DOI:10.2298/CSIS090608019P

Usage of Agents in Document Management

Dragoslav Pešović[1], Milan Vidaković[2], Mirjana Ivanović[1], Zoran Budimac[1], and Jovana Vidaković[1]

[1]Department of Mathematics and Informatics, Faculty of Science,
University of Novi Sad, Trg D. Obradovića 4,
21000 Novi Sad, Serbia
{dragoslav,mira,zjb,jovana}@dmi.uns.ac.rs
[2]Computing and Control Department, Faculty of Technical Sciences,
University of Novi Sad, Trg D. Obradovića 6,
21000 Novi Sad, Serbia
minja@uns.ac.rs

Abstract. EXtensible Java-based Agent Framework (XJAF) is a pluggable architecture of the hierarchical intelligent agents system with communication based on KQML. Workers, Inc. is a workflow management system implemented using mobile agents. It is especially suited for highly distributed and heterogeneous environments. The application of the above-mentioned systems will be considered in the area of Document Management Systems.

Keywords: Mobile Agents, Workflow Management Systems, Document Management.

1. Introduction

According to the most general definition, a mobile agent is a program that is able to stop its execution at one node in a computer network, and to transfer itself to another node where its execution continues. An important feature of mobile agents is their autonomous behavior: a mobile agent autonomously decides when and where it will be transferred.

EXtensible Java-based Agent Framework (XJAF) [32] is a pluggable architecture of the hierarchical intelligent agents system with communication based on KQML. This framework supports pluggable software managers that are dealing with a particular job. The system is designed so that it is possible to choose an arbitrary manager when configuring provided that it implements the given interface. This enables the use of arbitrary managers whose existence is not necessary at compile-time. The system is compliant to the FIPA specification and has been implemented using Java Enterprise Edition (JEE) technology.

Workflow [13, 37, 38] can be defined as the automated part of a business process, organized as a collection of activities, where documents, information or tasks are passed between participants according to a set of procedural rules. A workflow management system (WFMS) provides for defining, creating, and managing of workflow instances.

The usage of mobile agents [14] in modeling and implementation of a workflow [37] simplifies the workflow management. Workers, Inc. [25] consists of individual agents with autonomous behavior. Mobile agents carrying out workflow instances (the so-called workers) have the ability to move to different users, where they can interact with them locally, autonomously taking care of their current position, state, and further itinerary. In order to achieve the flow of work, workers split the work in logical parts, cooperate together, and synchronize themselves.

To allow the exchange of process definitions with various other workflow products (ranging from other workflow management systems to modeling and simulation tools), the system had to be made compliant with XML Process Definition Language (XPDL) [39], the proposed standard in the area of workflow definition languages. In order to comply with XPDL, the system first had to be modified to conform to the basic constructs of XPDL and the underlying meta-model. Moreover, a system-specific import layer had to be provided to allow the translation of XPDL process definitions, generated using a visual modeling tool, into worker execution contexts, their internal system representations.

The application of the above-mentioned systems will be considered in the area of Document Management Systems.

The rest of the paper is organized as follows. In the next section, the related work is presented. Main concepts of agent frameworks are described in the third section, introducing basic technologies that can be used for their implementation. The section 4 presents the architecture of XJAF, while sections 5 and 6 describe the architecture of Workers, Inc. The seventh section provides a brief introduction into the area of document management systems, discussing possibilities of an agent-oriented approach to the design and implementation of such systems. Finally, the eighth section concludes the paper.

2. Related Work

Agent frameworks can be analyzed from several points of view. From the problem domain point of view, frameworks can be general-purpose [1, 2, 8, 9], or specialized ones, which solve particular problems [18, 36]. Also, from the technology point of view, agent frameworks are based on either proprietary solutions or on solutions based on the distributed components technology. Agent frameworks like JAF (Java Agent Framework) [8] and JAT (Java Agent Template) [9] are based on proprietary solutions, while Aglets [1] and JADE (Java Agent DEvelpment framework) [2] are based on the RMI, CORBA and Java EE technology.

The large number of papers is related to the security issues in agent frameworks [3, 12, 30, 35]. Security issues regarding agent frameworks include: providing message integrity, code protection during agent migration and protecting agent frameworks from malicious agents.

This paper presents an implementation of an agent framework which is based on the Java EE technology. All important elements of this framework are implemented as plug-ins, which provides for flexibility in both design and implementation.

Several authors have recently suggested a usage of agents in workflow and document management.

Rather than going top-down in describing possible use of mobile agents in workflow management, we take bottom-up approach. Workers, Inc. is highly decentralized and consists solely of individual agents with autonomous behavior, which differentiates it from approaches in [5, 6, 15, 20]. The only centralized control in our system is the control of user rights to create, access, and change agents and templates, while all mentioned papers describe systems that had some forms of centralized control or services.

