FLOW-ORIENTED DEPLOYMENT OF A MULTI-AGENT POPULATION FOR DYNAMIC WORKFLOW ENACTMENT

A different view on how to use agents for workflow management

Sebastian Kanzow, Karim Djouani, Yacine Amirat
Laboratotory of Industrial Informatics and Automation(LIIA) ,University Paris 12, 120-122, rue P. Armangot, 92400 Vitry sur Seine, France

Keywords: workflow management, multi-agent systems

Abstract: In the virtual enterprise paradigm, workflow processes are shared between different businesses partners, leading to new requirements for workflow management applications. Several multi-agent systems have been proposed to cope with their inherently distributed nature. Most of those systems define agents as some kind of helper programs situated on (human) resource level, instantiated on some workflow participant's personal computer. We argue that this concept is not adequate and propose an approach to create and deploy agents on a virtual flow level, where one agent takes care of one workflow sub-process, instead of attaching one or more agents to an existing resource. Finally, we present a probabilistic classification approach to decide on the assignment of tasks to agents.

1 INTRODUCTION

Due to the introduction of electronic data processing in nearly every business domain, many companies have begun to use automated workflow management. Standards have been established for workflow description and information routing between the different departments of medium-sized and large business concerns (WfMC, 2002). Today, workflow management includes not only the different departments of the same company, but also information processing and task scheduling between various business partners, like suppliers, customers and sometimes even economic rivals. The virtual enterprise paradigm deals with the creation of a consortium of independent companies committed to the completion of a common product. This leads to new requirements for automated workflow concepts, by introducing the idea of confidentiality and security, as well as to the need for solutions of difficulties linked to the variety of operating software and the diversity of protocols and process description standards. The domain being inherently distributed, several research projects have been dealing with multi-agent approaches to overcome the drawback of centralized approaches (Jennings, 2000). Most often, those agents are attached to resources (human or other), their function is mainly communicative. For an example, see (Sacile *et al*, 2000) or (Chen, 2000).

2 RESOURCE BASED VS. PROCESS BASED AGENT DEPLOYMENT

Companies are in general organized around human resources and not along process flows, very much like in the beginning of industrialization, where manufacturing was organized in production cells. Of course, as to what concerns the manufacturing process, this concept has been abandoned for the sake of production lines, but it is still valid in most other domains, like administration, management and service. So the most natural way of introducing agents into workflow processes would be to consider agents as more or less automated helper programs, who are attached to physical resources, like desktop computers or production machines. This approach has the convenience of being similar to the

Kanzow S., Djouani K. and Amirat Y. (2004).
FLOW-ORIENTED DEPLOYMENT OF A MULTI-AGENT POPULATION FOR DYNAMIC WORKFLOW ENACTMENT - A different view on how to use agents for workflow management.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 510-514
DOI: 10.5220/0002623905100514

prevailing human role-actor model. On the other hand, the equivalence of resources and agents in a workflow management concept poses several problems. The first inconvenient is the lack of supervision between the execution of different tasks, which is needed to guarantee the overall progress of workflow execution. Secondly, scheduling gets quite complicated and needs a lot of inter-agent communication when every agent schedules locally its tasks, without consideration for the rest of the workflow, due to the fact that process execution logic is not necessarily linked to resource organization (Figure 1). Another problem is the reactivity in the case of disturbances, occurring during process execution, because one agent can't easily see the impact of a local perturbation on the rest of the process flow. Finally, most often tasks can be executed by a choice of several different resources, which means that supplementary inter-agent negotiation would be needed to assign a task.

In this paper, we suggest to rather consider agents as virtual entities on workflow sub-process level than as helper programs on resource level. The creation and the lifeline of agents should be transparent from the user's point of view. Starting from a standardized workflow description, the system should autonomously decide the number of agents it needs to guarantee optimal supervision of an ongoing process, while minimizing communication between different workflow participants.

3 RELATED WORKS

Multi-agent workflow enactment is a well-studied research domain. Most often, like for example in (Joeris, 2000), every task is coordinated by its own task coordination agent which interacts with related task agents by event passing. (Aalst, 2002) models hierarchical interorganizational workflows but doesn't deal with workflow enactment and automatic task assignment. (Dogac *et al*, 2000) describe another workflow concept based on communication between resource-situated agents and provide an authentication and certification process to enable agent communication *via* public networks. (Cichocki and Rusinkiewicz, 1997) propose their "migrating workflow" model with agents that are dynamically instantiated, following the execution of a workflow from host to host.

