A Distributed Multiagent Workflow System

César A. Marín and Ramón F. Brena

Centro de Sistemas Inteligentes
Tecnológico de Monterrey, Campus Monterrey
Eugenio Garza Sada 2501
C.P. 64849. Monterrey, Mexico
{cesarmp, ramon.brena}@itesm.mx

Abstract. The decentralized and distributed nature of workflow in organizations demands for support from decentralized and distributed computational systems. However, most conventional workflow applications use centralized architectures. Agent technology seems to be an adequate approach for supporting distributed systems. We have extended the capacities of a multiagent system for knowledge and information distribution in such a way that it can handle general workflow processes in a decentralized way. A working prototype is reported, and quantitative experiments have been conducted to show that the distributed workflow process flow control makes possible better scalability than the centralized counterpart.

1 Introduction

Within enterprises, streamlining processes have led to the implementation of paperless document circulation by means of *workflow* management systems (WfMS) [1, 2]. They are today a standard component of many enterprise-wide information systems and their value is widely acknowledged.

Within commercial and industrial domains, the business process execution and the process flow control are performed in a decentralized way because organizations are physically and often logically distributed. In other words, there is no central entity orchestrating each activity composing the whole business process. This decentralized and distributed nature of workflow in organizations demands for support from decentralized and distributed computational systems. However, most conventional workflow applications use centralized architectures.

In this paper we present an extension of an Information and Knowledge distribution system [3–5], which is an agent-based information system aimed to distribute the right piece of knowledge to the right person within different parts of an organization. In fact, distributing knowledge and information items could be thought of as a restricted kind of workflow, as it just comprises a document generation and its distribution, ending with the document reception by a final user. But if, for instance a document needs to pass through authorization in order to be distributed, then a more complex workflow is needed, and even this simple task is beyond the basic version of our knowledge distribution system. So, we enhanced our knowledge distribution system with general workflow capabilities.

© *A. Gelbukh, R. Monroy. (Eds.)*
Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 79-88

This paper structure is as follows: After this introduction, we present some background about our knowledge distribution multiagent system. Then, in section 3 we present our proposal. In section 4 a working prototype is presented, which is validated experimentally in section 5. Then, we compare our work with others in section 6, followed by a conclusion.

2 Background - Our System Architecture

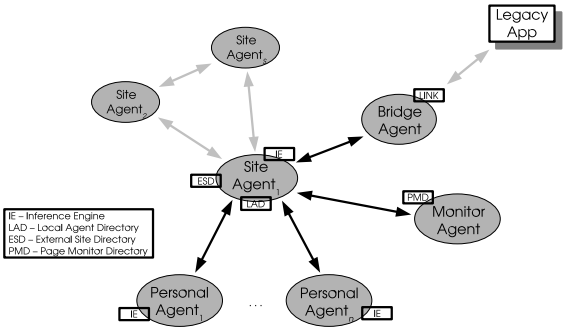


Fig. 1. Knowledge distribution system architecture.

Our workflow system is an extension of an information distribution system [6], which is based on a multiagent architecture shown in Fig. 1. It comprises some types of agents which appear in the mentioned figure but from which we are going to explain just those agents which are important for the work we are presenting:

Site Agent. This agent, works like a network router; it receives messages from any agent and distributes the information to the proper users under its site or domain. The distribution is made by first finding the corresponding users located in conceptual hierarchies. These hierarchies may represent organizational departments, interest areas, work groups, etc. Each Site Agent keeps in touch with others Site Agents so that they all together make a network of agents for information distribution.

Personal Agent. Each user may have one personal agent that filters the information addressed to the user and shows it through a web browser, sends him/her an e-mail or a message by a SMS service.

3 The Proposed Architecture

In our architecture, we are going to take the "personal agent" of JITIK as the basic workflow executers. The proposed solution for decentralized workflow process management consists in breaking down the workflow process execution

and the process flow control into small execution units handled by intelligent agents, and allowing the agents to reflect the organizational structure and the way processes are controlled and executed, i.e., distributed and decentralized.

For this purpose, two agent types are required: a new agent type named *Registry Agent* for holding process descriptions, keeping track of all the running processes at every moment and creating process instances on demand; and the existing *Personal Agent* for assisting its user/worker to perform his/her assigned tasks. In the end, Personal Agents are the actual organizational processes orchestrators in a distribution and decentralized fashion.