With respect to decentralization, our system resembles [28] that is based on static CORBA objects. While decentralization in [28] was one of explicit design goals and had to be explicitly implemented, decentralization in our system comes for free as a natural consequence of agent mobility and autonomous behavior. Moreover, our system is uniform – it is designed to use only one mechanism (mobile agents), without the need for additional mechanisms (transactions, HTTP protocol, HTML documents, Web browsers, CORBA, etc.)

While in [16] full decentralization of WfMS using mobile agents is shortly mentioned, the paper in fact describes the usage of mobile agents in centralized WfMS and only for external parties. Moreover, agents in [16] are specialized and created to one task only. Itinerary is saved on hosts instead in agents themselves which reduces the agent autonomy. Our system completely relies on agents and is fully distributed with autonomous agents.

Stromer in [29] describes similar goals and advantages of using mobile agents in WfMS as we are, but his implementation is different.

Among document management systems proposed over the years, there are some that are agent-based [7, 27] or only agent-enhanced (where an agent layer is added on top of existing infrastructure) [21]. However, none of the mentioned systems emphasizes the benefits of using agent mobility.

Proposed workflow and document management systems bring some fresh views not only in particular fields of workflow and document management, but in mobile computing as well. In both fields, the advantages of highly decentralized and distributed approach in designing a system have not been often recognized. The most cited advantages of mobile agents with respect to classical techniques of distributed programming are:

1. Potentially better efficiency of the whole system. A client program migrates to a server node, locally communicates with a server program, and returns to the original node with a result. In that way, the overall network traffic, including the number of remote interactions and the amount of data communicated over the network, is therefore potentially reduced.

2. Greater reliability, because the connection between nodes must not be established all the time.

Our workflow and document management systems emphasize the fact that mobile agent has organizational advantages as well. Mobile agent systems can be regarded as a separate programming paradigm and not only as an improvement of distributed programming style. Solutions to some problems are easier to program, understand, and maintain, if implemented using mobile agents.

3. Agent Frameworks

Agent technology [14] represents one of the most consistent approaches in distributed systems implementation. Software agents realize distributed component concept entirely. This means that besides solving the problem, agents utilize a certain degree of intelligence and autonomy that are needed to solve the problem. Therefore, agents represent software entities capable of searching and processing the large quantity of information, utilizing a certain degree of intelligence, autonomy and communication.

Agents need a programming environment which will create and enable agents to execute tasks. Agent framework [14] represents programming environment that controls agent life cycle and provides all necessary mechanisms for task execution (communication, agent mobility, services and security). Besides controlling life cycle of an agent, an agent framework also provides messaging and service subsystems to effectively support agents. Messaging allows agents to communicate to each other, and service subsystem gives them the possibility of accessing various resources or executing complex algorithms that are not needed to be implemented in the agent itself. An agent framework also provides agent mobility and security. Agent mobility allows agents to migrate from one agent framework to another. The security subsystem provides security mechanisms which protect both agents and frameworks. It is also necessary to provide mechanism of searching agents and services present in agent framework. This mechanism is called a directory and it represents searchable repositories of agents and services which can be used by both agents and their clients.

Agent framework implementations rely on Object Oriented (OO) techniques. It is possible to implement agents easily and effectively taking advantage of its features (encapsulation, inheritance, polymorphism, dynamic binding and persistence). Most of the existing agent frameworks are implemented using Java programming language [11]. Java-based agent frameworks usually use RMI (Remote Method Invocation) [11], CORBA (Common Object Request Broker Architecture) [19] and Java EE (Java Enterprise Edition) [10] technologies for distribute code execution.

CORBA is a standard created by the Object Management Group (OMG) consortium. This standard defines the framework for creating objects being executed on the server side, and it also defines the servers themselves. It is based on the Internet Inter-ORB Protocol (IIOP). This standard anticipates the execution of components written in all supporting programming languages. Besides above mentioned concepts, CORBA supports transactions and its own components naming and search system (Object Naming Service - COS Naming).

The Java EE technology is particularly useful for agent framework implementation because it comprises a large set of technologies and provides for scalability, reliability and has a large number of implementations. One element of the Java EE technology is particularly useful – the EJB (Enterprise JavaBeans) technology. This is a technology of distributed software components which are created, executed and destroyed in the application servers. All performance-related issues like load-balancing, distribution-per-server, etc. are left to the implementation of the application server. Besides supporting distributed components, Java EE also has all other technologies for the agent framework implementation: JMS (Java Message Service) for message exchange, JNDI (Java Naming and Directory Interface) for directory implementation, Java Security, etc.

4. XJAF

The EXtensible Java-based Agent Framework (XJAF) [32] is based on the J2EE technology. The system consists of clients and facilitators. The clients refer to the facilitators for task execution. The task is being executed by the agents engaged by the facilitator. The Figure 1 shows the link between a client and an agent framework.