4 DISTRIBUTED WORKFLOWS

Work processes that are shared between several companies can't rely on centralized workflow engines for reasons of confidentiality. Owning the central server means being able to control the status of all connected participants, which is not desirable in commercial relationships between business rivals. Who should for example host the data server in the case of a workflow, which is distributed between two competing companies and their common supplier? For this reason, every participant hosts his own data and exchanges only limited information, which is needed for scheduling and global progress supervision.

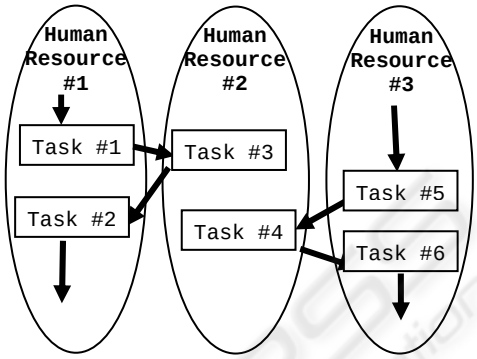

Figure 1: The logic of task execution does not necessarily follow physical resource layout (the arrows indicate precedence constraints between tasks)

4.1 Requirements for inter-organizational workflow management

We fixed four main corner stones to define the frame of a workflow engine adapted to the virtual enterprise concept:

The system has to be reactive and dynamic in order to be able to attenuate the global impact of local disturbances and adapt itself to changes during task execution.

Confidentiality must be respected in any case and only the minimal information necessary for optimal task execution may be transmitted to other participants.

The system should be organized in autonomous entities, in order to be able to deal with a variable number of resources (scalability).

It has to be universal, which means it must be independent of computer platforms and it should use a standardized language to describe workflow

processes in term of tasks, resources, security matters and information flow.
The workflow engine should take care of task assignment to resources and supervision of the progress of task execution, but does not necessarily include task execution itself, because it depends entirely on each participant's specialized knowledge.

4.2 Workflow description language

The workflow description is used to build dynamically a workflow execution environment. It contains information about tasks and their resources, estimated durations and precedence constraints. Resource constraints depend on the type of task, it can for example be a human resource or a capacity needed to accomplish the task. A precedence constraint means that one task's execution must be finished, before the next task's execution starts. Due to the workflow's distributed nature, its description is fragmented and hierarchical. Precedence constraints are only known locally, which means that a local workflow description contains references to immediately following or preceding tasks.
We use an XML-based workflow description language, containing the tags <role>, <task> and <successor> (Kanzow *et al*, 2003) The role tag contains task's to be accomplished by a resource belonging to a specified role, e.g. "accountancy" or "secretary". The task tag defines the name of a task, its estimated duration and can contain one or more successor tags, to define precedence constraints. An example:

```
<role name="accountancy">
    <task name="register_bill" duration="5">
        <successor name="send_product" role="shipper">
    </task>
</role>
<role name="shipper">
    <task name="send_product" duration="7" />
</role>
```

We're also working on the integration of more sophisticated routing elements, as defined in the eXchangeable Routing Language (XRL) (Aalst et al, 2001), introducing elements like <choice> and <sequence> to allow for more flexible workflow modelling.

5 FLOW-ORIENTED AGENTS

As shown in the preceding section, workflow fragments are defined locally using a resource-based view. Creating resource-oriented agents from this description would be straightforward, but to create flow-oriented agents, we first need to classify the tasks in order to decide which of them belong to the same workflow sub-process. The main decision criterion is the number of precedence constraints that link a task to its successors. Tasks should be assigned to agents in a way that:
minimizes the existence of inter-agent precedence constraints and
minimizes parallelism in task's belonging to the same agent.
Optimally, each agent deals with the execution of one task at a time and has only intra-agent precedence constraints (= Tasks that have precedence links only to other tasks belonging to the same agent). In reality though, there'll exist a certain number of inter-agent dependencies, because real-world workflow's sub-processes are in general interconnected.

5.1 Algorithmic approach for task assignment

We negotiate task assignment in three phases:
Creation of initial agents and iterative assignment of tasks to those agents
Decision process to assign tasks that are not clearly assigned during the first phase
Split and merge process of agents to ensure minimal temporal parallelism during execution of one agents' tasks
Workflows are executed in more or less uncertain environments. That means that most of the given parameters (like task execution duration) are in reality probability distributions instead of fixed values. During the first negotiation phase, we calculate the degree of probability for the assignment of some task to each of the agents. Initially, for every task that doesn't have any predecessor, an agent is created. Those newly created agents start an iterative assignment process. For every task T that is assigned to an agent A_x, it analyzes the successor list and calculates for each of the successors T_j a probability value:

$$P(T_i \in A_x) := \frac{\sum_{j \in pred.} C(T_j, T_i) P(T_j \in A_x)}{\sum_{j \in pred.} C(T_j, T_i)} \quad (1)$$

