed workflow architecture.
- The system is highly scalable since the components are activated only when they are actually needed.
- The system has an open infrastructure based on CORBA and XML.

Currently we are in the process of implementing the proposed architecture. Future work includes research in how the workflow system architecture can be enhanced to meet the requirements of supply-chain management across organizations.

References

[1] N. Adam, and Y. Yesha, *Electronic Commerce: An Overview*, Springer Verlag, 1996.

[2] S. Arpinar, A. Dogac, and N. Tatbul, *An Open Electronic Marketplace through Agent-based Workflows: MOPPET*, International Journal on Digital Libraries, to appear.

[3] A. Dogac, Guest Editor, ACM Sigmod Record Special Section on Electronic Commerce, 27(4), December 1998.

[4] A. Dogac, I. Durusoy, S. Arpinar, E. Gokkoca, N. Tatbul, and P. Koksal,*METU-EMar: An Agent-based Electronic Marketplace on the web*, in [12].

[5] A. Dogac, I. Durusoy, S. Arpinar, N. Tatbul, P. Koksal, I. Cingil, and N. Dimililer, *A Workflow-based Electronic Marketplace on the web*, in [3].

[6] A. Dogac, Guest Editor, Distributed and Parallel Databases, Special Issue on Electronic Commerce, 7(2), Kluwer, April 1999.

[7] A. Dogac, C. Beeri, A. Tumer, M. Ezbiderli, N. Tatbul, C. Icdem, G. Erus, O. Cetinkaya, and N. Hamali, *MARIFlow: A Workflow Management System for Maritime Industry*, MAREXPO Book, to appear.

[8] A. Dogac, M. Ezbiderli, N. Tatbul, A. Tumer, C. Icdem, G. Erus, O. Cetinkaya, and C. Beeri, *A Workflow System through Cooperating Agents for Document Flow over the Internet*, Technical Report, SRDC, Middle East Technical University, April 1999.

[9] P. Koksal, I. Cingil, and A. Dogac, *A Component-based Workflow System with Dynamic Modifications*, in Proc. of the Next Generation Information Technologies and Systems (NGITS'99), Israel, 1999.

[10] B. Meltzer, and R. Glushko, *XML and Electronic Commerce: Enabling the Network Economy*, in [3].

[11] P. Muth, J. Weissenfels, and G. Weikum, *What Workflow Technology Can Do for Electronic Commerce*, in Current Trends in Database Technology, A. Dogac, T. Ozsu, O. Ulusoy, editors, Idea Group Publishing, 1998.

[12] C. Nicolaou, and C. Stephanidis, editors, *Research and Advanced Technology for Digital Libraries*, Lecture Notes in Computer Science, Springer.

[13] R. Orfali, D. Harkey, and J. Edwards, *The Essential Client/Server Survival Guide*, Second Edition, John Wiley, 1996.

[14] R. Orfali, and D. Harkey, *The Essential Client/Server Programming with JAVA and CORBA*, John Wiley, 1997.

[15] P. Timmers, *Internet Electronic Commerce Business Models*, http://www.ispo.cec.be/ ecommerce/ busimod.htm.

[16] *Extensible Markup Language (XML)*, http://www.w3.org/XML/.

[17] *VEO Systems Inc.*, http://www.veosystems.com, 1998.