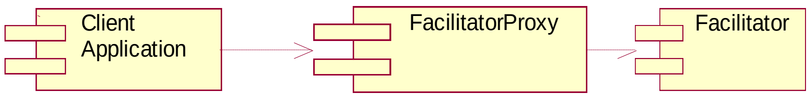

Figure 1. Client and Agent framework link

The client assigns the task to the facilitator; the facilitator engages an agent to execute the task and returns the result to the client. The FacilitatorProxy class ensures that the client application can access the facilitator. It also hides all techniques necessary for work with agents from the client. The client only needs to create an object of the FacilitatorProxy class and to pass it the class representing the task or the KQML message, as well as the corresponding listener, which would notify it of the result. All other details are managed by the FacilitatorProxy class.

The client is any Java application. The facilitator is an instance of the Facilitator class. The Facilitator class realizes the facilitator functionality. The agent is an instance of a class implementing the Agent interface and realizing the functionality of an individual agent.

Extensibility of this framework is based on the plug-in concept. Plug-ins are realized as pluggable managers. The facilitator forwards the parts of its job to the corresponding pluggable managers. The managers are instances of classes implementing the corresponding managerial interfaces. The AgentManager interface is responsible for allocating and releasing agents. The TaskManager interface manages the tasks. The MessageManager interface is responsible for interagent communication. The ConnectionManager interface manages facilitator connection and relations. The SecurityManager handles security of inter-agent communication.

The classes that implement the mentioned interfaces implement the corresponding algorithms for individual functions. The system is designed so that it is possible to choose an arbitrary manager when configuring provided that it implements the given interface. This enables the use of arbitrary managers whose existence is not necessary at compile-time, but not until initialization (plug-in concept). The Figure 2 lists all the managers in the framework.

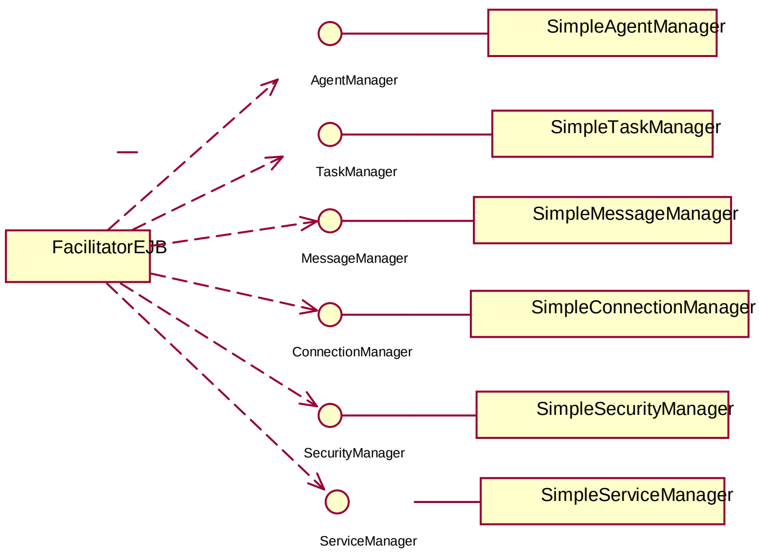

Figure 2. Functionality of individual parts is assigned to managers

4.1. Agent Manager

Agent management is done using the AgentManager component. Controlling the agent life cycle means creating and destroying an agent.

This component is also used as an agent directory. All relevant data is kept in a repository. The repository can be a database, or an LDAP server, or any other data storage. This manager also keeps track of all local agents required by external facilitators, and of all agents that have been moved to another facilitator.

4.2. Task Manager

The TaskManager component manages tasks to be performed by the agent framework. It is realized through the class which implements the TaskManager interface. It also provides a way of notifying the client about the task execution progress.

Each task is stored in this component. When completed, it is removed from it. Tasks are instances of classes which implement the AgentTask interface. There are two types of task execution: programmatically or by sending a KQML message to the agent.

The method execute() is run asynchronously, i.e. it does not block the execution of the client's code for the time of task execution, but it calls the instance of the class inheriting the interface AgentListener created by the client application upon and during the task execution. In this way the client application can proceed with the code execution and it will be notified of the agent's results by the listener reference.

When executing a task by sending a KQML message to the agent, the client application sends the KQML message to the Facilitator component. This component looks for the appropriate agent and sends the message to it. When the task is completed, the agent replies to the original message and the message is forwarded to the client using the FacilitatorProxy component.

4.3. Message Manager

The exchange of messages between the agents is actually KQML messages exchange. These messages are encapsulated in the base class - the class KQMLMessage. The messages are exchanged by passing the KQMLMessage class objects between the dialog participants. All the communication is done by the MessageManager component.

When an agent sends a KQML message to another agent, it is embedded into a JMS message. The JMS message is sent to all agent frameworks subscribed to this service, but only the agent framework having the destination agent will receive the message and extract the KQML message from it. This KQML message is then sent to the agent.

4.4. Connection Manager

The ConnectionManager component defines an inter-facilitator connectivity mechanism. This mechanism defines how separate facilitators form a network. Each facilitator is a node in this network and is automatically registered on the network at the initialization time. This means that the programmer does not have to know the exact address of an arbitrary facilitator and does not have to maintain the list of all available facilitators.

Instead, the nodes are registered automatically and the list of all available facilitators is maintained automatically.