This is the sum of all precedence constraints linking the task to its predecessor tasks, multiplied by those tasks' probabilities to belong to the agent A_x, divided through the total number of the successor's precedence constraints. The agent with the highest probability value takes the successor task, but also remembers the probability values for the assignment of this task to other agents. That means, even after some agent has taken a task, there can still be a probability greater than 0 that it belongs to some other agent. If there's no clear winner, a second criterion is evaluated, in order to choose the solution where possible temporal parallelism during some agents' tasks execution is minimized. In other words, we assign a task T_i to an agent A in a way to minimize the risk of the same agent having to take care of several parallel task executions at a time. The criterion is calculated by the following equation:

$$P_{Ti\|\mathbf{P}i}(T_i \in A_x) := \frac{\sum_j P(T_j \in A_x) + 1}{n}, j \neq i, T_j \in \mathbf{P}_i \quad (2)$$

$\mathbf{P}_i$ is the set of all tasks that can possibly be executed at the same time as task T_i, which means that there are no direct or indirect precedence constraints between the members of $\mathbf{P}_i$ and T_i. n is the cardinality of $\mathbf{P}_i$ and the value $P(T_j \in A_x)$ has been calculated in equation (1).

The last phase is a "split or merge" negotiation, where agents can fusion or new agents can be created, based on the estimated occurrence of parallelism of one agent's tasks.

5.2 Workflow execution

When every task has been assigned to an agent, workflow execution can start. Every agent moves from resource to resource, along with the progress of its tasks' execution. This way, the current state of process execution will not be sent to some centralized server and thus the confidentiality of data is respected. Only one centralized network node is needed to keep the white pages of active agents up to date and to be able to react in the case of some agent's premature death, e.g. because of a system crash. On every resource, on arrival of some agent with its list of tasks to execute, the local scheduler agent, based on priority criteria specific to each domain, creates a local schedule. If several resources capable of executing some task do exist, the newly arrived agent can choose the resource as a function of the proposed schedules.

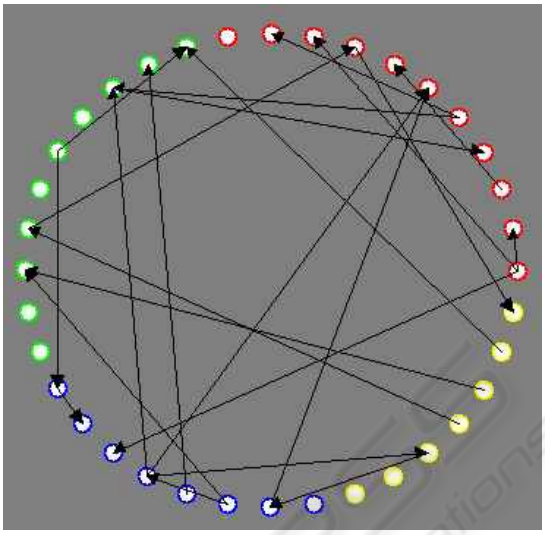

Figure 2: A randomly generated test scenario of 36 tasks with precedence constraints (indicated by arrows)

5.3 Benefits and difficulties of flow-oriented agent approach

The main advantage of our flow-oriented approach compared to other resource-oriented agent approaches lies in the fact that in our case workflow execution supervision is inherent. One of the most difficult points in distributed workflow management is to guarantee the liveliness of every running process by detecting tasks that have got stuck. A flow-oriented agent supervises closely its tasks, ideally one at a time, and can thus react immediately, should a disturbance occur. The agent's objective is clear: it has to reach the end of its sub-workflow. The main difficulty concerns the ability of an agent to move from one resource to another, the same multi-agent platform has to be installed on all workflow participants' computers. Furthermore, the agent platform must be able to encode every agent's state, to send it along some network and to recreate the same agent on a distant node. To summarize, we have a look at how our concept respects the four main requirements from section 2: reactivity and dynamicism are ensured, confidentiality is respected and the system is scalable, due to the agents' autonomous nature. The only one of our four corner stones that poses difficulties is the lack of independence of computer platforms, but with the gain of importance of multi-

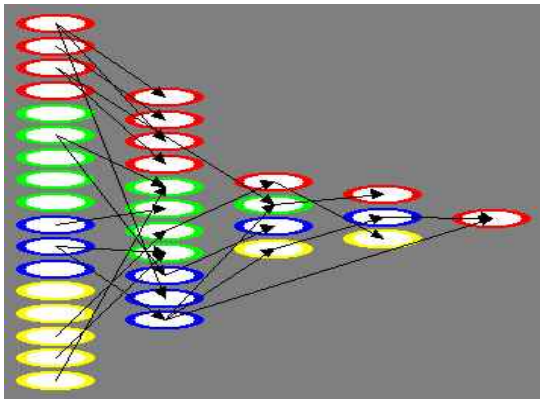

Figure 3: Graphical re-arrangement of tasks from Figure 2, based on precedence constraints

agent technologies and particularly the progress in agent mobility a standardized agent platform might soon emerge.

