DYNAFLOW: AGENT-BASED DYNAMIC WORKFLOW MANAGEMENT FOR P2P ENVIRONMENTS

Adriana S. Vivacqua[1], Wallace A. Pinheiro[1], Ricardo Barros[1], Amanda S. de Mattos[1]
Nathalia M. Cianni[1], Pedro C. L. Monteiro Jr.[1], Rafael N. De Martino[1], Vinícius Marques[1]
Geraldo Xexéo[1,2], Jano M. de Souza[1,2] and Daniel Schneider[1]
[1]*COPPE/UFRJ, Graduate School of Engineering*
[2]*DCC-IM Dept. of Computer Science, Institute of Mathematics*
Federal University of Rio de Janeiro, Brazil

Keywords: Dynamic Workflows, CSCW, Agents.

Abstract: Many projects are characterized by their flexibility and high number of changes before a definitive solution is implemented. In these scenarios, the people involved may change, as may deadlines, assignments and roles. Traditional workflow systems don't handle dynamic scenarios well, as they are centralized and pre-defined at the start of the project. To address these problems, we propose an agent-based approach to dynamic workflow management, where participants may join or leave and roles may change depending on the situation.

1 INTRODUCTION

With the evolution of computing and networking technology and the growth of services and information available in the Internet, it has become possible to create mechanisms that enable users to collaborate and share information through their computers, regardless of their location.

Workflow Systems are popular tools for corporate collaboration, as they allow diverse task structure representation, organization, scheduling and distribution throughout the organization. Users can execute processes as a group, and keep business process knowledge inside the organization (Ellis, 1999).

The majority of existing Workflow Management Systems is limited, because they are based on client/server architectures and are fairly inflexible, not offering appropriate support to the dynamism found in real world situations, such as role or task changes when unexpected events happen (Weske, 1999).

With the technological advances and the adoption of distributed environments, more flexibility has also become necessary. In these environments, problems such as unexpected participant changes have to be managed on the fly. Furthermore, lengthy processes may be executed in this kind of environment, which also demands flexibility, as any changes during workflow execution need to be handled so as not to lose work already done.

Given this context, the goal of our research is to analyze the main problems inherent to the definition and execution of dynamic workflows, in environments characterized by flexibility and distribution, and formulate solutions to the problems found in these environments. The focus of our analysis is on decentralized, heterogeneous, dynamic agents that enable spontaneous group formation by physically dispersed participants. The goal is to add flexibility to workflows, making its structure more dynamic regarding the definition and execution as well as data and control distribution.

This paper is organized as follows: section 2 presents the DynaFlow architecture and section 3 a scenario. We finish with a discussion in section 4.

S. Vivacqua A., A. Pinheiro W., Barros R., S. de Mattos A., M. Cianni N., C. L. Monteiro Jr. P., N. De Martino R., Marques V., Xexéo G., M. de Souza J. and Schneider D. (2007).
DYNAFLOW: AGENT-BASED DYNAMIC WORKFLOW MANAGEMENT FOR P2P ENVIRONMENTS.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - SAIC*, pages 275-278
DOI: 10.5220/0002361102750278

2 DYNAFLOW

The proposed approach is a WFM that allows flexible definition, execution and management of workflows in distributed environment. Throughout this paper, a process should be understood as a set of sequential activities, where each activity has an associated competence (or skill).

DynaFlow includes both phases of the workflow management: definition, where the process activities and their execution order are described; and execution, where all activities are executed in the proper order. In this fashion, each member of a group can assume the role of workflow publisher or executor. The publisher will be responsible for the definition and publication of activities to the neighboring peers. Executors are those members that choose at least one of the available activities to execute.

As the publisher can receive more than one proposal (from different executor) to execute the same activity, it's necessary that each executor send a contract to the publisher, describing its proposal to execute each activity. Thus, the publisher will be able to analyze all contracts and choose the more appropriate one, selecting the best ones. The contract will be negotiated among publisher and executors, and must contain the activity to be performed, its price, execution time, current state, etc. The next sessions describe the system's regular flow of operation and the agents used to execute these operations.

2.1 System Architecture and Agent Description

DynaFlow defines two applications, one Publisher and one Executor. The publisher user defines manually the activities, their structure and flow and publishes them. From there on, all remaining actions will be executed autonomously by agents: contract receipt and analysis, activity republication, task result receipt, activity execution order definition, and so on. At the executor side, agents receive available activities, approved contracts and execution orders. There are also agents to send notifications to the publisher. These notifications can propose, confirm or finalize a contract (that is initialized manually).

Figure 1 shows the system architecture and inter-agent message flow. Dotted lines represents the activity flow, while the full line represents the contract flow (each contract with a status: proposed, approved, disapproved, confirmed, finished, or an execution order). It is important to mention that contract flow is not a broadcast, but direct communication, as both sender and receiver are known (unlike the activity flow, represented by the first and second steps).