The facilitators form a certain hierarchy structure. One approach is to form a tree structure with the primary facilitator in the root. The Figure 3 shows this organization.

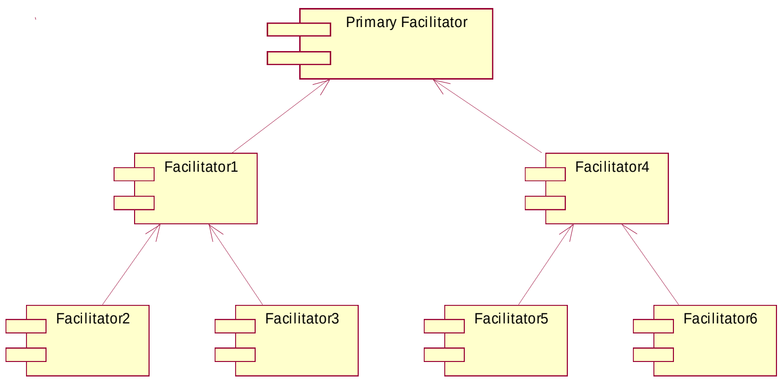

Figure 3. Component diagram of facilitator hierarchy

4.5. Security Manager

SecurityManager component [33] handles security issues. It provides encryption, decryption, signature generation and verification for all messages passing through the framework. Also, this manager handles access to local resources. Access control segment of security subsystem insures integrity of data and code. It provides for integrity of data exchanged between agents and also protects agent framework from malicious agents. Any cryptographic system can be used since this manager is proposed by the SecurityManager interface, and implementation is left to the developer.

4.6. Service Manager

The ServiceManager component implements service directory subsystem. This component manages the set of services available to agents.

The ServiceManager component includes the service repository which holds all available services. Services can be added, removed, searched and used. When the service is not needed anymore, it must be returned to the repository. Services are implemented as Java classes which implement the Service interface.

5. Workflow Management System Using Mobile Agents

Workers, Inc. [4, 22, 23, 24, 25, 26] is a workflow management system under development at the University of Novi Sad. The system is implemented using the technology of mobile agents and is therefore especially suited for highly distributed and heterogeneous environments.

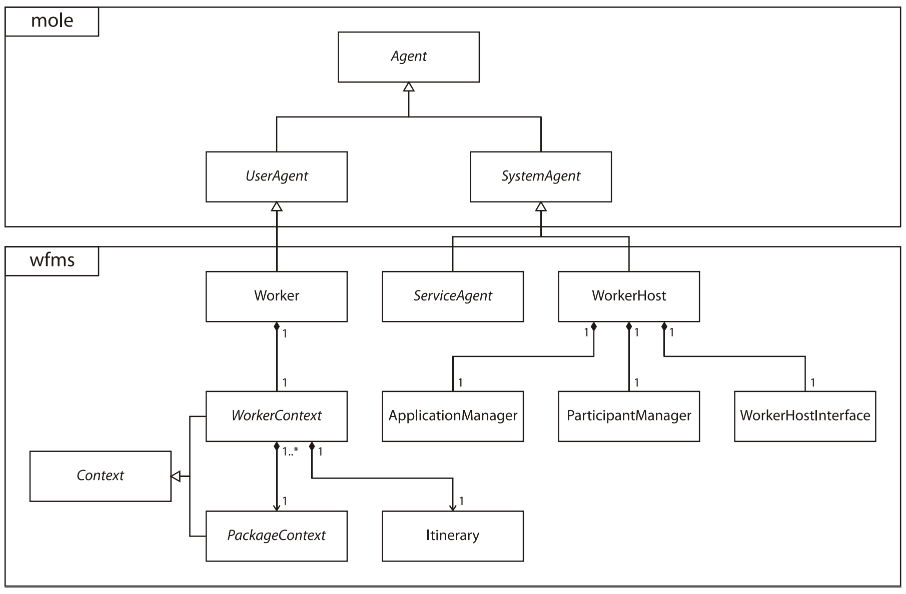

Figure 4. The architecture of Workers, Inc.

Workers, Inc. is envisioned as a community of cooperative agents, its main characteristics being full decentralization and distribution of workflow functions. The current architecture is essentially two-part, consisting of work-agents (workers) and host-agents (worker hosts). Workers, Inc. is built on top of a Java-based mobile agent system, and uses Java as the language of implementation as well as of the agent development. Agent migration and inter-agent communication benefit from Java RMI and class serialization, and Java sandbox security model is the basis for providing secure agent execution environment. Java API for XML Processing (JAXP) is used for XPDL document parsing.

Process definitions are being completely handled by workers, while the enactment is achieved through the cooperation of a worker carrying a process definition (or a set of workers when concurrency or subprocesses are involved) and worker hosts residing at every node of the network. Worker hosts represent central components of the system mediating between the underlying system, workers, and human users.

5.1. Workers

A worker is the key system component encapsulating both the process definition and the execution state of a workflow. While performing a workflow, a worker itinerates among distributed resources carrying process-specific information and autonomously taking care of its execution state. In that way, workers manage not only to perform workflow activities locally with respect to assigned resources, but to avoid the need to consult a central server or the originating machine at every step.