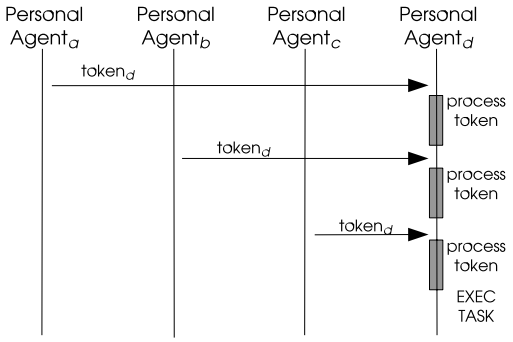
Once all agents (one Registry Agent and one Personal Agent for each user participating in a process) are up and running, the Registry Agent receives a process description [7] as input and segments it into atomic task descriptions. A *task description* is composed by the process identifier this task belongs to; the information to be handled which can be a link to a document; the assigned Personal Agent referenced by its user description in terms of the organization, i.e., the Personal Agent of the user in department D and position P; a list of tasks to be enabled right after this task finishes its execution containing the corresponding Personal Agent reference; the join and split operations to apply; and the number of flows that converge to this task. After process description segmentation, the Registry Agent distributes each task description to the corresponding Personal Agent executor. This way, all Personal Agents know what to do in advance when a task of a process instance is running, resembling the way an organization works. It is assumed that each task is assigned to only one user, i.e., a unique Personal Agent.

3.1 Agent Communication

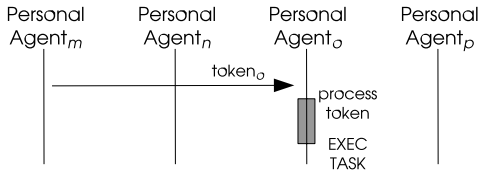
Since all tasks are distributed, Personal Agents need to send messages among them in order to enable tasks of the same process instance. In Petri Nets, a *token* is a marker that specifies in which part of the net is occurring the actual processing. In our system, a token is an agent message which contains a process ID, an instance ID and a task ID over which the message recipient must operate.

A task is enabled when its Personal Agent receives the necessary tokens for task enabling according to a join operation (AND, OR, XOR), e.g. let us assume that in a process, tasks t_a, t_b, t_c and t_d exist and are owned by Personal Agents PA_a, PA_b, PA_c and PA_d respectively, and t_a, t_b and t_c are direct predecessors of t_d which in turn synchronizes the three incoming flows, i.e., PA_d must perform an AND-join operation in order to enable t_d. Therefore, right after PA_a, PA_b and PA_c finish its task execution, each of them send a token to PA_d. And only when all three incoming tokens are received task t_d is enabled and ready for execution. A sequence diagram showing this token passing is illustrated in Fig. 2(a).

When a Personal Agent finishes a task execution and is about to enable the successive tasks in the process flow, it sends a single enabling token for each successor task to its owner Personal Agent according to a split operation (AND, OR, XOR), e.g., let us assume that in a process, tasks t_m, t_n, t_o and t_p exist



(a) AND-join operation.



(b) XOR-split operation.

Fig. 2. Enabling workflow tasks.

and are owned by Personal Agents PA_m, PA_n, PA_o and PA_p respectively, and t_n, t_o and t_p are direct successors of t_m which in turn selects one of the three outcoming flows, i.e., PA_m must perform an XOR-split operation in order to enable only one of t_n, t_o or t_p. A sequence diagram showing this token passing is illustrated in Fig. 2(a), here, task t_o was selected and thus the token was sent to PA_o.

A token can be sent by the Registry Agent or a Personal Agent. When the Registry Agent enables one or more tasks is because a process instance has just been created by it and the first tasks in such instance process are being enabled. This is the only case in which the Registry Agent is involved in the process flow control. When a Personal Agent enables one or more tasks is because it just finished the execution of one of its tasks. Notice that several tokens can be sent at a time by each Personal Agent for different process instances. Moreover, when a task status changes (e.g. from enabled to in-execution), the task owner sends a message to the Registry Agent to inform the event. This is for monitoring purpose and will not be explained here.

4 Prototype

The developed prototype for distributed and decentralized workflow process execution consists in several software layers shown in Fig. 3.

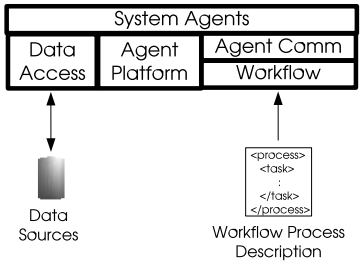


Fig. 3. Agent-based workflow software layers.