6 IMPLEMENTATION AND RESULTS

We implemented a multi-threaded agent testbed (Figure 2) with XML-based communication (Kanzow, 2004). Task configurations are generated automatically, using random distributions for numbers of tasks, resources and constraints. In the case of workflow scenarios with relatively few precedence constraints (less than 30% of tasks are linked through precedence constraints), tasks can easily be assigned to virtual agents using the simple negotiation algorithm described above (Figure 2 – Figure 4), but for more complex settings more investigation into the second and the third negotiation phase will be needed. We're working on a mathematical formulation of the criteria to minimize (inter-agent dependencies and degree of task parallelism). Nevertheless, the first results we obtained endorse our idea that workflows agents should logically correspond to sub-flows, rather than being situated on some workflow participant resource.

REFERENCES

van der Aalst, W.M.P., 2002, Inheritance of Interorganizational Workflows to Enable Business-to-Business, In *Electronic Commerce Research, Volume 2, Number 3, July 2002*

van der Aalst, W.M.P.; Verbeek, H.M.W.; Kumar, A., 2001, Verification of an XML/Petri-net based language for inter-organizational workflows, In *Proceedings of the 6th Informs Conference on Information Systems and Technology (CIST-2001)*

Chen,Q. and Dayal, U., 2000, Multi-Agent Cooperative Transactions for E-Commerce", In *Proceedings of the Fifth IFCIS Conference on Cooperative Information Systems (CoopIS'2000)*

Cichocki, A. and Rusinkiewicz, M., 1997, Migrating workflows, *Advances in Workflow Management Systems and Interoperability, Istanbul, Turkey*

Dogac, Beeri, Tumer *et al.*, 2000, The MariFlow Workflow Management System, In *Proceedings of the 16th International Conference on Data Engineering (ICDE-2000)*

Jennings, N. R. and Faratin, P. and Norman, T. J. and O'Brien, P. and Odgers, B. 2000, Autonomous Agents for Business Process Management, In Int. *Journal of Applied Artificial Intelligence14(2)*

Joeris, G., 2000, Decentralized and flexible workflow enactment based on task coordination agents, In *Second International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2000)*

Kanzow, S. 2004, A collection of VC++ classes to create a multi-agent testbed, *http://www.liia-paris12.net*

Kanzow S, Djouani K., Amirat Y., 2003, A framework for distributed workflow enactment using dynamic software agent generation, In *7th International Conference on Automation Technology, (AUTOMATION2003)*

Sacile, R. Montaldo, E. *et al*, 2000, Agent-based architectures for workflow management in manufacturing, In *International Conference on Advances in Infrastructure for Electronic Business, Science and Education on the Internet (SSGRR-2000)*

WfMC, 2002, Workflow Process Definition Interface -- XML Process Definition Language (XPDL), In *http://www.wfmc.org/standards/docs.htm*

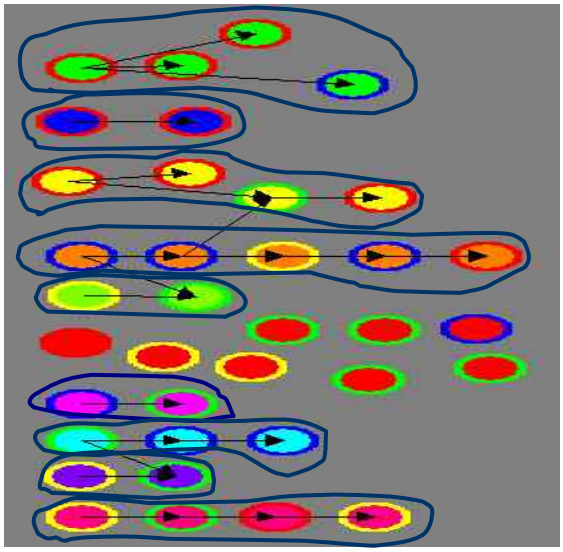

Figure 4: Result of task classification using the algorithm described in section 5.1