A worker's behavior is entirely defined by its execution context. A worker context is an executable process definition, a worker being just a medium through which its context is transmitted and accomplished. When a worker migrates, its entire execution context as an object net is being encompassed by object serialization, and then transported and reconstructed at the target location.

The most important part of a context is the worker itinerary, which represents a flow of a worker through a network. By representing itineraries with directed graphs we are able to represent complex flow patterns that could be needed by workflow applications.

To allow concurrent activity execution, agent social abilities are employed. When a single thread of control needs to split into two or more threads, which can be executed in parallel, the worker context is cloned and multiple worker instances are allowed to be executed simultaneously. On the other hand, when multiple parallel threads of execution need to converge into a single thread, agent coordination mechanisms and synchronization techniques are employed.

To strengthen security of the system, mobile agents and thus workers are forbidden to access any system resources directly. Critical resources can be accessed only by communicating with system agents, i.e. worker-hosts.

5.2. Worker Hosts

Every node in the network contains a worker host, which is implemented as a stationary system agent, having special privileges for the access to host system resources. A worker host is a passive entity, which spends most of its lifetime receiving requests from workers or users and coordinating their actions. There are three main subcomponents of a worker host: an application manager, a participant manager, and a user interface.

5.3. Other Specialized Agents

Although workers are almost fully autonomous, they may need additional services to finish their work. Those services cannot be embedded directly into the workers as this would prevent keeping workers as small as possible.

Services are therefore implemented separately, as specialized stationary agents.

Workers, Inc. is a fully distributed system, without central administration, control, and maintenance. All reports, control, and management can be achieved by creating and sending specialized agents that will communicate with other agents in the system and achieve the intended results. Those agents may be mobile or stationary, depending on the nature of the task they are intended to accomplish.

6. Worker Execution Contexts

The design of an execution context is done so as to comply with the workflow meta-model specification. From the control-flow perspective, the itinerary is the most important part of a context.

6.1. Itinerary

The itinerary has the structure of an arbitrary complex directed graph, where vertices of the graph represent process activities, and edges of the graph correspond to process transitions.

6.2. Activities

An activity is the smallest, atomic unit of work in a business process. The three main properties of an activity specification, which can be seen as answers to the accompanied questions, are:

- Performer assignment (Where?) – It specifies the performer of the activity. In the process of workflow participant resolution, the actual location of a participant is determined. By evaluating a performer expression, a worker knows where its activity needs to be carried out, and will transfer itself over the network accordingly.
- Implementation specification (What?) – It specifies what the concrete realization of the activity is. It can be a call to a declared application, another workflow process, or an embedded activity set. Also, the activity may have no implementation at all, in which case it supports complex flow transitions or manually performed activities.
- Automation modes (How?) – Information on whether the activity is to be started / finished manually by the user or automatically by the worker itself.

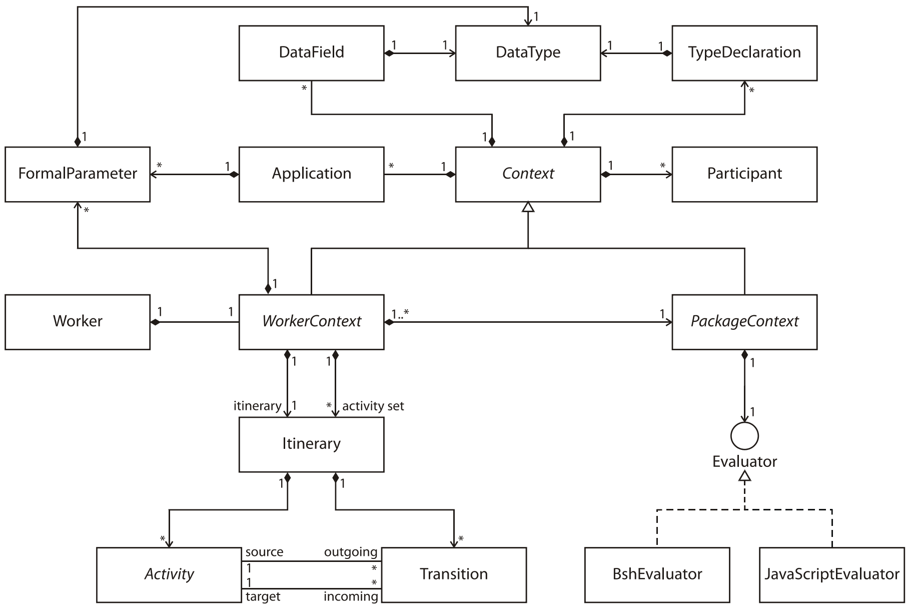

Figure 5. Worker and its contexts

6.3. Transitions

Transitions connect individual activities. A transition may contain a condition which must be fulfilled for the worker to start performing the target activity. If the transition does not contain a condition, the worker will start the target activity immediately after the source activity has completed. If the performer assigned to the target activity is different than the one of the source activity, the worker will first transfer itself to the appropriate node in the network, before it actually starts the activity.