Agent Platform. The chosen agent platform for developing and executing our system agents was JADE because of its robustness [8]. Additionally, we used the JADE ACL messaging mechanism for agent communication.

Data Access. This layer is used for information access support. It allows agents to acquire information about their user or search other users' Personal Agents.

Workflow. Workflow process descriptions [7] taken as system input, are parsed and then segmented into atomic task description. This way, Personal Agents are able to know their assigned activities in advance and perform their tasks when a process instance is generated. This layer also works as information provider to the upper layer as explained below.

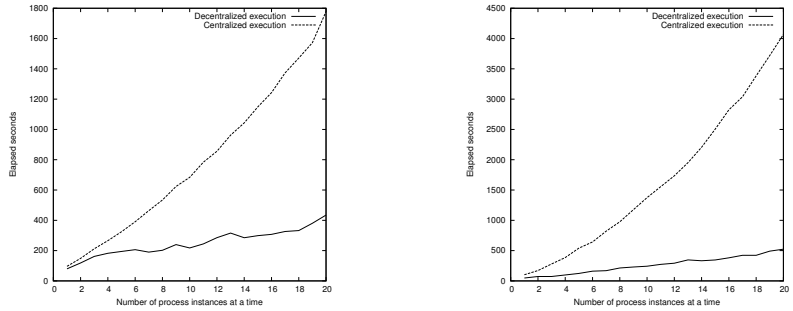
Agent Communication. On top of the Workflow layer the communication components were developed. These components are used for translating task descriptions into Tasks, as objects, so that Personal Agent can manage them. Based on these Tasks, Tokens can be generated and passed among agents for workflow enactment.

System Agents. Personal Agents, a single Registry Agent (and other system agent) are running constantly in the platform; they acquire information about users, such as who and where is his/her Personal Agent, through the Data Access layer; they rely on the Agent Communication layer for process instance creation, token passing and task enabling; and furthermore, the Registry Agent creates process instances and keeps track of all active processes.

5 Experiments

Since it is well known that a distributed application (e.g. using agents) diminishes the workload among its element while increases the communication, the objective of the experiments is to demonstrate that the proposed decentralized execution of workflow processes can be implemented (concept proof) and that performs better than a centralized approach. Thus, the performance of both approaches were compared using the elapsed time for executing certain number of process instances at a time.

Three processes were designed for this purpose, each of them representing some basic workflow patterns [9]. It was decided to test using the basic workflow



(a) Tasks in sequence (b) AND split and join

Fig. 4. Comparison between the centralized and the decentralized versions of workflow

patterns since when combined they form complex workflow processes. The first testing process represents the sequence pattern which, according to its nature, was combined with no other workflow pattern. And the second testing process represents the combination of the parallel split and the synchronization workflow patterns since they match, i.e., they are the AND split and joint.

An experiment consisted in the creation of an increasing number of process instances at a time, i.e., first one process instance, then 2, then 3 and so on up to 20 instances at a time. For each block of instances, the seconds elapsed from the first instance creation until the last task in terminate of the last process instance was measured. Thus, at certain moment there were several process instances running at a time.

The execution of a task consisted on waiting certain amount of simulation cycles. All tasks were standarized to 5 cycles and each cycle lasts 10 milliseconds, which means that the total amount of time for executing a task is 50 milliseconds. Notice that there was no central control on the passing time, i.e., each agent had to decide how much time had elapsed by its own.

For resembling the centralized approach, all tasks of the testing processes were assigned to one single agent who had to perform the whole work by itself. There were other threads along with the centralized version in order to allow all agents to operate under the same conditions.

For simulating the distributed approach, all tasks of the testing processes were completely distributed, i.e., one agent were assigned to perform only one task.

For testing our system with respect to sequential processes, a simple process were defined in which 42 tasks is sequence were put. In the decentralized case, each task was assigned to one single Personal Agent. And in the centralized case, each tasks was assigned to a single Personal Agent. For the centralized case, when processing a single instance, the elapsed time was of 96.275 seconds and for 20 instances the elapsed time was of 1778.313 seconds. For the decentralized case the elapsed time was of 79.811 seconds for 1 process instance and 434.517 seconds

for 20 instances. As can be appreciated in Fig. 4(a) even for 1 process instance, the decentralized approach overcomes the centralized one.