The system was built on top of the COPPEER framework (Miranda and Xexeo, 2006). All communication relies on the COPPEER framework, using the *Negotiator* agent, which is responsible for taking the contract between publisher and executor.

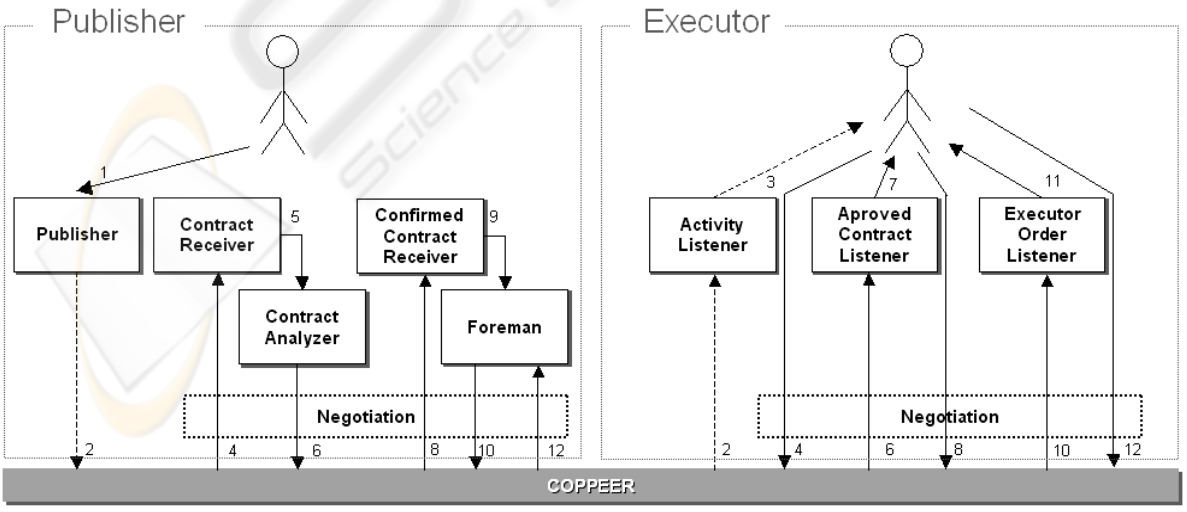

Figure 1: DynaFlow Architecture.

In Figure 1, the flow starts when a user uses the Publisher application to build a workflow (1). The Publisher agent broadcasts activities to other peers (2). When an activity is filtered by a peer's ActivityListener agent, the Executor application alerts its user about the incoming activity offer (3). If the user desires to execute some of the activities, he must create and send a contract proposal for each desired activity to the Publisher (4). A ContractReceiver agent waits for incoming proposals sent by executor peers (5). The ContractAnalyzer agent processes all contracts and sends each one back to its related executor, with the status (approved or disapproved) (6). An AprovedContractListener agent receives the approved contracts and delivers them to the user (7). The user confirms his interest at some of those contracts, sending them back to the Publisher, as confirmed (8). A ConfirmedContractReceiver agent waits for contract confirmations and sends these to the Foreman agent, which coordinates activity execution (9). The Foreman agent sends each contract to the associated executor (10). The ExecutorOrderListener agent waits for execution orders sent by the publisher. Each order is passed on to the user (11). On the executor side, when a user finishes an activity, he must signal its contract completion (12). On the publisher side, the Foreman agent receives that signal and sends the next contract to the related executor peer, if this activity depends on the previous one(12). Independent activities can be executed simultaneously. Steps 10, 11 and 12 are repeated for all activities.

3 TECHNICAL DETAILS

In this section we present the agent architecture and exception handling behavior used in DynaFlow. The initial version of DynaFlow has been implemented using COPPEER (Miranda and Xexeo, 2006), an agent based platform for the construction of distributed applications.

All agents introduced in the section 3.1 have been implemented, as well as the basic communication protocol. Agents are specialized, simple reflex agents, with basic rules that govern their behavior. Each agent has a specific task, as described in the previous section.

Publishers can inform others of a workflow description and activities needed, and executors can bid for contracts. At the moment, simple price-based selection is used, but implementation of more complex contract selection methods will be done. Exception handling is being implemented incrementally, to test each situation and not compromise already implemented steps. Thus far, we have implemented activity republishing, for those cases where no contracts were received, or when no confirmation was sent.

3.1 System Walkthrough

System operation begins when a user defines the activities of a workflow with the necessary competencies for their accomplishment. These are broadcast to other users.

To be notified about activities, users must define their competencies. Thus, when the competence of a published activity matches a user's competencies, that activity will show up as an executable activity. Once it has been notified about an available activity, the user can manifest its desire of executing that activity by sending a contract to its publisher. This contract defines the time and cost associated with activity execution. While these steps involve user action, the following are completely automated, being executed solely by the agents.