The layout of transitions within a process graph may cause the sequential or parallel operation of individual process activities. If there are multiple incoming or outgoing transitions of an activity, control flow restrictions and condition evaluation semantics may be expressed within the appropriate activity: split as a form of post-activity processing in the source activity, and join as a form of pre-activity processing in the target activity.

7. Document Management

A document management system (DMS) [7, 17, 21, 27] is a computer system (or set of computer programs) used to track and store electronic documents and/or images of paper documents. Document management controls the life cycle of documents in an organization – how they are created, reviewed, published, and consumed, and how they are ultimately disposed of or retained.

A well-designed document management system promotes finding and sharing information easily. It organizes content in a logical way, and makes it easy to standardize content creation and presentation across an enterprise. It promotes knowledge management and information mining. It provides features at each stage of a document's life cycle, from template creation to document authoring, reviewing, publishing, auditing, and ultimately destroying or archiving.

7.1. Main Features of Document Management Systems

There are several common issues that are involved in the document management. Document management systems commonly address the following issues:

- Location. Where will documents be stored? Where will people need to go to access documents? How content moves between locations? It may be necessary to move or copy a document from one site or library to another at different stages of its life cycle. For example, the publishing process may include moving a document from a staging site to a public Internet site. If content needs to be converted from one format to another as it moves from site to site, content conversions must be specified.
- Filing. How will documents be filed? What methods will be used to organize or index the documents to assist in later retrieval? Documents can be organized in free-form document libraries for ad-hoc document creation and collaboration, or specialized sites such as team sites and portal sites. Databases can be used to store filing information.
- Retrieval. How will documents be found? Typically, retrieval encompasses both browsing through documents and searching for specific information.
- Security. How will documents be kept secure? How will unauthorized personnel be prevented from reading, modifying or destroying documents?
- Retention period. How long should documents be kept, i.e. retained?
- Archiving. How can documents be preserved for future readability?
- Distribution. How can documents be available to the people that need them?
- Workflow. By planning workflows, one can control and track how documents move from one team member to another as each participant collaborates in a document's life cycle. A system may include workflows for common team tasks such as reviewing and approving documents. It may also support creating and installing custom workflows.
- Creation. How are documents created? This question becomes important when multiple people need to collaborate, and the logistics of version control and authoring arise.
- Authentication. Is there a way to vouch for the authenticity of a document?

- Content types. Content types can be used to organize information about types of documents, such as metadata, document templates, policies, and workflow processes.

7.2. Agent-Oriented Approach

The usage of software agents in modeling and implementation of a document management system simplifies the document management. The organization and implementation of the system are easy to understand and follow, because most of its parts are uniformly implemented as (mobile) agents:

- User agents to assist individual users (with incorporated access rights). Every user of the document management system would have a devoted user agent to assist him/her in the authoring and access processes. Those user agents would communicate with other agents in the system directly, or create and send specialized mobile agents in order to achieve the intended results. Specific user's rights will be incorporated in his/her agent, controlling access to other agents or registered system services. Every user agent will need a user interface for communication with the user.
- Specialized agents for document retrieval, indexing, archiving, etc. Those agents may be mobile or stationary, depending on the nature of the task they are intended to accomplish. For example, a retrieval agent would be mobile, searching for documents on every site or library in the system, and gathering a report. Once back, it will present the report of the status and location of all found documents.
- Workflow agents to support all kinds of workflows within the system, including collaboration, versioning, and publishing. For example, publishing a document may involve the procedures of proofreading, peer or public reviewing, authorizing, printing and approving etc. Collaboration procedures, on the other hand, define how a group of users can work on the same document(s). Workgroups can benefit from agents to coordinate their access efforts. The Joint Paper Worker, presented in [24, 25], is an example of a collaboration workflow. A number of common workflows can be provided in advance, while keeping the possibility to create custom workflows at any time.

Since the system consists of many autonomous agents, the system is easily changed, extended, and improved. It is often needed just to introduce new agents, without the need to change and even to understand the rest of the system.

8. Concluding Remarks

The idea of implementing a document management system involved two modern, attractive and promising fields in computer science:

1. Software agents with the attributes they possess: autonomy, social ability, responsiveness, proactiveness. Also, two distinguishing characteristics are very important and make agents more promising for application in different areas:
   - high-level tasks can be delegated to agents who will autonomously carry them out,
   - agents are situated in an environment which can dynamically affect their problem solving behavior and strategy.
2. Workflow is concerned with automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve or contribute to an overall business goal.

The approach to an agent framework implementation using the Java EE technology provides for scalability and reliability. This approach offers agent and service directory services, security, message exchange and agent mobility. The EJB technology, as a part of the J2EE concept, offers the simple use of all technologies necessary to implement this approach. The future work will include defining and implementing the specialized agent language, based on KQML. Also, all concepts handled by managers will be developed further. The research in the field of interoperability among different platforms (including web services) and improvement of security will be taken into consideration for further real implementation.