In other experiments, we tested our system with respect to parallel split and synchronization, The process used for this test consisted of a single task (thread) that splits into 40 different thread composed by one task each. Afterwards, all threads converges into another single one. In the centralized case, the duration of one instance execution was 102.411 seconds and for 20 instances it lasted 4068.367 seconds, i.e., over an hour for executing 20 instances. And the decentralized case lasted 47.31 seconds for one instance and 523.287 seconds for executing 20 instances of the same process. Figure 4(b) shows a comparison between these two approaches for workflow process execution. It is clear that a decentralized approach overcomes a centralized one in execution time.

The results presented in this chapter demonstrate that the distributed and decentralized execution of workflow processes outperforms a centralized architecture for the basic workflow patterns. It is clear that these results extrapolate to more complex patterns, which are combinations of the basic ones.

We think these results are clear indication that the decentralized architecture has advantages in terms of scalability, which is a very important issue for large organizations. Indeed, in the experiments we can see that some of the performance curves for the centralized version grew faster than linear.

The reason why the decentralized architecture outperforms the centralized approach, in terms of scalability, is that in the latter we are increasing the number of process instances over one single thread of execution (one Personal Agent), eventually saturating it; this single thread becomes a bottleneck and produces an increasing time overhead. That explains why, in the graphs presented, with an increasing number of instances, time increases not linearly, but worse (we did not investigate whether in the centralized case time was polynomial, exponential or other, but clearly is not linear).

6 Related Work

In general, other agent-based workflow architectures [10–16], emphasize the negotiation aspect of multiagent systems and their distributed nature. Thus, they proposed a distributed workflow system as well. However, they centralize the workflow process execution in one single agent (called Workflow Agent or Trigger Agent). In section 5, a comparison between a decentralized process execution and a centralized one was presented. Results demonstrate that a decentralized workflow process execution is better than a centralized one in terms of scalability. In addition, in those architecture there are several agent instantiation at run time under no control. In other words, they assume an environment with unlimited resource while in real environments that cannot be assumed. Our system does not makes that assumption since all agents are predefined to run at system start up. Snd besides, the required quantity of agents in our system is linear to the quantity of workers in the organization.

Compared to agent-enhanced approaches [17–19] our system architecture allows to automate behavior, i.e. agents can execute tasks on its own without human involvement, agents react to its environment, agents can adjust themselves, e.g., they can create new tasks or new routing depending on the circumstances, and finally, agents have high level features such as learning, negotiation, and planning [20]. In other words, an agent-based application has more benefits than an agent-enhanced workflow application since in the agent-enhanced workflow application agents' behavior is limited to the possibilities of the underlying WfMS.

Other architectures have been proposed for distributed workflow engines [21], distributed components of workflow patterns [22], and a distributed architecture in which components get communicated via ontological messages [23]. However, in traditional distributed system, all decisions, coordination and cooperation are hard-coded at design time. Additionally, the elements of these systems share a common goal. These are remarkable differences between this kind of systems and multiagent systems [24] since in the latter, the agents may not share common objectives and therefore they must act strategically, so that they can achieve the outcome they most prefer. In addition, agents are assumed to make decisions about what to do at run time (acting autonomously) while traditional distributed systems cannot.

7 Conclusions

We presented in this paper a multiagent-based architecture that supports decentralized workflow processes execution. The proposed solution for this purpose consisted in breaking down the workflow process execution and the process flow control into small execution units handled by distributed agents.

A prototype was developed in order to prove that the proposed solution for decentralized workflow process execution performs better than a centralized approach. Experiments were setup combining some of the workflow patterns and for different number of process instances. The results were, in the two experiments, conclusive since the decentralized approach outperforms the centralized version. These results prove that a decentralized approach for workflow process execution is more scalable than a centralized one.

As future work, we plan to include support for the remaining and more complex of the workflow patterns [9], allow agents to perform automated tasks without human intervention, e.g., when a task requires to incorporate information automaticaly from particular information sources or alarms to be triggering because of something happened in a legacy application.

Also, our decentralized proposal makes it possible to start a process execution and at some part continue it within another organization. This would be inter-organization workflow, which has great economic potential.

Acknowledgements

This work was supported by the Monterrey Tech's Research Grant CAT011.