After the publisher analyzes the different contracts received and identifies the best ones, users are notified about the acceptance or rejection of their contracts. When a notification of approval is received, this reception is acknowledged and the agents stand by for the publisher's activity request, so that execution can be initiated. After receiving confirmations from the activity executors, the publisher sends out an execution order, so that the executor user for the first activity in the workflow may initiate execution. Upon receipt of an execution order, the executor initiates the activity and notifies the publisher when it is done. When the publisher receives a finalization notification, it sends out an execution order for the next activity in the sequence of the workflow. Non-dependent activities can be executed simultaneously. When all activities have been completed, the workflow is considered concluded.

4 DISCUSSION

One of the main problems with the first generation of the WFMs is that they had a predefined, immutable structure, which made it hard to adapt dynamic changes. It is not so easy for workflow administrators and users to foresee all situations

which should compose the workflow specification. In these cases, a feature to allow rapid changes in workflow structure when a change happens is needed (Weske, 1999). Environmental changes and technological advances are the main factors that lead to a need for dynamic workflow management (Han et al., 1998).

Dynamic workflow tools should enable operations on running workflow instances (Weske, 1999). Suspend and resume are needed to allow adaptation to changes. When a workflow is suspended, the system has to save the current state of that workflow so that it can be retrieved later through a resume operation. Another important feature is to provide resources to undo actions when a workflow instance doesn't work properly, in order to allow workflow users to return to a successful point of workflow execution.

WFMs are usually developed based in well defined processes, and that leads to inflexibility in current tools (Joeris, 1999). In order to support dynamic workflows, tools should deal with two types of flexibility: a priori and a posteriori. The former focuses on flexible behavior specification in order to achieve a behavior more precise and less restrictive in terms of flow advance. The latter focuses in flexibility for dynamic changes which should allow changes in the specification due to changes in the real world. In this case, it must be defined when and in what states these changes should be permitted to guarantee process consistency.

There are two types of changes in process flow: ad-hoc changes caused by an error, a rare event or a customer specific demand; and evolutionary changes that are the result of new strategic businesses, re-engineering efforts and permanent changes in external conditions (Aalst, et al, 1999). Workflow changes are not only changes in the process flow, and can also include participant and role changes, timing changes (e.g. activity start time), etc.

The possibility of negotiating task assignments and deadlines, and publishing revised workflows after execution has begun, coupled with the P2P architecture and scalability leads to new opportunities in for dynamic workflow control, breaking away from the strict structures normally found in traditional workflow systems.

More efficient strategies for contract negotiation, workflow definition and execution in P2P work environments can be studied, providing more flexibility and dynamicity to the process, without losing the control and coordination provided by workflow systems.

The current prototype restricts workflow creation to only one peer, which means that only one peer can be the publisher of a workflow. One possibility would be to allow peers to suggest alterations or improvements to the workflow, or even the group definition of a workflow. This would lead to extra research questions, such as how to identify peers that might share a workflow; how to define criteria for the selection of executors for a workflow; or the execution order of the workflow.

REFERENCES

Aalst, W.M.P. van der; Basten, T.; Verbeek, H.M.W.; Verkoulen,P.A.C.; Voorhoevoe, M., 1999. Adaptive Workflow On the Interplay Between Flexibility and Support. Proc. of the 1st Int. Conf. Ent. Information Systems, Vol 2, pages 353–360, Setubal, Portugal.

Ellis, C., 1999. Workflow Technology. In Beaudouin-Lafon, M. (ed.) Computer Supported Co-operative Work, England: John Wiley and Sons.

Georgakopoulos, D., 1995. An Overview of Workflow Management: From Process Modeling to Workflow Automation. Distributed and Parallel Databases, n.3, p. 119-153.

Han, Y.; Sheth, A; Bussler,C. A., 1998. Taxanomy of Adaptive Workflow Management Proceedings of the CSCW-98 Workshop Towards Adaptive Workflow Systems, Seattle, USA.

Jablonski, S., Bussler, C.,1996. Workflow Management. Modeling Concepts, Architecture and Implementation. London, Thomsom Computer Press.

Joeris, G., 1999. Defining Flexible Workflow Execution Behaviors in P. Dadam, M. Reicher (ed.) Enterprise-wide and Cross-enterprise Workflow Management - Concepts, Systems, Applications, GI Workshop Proceedings - Informatik' 99, Ulmer Informatik Berichte Nr. 99-07, University of Ulm.

Miranda, M.; Xexeo, G. B.; Souza, J. M, 2006. Building Tools for Emergent Design with COPPER. Proceedings of 10th Int. Conf. CSCWD, Nanjing, v. I. p. 550-555.

Weske, M., 1999. State-Based Modeling of Flexible Workflow Executions in Distributed Environments. Society for Design and Process Science Printed in the United States of America, Vol. 3, n. 2, p. 49-62.

Yan, J.; Yang, Y.; Raikundalia, K.G, 2006. SwinDeW ─ A p2p-based Decentralised Workflow Management System. In ASWEC'06, Proceedings of Australian Software Engineering Conference, pp.61-69.