The main characteristics of a workflow system suggested in this paper are almost full decentralization and distribution of workflow functions. The proposed organization mimics usual user activities in a real flow of work. Moreover, it relieves them (or any centralized control) from the need to know what to do next with the work-agent. Every user takes care only of work-agents that are currently on its node. Where they came from, why they are here, and where they will go later, is not concern of the user.

Finally, the paper presented a possibility of using the two very attractive fields (agent technology and workflow) for the document management implementation, emphasizing the basic features of the proposed system. The work in these directions has already been started and more significant results are expected in the following period.

References

1. Aglets Home Page, http://www.trl.ibm.com/aglets/
2. Bellifemine, F., Poggi, A., Rimassa, G.: "JADE – A FIPA-compliant agent framework", Proceedings of Practical Applications of Intelligent Agents (PAAM'99), London, April 1999, pp. 97-108.
3. Binder, W., Roth, V.: "Secure mobile agent systems using Java: where are we heading?", Proceedings of the 2002 ACM symposium on Applied computing, 2002, Madrid, Spain, ISBN:1-58113-445-2, pp. 115-119.
4. Budimac, Z., Ivanović, M., Popović, A.: “Workflow Management System Using Mobile Agents”, Proc. of ADBIS ‘99, Lecture Notes in Computer Science 1691, Springer Verlag, Berlin, (Maribor, Slovenia), pp. 169 - 178, 1999.

5. Cai, T., Gloor, P.A., Nog S.: “Dartflow: a workflow management system on the web using transportable agents”, Technical report PCS-TR96-186, 1996.
6. Chang, W., Scott, C.: “Agent-based workflow: TRP support environment”, Computer networks and ISDN systems vol. 28, issues 7-11, 1997.
7. Ginsburg, M.: “An Agent Framework for Intranet Document Management”, Journal of Autonomous Agents and Multi-Agent Systems, volume 2, issue 3, pp. 271-286, 1999.
8. Java Agent Framework Home Page, http://mas.cs.umass.edu
9. Java Agent Template Home Page, http://java.stanford.edu
10. Java Enterprise Edition Homepage, http://java.sun.com/javaee
11. Java homepage, http://java.sun.com
12. Kim Tan, H., Moreau, L.: "Certificates for mobile code security", Proceedings of the 17th symposium on Proceedings of the 2002 ACM symposium on applied computing, 2002, Madrid, Spain, ISBN:1-58113-445-2, pp. 76-81.
13. Leymann, F., Roller, D.: “Production Workflow – Concepts and Techniques”, Prentice Hall PTR, New Jersey, 2000.
14. M. Knapik, J. Johnson. Developing Intelligent Agents for Distributed Systems. McGraw-Hill, 1998, pp. 3, 37-39.
15. Meng, J., Helal, S., Su, S.: “An ad-hoc workflow system architecture based on mobile agents and rule-based reasoning”, Proc. of Int. Conf. on parallel, and distributed computing techniques and applications, Las Vegas, 2000.
16. Merz, M., Liberman, B., Lamersdorf, W.: “Using mobile agents to support inter-organizational workflow management”, Int. J. on applied artificial intelligence 11(6), pp. 551-572, 1997.
17. Microsoft TechNet Library, http://technet.microsoft.com/en-us/library/cc261933.aspx[5]
18. Nardi, B., Miller, J., Wright, D.: "Collaborative, programmable intelligent agents", Communications of the ACM, Volume 41 , Issue 3 (March 1998), pp. 96-104.
19. OMG. Common Object Request Broker: Architecture and Specification. OMG Specification Revision 2.0, July 1995.
20. Padalkra, A., Nabar, P., Arora, S., Naik, P.: “SWIFT: Scalable workflow management system using mobile Agents”, http://www.iitb.ac.in/~pranav/php/paper.pdf, 2000.
21. Papaspyrou, N., Sgouropoulou, C., Skordalakis, E.: “A Model of Collaborating Agents for Content-Based Electronic Document Filtering”, Journal of Intelligent and Robotic Systems, vol. 26, no. 2, pp. 199–213, October 1999.
22. Pešović, D., Budimac, Z., Ivanović, M.: “Distributed Mobile Agent Workflow - Activity Coordination Constructs in Workflow Process Graphs”, Proc. of 11th International Multiconference Information Society 2008, Ljubljana, Slovenia, Oct 13-17, 2008.
23. Pešović, D., Budimac, Z., Ivanović, M.: “Towards a Visual Definition of a Process in a Distributed Environment”, 2nd International Symposium on Intelligent Distributed Computing - IDC'2008, Catania, Italy, 2008.
24. Pešović, D., Budimac, Z.: “Advanced Joint-Paper Worker”, CCS Journal, 4th issue, pp. 44-46, March 2003.
25. Pešović, D.: “A High-Level Language for Defining Business Processes”, PhD Thesis, University of Novi Sad, Novi Sad, 2007.
26. Pešović, D.: “The Implementation of a Workflow Management System Using Mobile Agents”, Master Thesis, University of Novi Sad, Novi Sad, 2002.
27. Roberto, V., Della Mea, V., Di Gaspero, L., Conti, A.: “MANTHA: Agent-based Management of Hypermedia Documents”, Proceedings of 6th IEEE Int. Conf. on