References

1. Koulopoulos, T.M.: The Workflow Imperative. Van Nostrand Rainhold, New York, USA (1995)
2. Simon, A.R., Marion, W.: Workgroup Computing. Workflow, Groupware and Messaging. McGraw-Hill, New york, USA (1996)
3. Aguirre, J., Brena, R., Cantu, F.: Multiagent-based knowledge networks. Expert Systems with Applications **20** (2001) 65–75
4. Brena, R., Aguirre, J.L., Trevino, A.C.: Just-in-time information and knowledge: Agent technology for km bussiness process. In: Proceedings of the 2001 IEEE Conference on Systems, Man and Cybernetics, Tucson, Arizona, Octubre 7-10, IEEE Press (2001)
5. Ceballos, H., Brena, R.: Combining local and global access to ontologies in a multiagent system. Journal of Advanced Computational Intelligence and Intelligent Informatics **9** (2005) 5–12
6. Brena, R., Aguirre, J.L., Treviño, A.C.: Just-in-Time Information and Knowledge: Agent Technology for KM Bussines Process. In: Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference, Tucson, USA (2001)
7. P., C.A.M.: Decentralized Execution of Workflow Processes Using a Multiagent Architecture. Msc. in intelligent systems, Tecnológico de Monterrey, Campus Monterrey, Monterrey, México (2005)
8. Bellifemine, F., Poggi, A., Rimassa, G.: Jade - a fipa-compliant agent framework. In: Proceedings of PAAM99, London. (1999)
9. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patters. Technical Report FIT-TR-2002-02, Queensland University of Technology, Brisbane, Australia (2002) http://is.tm.tue.nl/research/patterns/.
10. Chang, J.W., Scott, C.T.: Agent-based Workflow: TRP Support Environment. In: Fifth International World Wide Web Conference, Paris, France (1996)
11. Jennings, N., Faratin, P., Johnson, M., Brien, P., Wiegand, M.: Using Intelligent Agents to Manage Business Processes. In: First International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96), London, UK (1996) 345–360
12. Manmin, X., Huaicheng, L.: Cooperative Software Agents for Workflow Management System. In: Fifth Asia-Pacific Conference on Communications and Fourth Optoelectronics and Communications Conference (APCC/OECC'99), Beijin, China (1999) 1063–1067
13. Yunlong, Z., Hongxin, L., Jinsong, X., Hongtao, W.: The Design of Cooperative Workflow Management Model Based on Agent. In: 31st International Conference on Technology of Object-Oriented Language and Systems, Nanjing, China (1999)
14. Gou, H., Huang, B., Liu, W., Ren, S., Li, Y.: An Agent-based Approach for Workflow Management. In: IEEE International Conference on Systems, Man, and Cybernetics, Nashville, USA (2000) 292–297
15. Botha, R.A., Eloff, J.H.P.: Access Control in Document-centric Workflow Systems – An Agent-based Approach. Computers & Security **20** (2001) 525–532

16. Stormer, H.: A Flexible Agent-based Workflow System. In: The 5th International Conference on Autonomous Agents, Montreal, Canada (2001)
17. Odgers, B., Shepherdson, J., Thompson, S.: Distributed Workflow Co-ordination by Proactive Software Agents. In: Intelligent Workflow and Process Management. The New Frontier for AI in Business IJCAI-99 Workshop, Stockholm, Sweden (1999)
18. Dogac, A., Tambag, Y., Tumer, A., Ezbiderli, M., Tatbul, N., Hamali, N., Icdem, C., Beeri, C.: A Workflow System through Cooperating Agents for Control and Document Flow over the Internet. In: CooplS '02: Proceedings of the 7th International Conference on Cooperative Information Systems, London, UK, Springer-Verlag (2000) 138–143
19. Hulaas, J.G., Stormer, H., Schonhoff, M.: ANAISoft: An Agent-based Architecture for Distributed Market-based Workflow Management. In: Software Agents and Workflows for Systems Interoperability workshop of the Sixth International Conference on CSCW in Design, London, Canada (2001)
20. Yan, Y., Maamar, Z., Shen, W.: Integration of Workflow and Agent Technology for Business Process Management. In: The Sixth International Conference on CSCW in Design, London, Canada (2001)
21. Ceri, S., Grefen, P., Sáchez, G.: WIDE - A Distributed Architecture for Workflow Management. In: IEEE 7th International Workshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications, Birmingham, UK (1997)
22. Ferreira, J.P., Ferreira, H., Toscano, C.: Distributed Workflow Management Enactment Engine. In: International Conference on Industrial Engineering and Production Management, Porto, Portugal (2003)
23. Blake, M.B.: Agent-Based Communication for Distributed Workflow Management using Jini Technologies. International Journal on Artificial Intelligence Tools **12** (2003)
24. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley and sons, LTD, Baffins Lane, England (2001)