Multimedia Computing and Systems (IEEE ICMCS '99), Firenze, IEEE Computer Society, vol II, pp. 814-818, 1999.
28. Sheth, A., Kochut, K., Miller, J., Worah, D., Das, S., Lin, C., Palaniswami, D., Lynch, J., Shevchenko, I.: “Supporting State-wide Immunization Tracking using Multi-Paradigm Workflow Technology”, Proc. of 22nd VLDB Conference (Bombay, India), 1996.
29. Stormer, H.: “A flexible agent-based workflow system”, Workshop on Agent-based approaches to B2B, 2001.
30. Varadharajan, V.: "Security enhanced mobile agents", Proceedings of the 7th ACM conference on Computer and communications security, Greece, Athens, November 2000, pp. 200-209.
31. Vidaković M., Konjović Z.: "One Implementation of Virtual Library Based on Agent Technology", Proceedings of the 4th Conference on Informatics and Information Technology, Bitola, 11-14 of December, 2003
32. Vidaković, M., Konjović, Z., "EJB Based Intelligent Agents Framework", Proceedings of the 6th IASTED International Conference on Software Engineering and Applications (SEA 2002), Cambridge, USA, November 4-6, 2002, pp. 343-348
33. Vidaković, M., Sladić, G., Konjović, Z., "Security Management In J2EE Based Intelligent Agent Framework", Proceedings of the 7th IASTED International Conference on Software Engineering and Applications (SEA 2003), Marina Del Rey, USA, November 3-5, 2003, pp. 128-133
34. Vidaković, M., Sladić, G., Zarić, M.: "Metadata Harvesting Using Agent Technology", Proceedings of the 8th IASTED International Conference on Software Engineering and Applications (SEA 2004), Cambridge, USA, November 9-11, 2004., pp. 489-493
35. Vuong, S., Fu, P.: "A security architecture and design for mobile intelligent agent systems", ACM SIGAPP Applied Computing Review archive, Volume 9 , Issue 3 (Fall 2001), pp. 21-30.
36. Wilson, L., Burroughs, D., Sucharitaves, J., Kumar, A.: "An agent-based framework for linking distributed simulations", Proceedings of the 32nd conference on Winter simulation, 2000 , Orlando, Florida, ISBN:1-23456-789-0, pp. 1713 – 1721.
37. Workflow Management Coalition, Hollingsworth, D.: “The Workflow Reference Model”, Homepage of Workflow Management Coalition, 1995.
38. Workflow Management Coalition: “Terminology and Glossary”, Homepage of Workflow Management Coalition, 1999.
39. Workflow Management Coalition: “Workflow Process Definition Interface – XML Process Definition Language, Version 1.0”, Homepage of Workflow Management Coalition, 2002.

Dragoslav Pešović is an assistant professor at Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad. His research interests include: mobile agents and distributed systems. He has published 10 scientific papers in proceedings of international conferences, has written 2 university textbooks.

Milan Vidaković is holding the associate professor position at the Faculty of Technical Sciences, Novi Sad, Serbia. He received his PhD degree (2003) in Computer Science from the University of Novi Sad, Faculty of Technical Sciences. Since 1998 he has been with the Faculty of Technical Sciences in Novi Sad. Mr. Vidaković participated in several science projects. He published more than 60 scientific and professional papers. His main research interests include web and internet programming, distributed computing, software agents, and language internationalization and localization.

Zoran Budimac is a professor at Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad. He graduated in 1983 (informatics), received master's degree (computer science) in 1991 and doctor's degree (computer science) in 1994. His research interests include: mobile agents, e-learning, software engineering, case-based reasoning, implementation of programming languages. He has been project leader for several international and several national projects. He has published over 180 scientific papers in proceedings of international conferences and journals, has written more than 12 university textbooks in different fields of informatics.

Mirjana Ivanović is a professor at Faculty of Sciences, Department of Mathematics and Informatics, University of Novi Sad. She graduated in 1983 (informatics), received master's degree (discrete mathematics and programming) in 1988 and doctor's degree (computer science) in 1992. Her research interests include: multi-agent systems, e-learning and web-based learning, data mining, case-based reasoning, programming languages and tools. She actively participates in more than 10 international and several national projects. She has published over 190 scientific papers in proceedings of international conferences and journals, has written more than 10 university textbooks in the field of informatics and ICT. She is the Head of Computer Science Chair.

Jovana Vidaković is a teaching assistant at Faculty of Sciences, Department of Mathematics and Informatics, University of Novi Sad. She graduated in 1999 (informatics), received master's degree in 2003. Her main research interests include databases, XML technologies, web and internet programming. She has published 7 scientific papers and has been coauthor of 1 university textbook.

Received: June 08, 2010; Accepted: February 04, 2010.