Dynamic Workflow Management for P2P Environments Using Agents

Wallace A. Pinheiro[1,3], Adriana S. Vivacqua[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], and Jano M. de Souza[1,2]

[1] COPPE/UFRJ, Graduate School of Engineering
[2] DCC-IM, Dept. of Computer Science, Institute of Mathematics
Federal University of Rio de Janeiro, Brazil
PO Box 68.511 - ZIP code: 21945-970 – Rio de Janeiro, RJ – Brazil
[3] IME, Military Institute of Engineering
Pr. General Tibúrcio, 80 - ZIP code: 22290-270 - Rio de Janeiro, RJ – Brazil
{awallace, avivacqua, rbarros, amandasm, nathalia, calisto, rmartino, vgmarques, xexeo, jano}@cos.ufrj.br

Abstract. Traditional workflow systems don't handle dynamic scenarios well, as they are centralized and pre-defined at the start of the project. In this paper, we present a P2P framework to support dynamic workflows, which uses contracts to deal with unexpected changes in the flow of activities. DynaFlow is an agent based framework, where agents take action when exceptions occur. DynaFlow could provide adequate support for E-Science experiments mapped into workflows instances, with tasks operating in distributed environments and diverse types of computational resources and data.

Keywords: P2P Systems, Dynamic Workflows, Workflow Flexibility, Agents.

1 Introduction

In-silico experiments are scientific processes in which structured activities can be designed to address questions that arise in scientific problem-solving [1]. These experiments can be mapped to scientific workflows that automate these processes, managing various interconnected tools and large scale data in multiple data formats, distinct environments, algorithms, applications and services. E-Science areas can benefit from workflow technologies, data parallelism and distributed environments to minimize execution time and enable collaboration, regardless of locations.

In these environments, problems such as node failure or unexpected participant changes have to be managed on the fly, creating a need for more flexibility. Furthermore, lengthy processes may have to be executed and any changes during workflow execution need to be handled so as not to lose work already done.

In order to support dynamic workflows, management tools should deal with two types of flexibility: a priori and a posteriori. The first one focuses on flexible behavior specification in order to achieve a behavior more precise and less restrictive in terms

Y. Shi et al. (Eds.): ICCS 2007, Part III, LNCS 4489, pp. 253–256, 2007.

of flow advance. The second one enables changes in the specification. In this case, it must be defined when and in what states these changes should be allowed, to guarantee consistency of the experiment throughout its life cycle.

Centralized coordination causes problems such as vulnerability, loss of flexibility and no guarantee of availability. The adoption of peer-to-peer (P2P) technology enables the construction of a system that decentralizes workflow control and management [2], adopting a low coupling structure with no central data repository, to increase workflow flexibility. Some example of E-Science workflows systems are Taverna [3], Kepler [4] and GridOneD [5].

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. We adopt a P2P agent based environment, because they are decentralized, heterogeneous and dynamic. Besides, they enable spontaneous group formation by physically dispersed participants, leading to added flexibility in workflows. This paper is organized as follows: the next section presents the DynaFlow architecture and is followed by a brief discussion.

2 DynaFlow: Agents to Handle Workflows

DynaFlow is a peer-to-peer, agent based, framework to support dynamic workflows, which uses contracts to deal with unexpected changes in the flow of activities. Each peer can assume the role of workflow publisher or executor. The publisher peer will be responsible for the definition and publication of activities to the neighboring peers, and the executor peers are the peers that volunteer to execute at least one of the available activities. Each peer is supported by a group of agents that handles contracts and task assignment and execution, to enable dynamic adjustment of the system.

2.1 System Architecture

DynaFlow defines two applications built on top of the COPPEER framework [6], one Publisher and one Executor, each running on one peer. The following agents are implemented to control the workflow:

- Publisher – is the agent responsible for publishing workflows activities. This agent is started when a workflow has to be published or republished
- ActivityListener – is constantly waiting for new published activities. When it receives an activity that demands the same competence manifested by the executor, it inserts this activity on an activity list.
- Negotiation – the goal of this agent is to move contracts from executor to publisher and vice-versa.
- ContractReceiver – this agent receives contract proposals send by the Executor for the time established by the Chronometer Agent.
- Chronometer – controls system timeouts.
- ContractAnalyser – analyses contract proposals sent from the Executors. This agent can use several strategies to select which Executor will undertake an activity. For example, it can consider the minimum of time and cost.

- ApprovedContractListener – this agent receives approved contracts from the Publisher. It creates a list with the approved contracts. The Executor uses this list to confirm a contract to the Publisher.
- ConfirmedContractReceiver – this agent receives confirmed contracts (send by the Executor) and sends them to the Foreman Agent.
- Foreman – manages the execution orders. It sends the orders to Executors in the correct sequence and, when an activity depends of other activity, it waits the conclusion of its predecessor.
- ExecutionOrderListener – receives execution orders and shows to the Executor.

The publisher defines the activities, their structure and flow manually. 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 will 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.

2.2 Contract

DynaFlow uses contracts to establish rewards and punishments that can be converted into a measure of reputation. These can be used to handle issues such as change in activity specification or incomplete execution. In this case, upon fulfillment of a contract, a peer increases its reliability degree, while a peer that breaks a contract has its reliability reduced. Table 1 shows typical contract terms.

Table 1. Contract Terms

Contract Terms
Publisher Reputation Grade and *Executor Reputation Grade*
Number of Evaluations (received by the publisher and executor)
Approval Limit Date (for the publisher to accept the contract proposal made by the executor)
Execution Order Limit Date (for the publisher to order the task execution)
Signature Date
Execution Period (after this, the executor pays a delay punishment)
Execution Period Extension (after this, the contract is rescinded)
Task Cost (processing time, trustworthiness, etc)
Period of Result Availability (after this, the executor can discard the result)
Subtasks Delivery (flag that indicates if the subtasks will be delivered after concluded)
Task Description, Subtask Descriptions and Subtask Weights to the Task
Status Check Period (for the executor)
Delay Punishment Value (processing time, trustworthiness, etc)
Rescission Conditions and Punishment Value (processing time, trustworthiness, etc)

Reputation can be a fundamental factor to decide whether a peer is trustworthy or not. Reputation systems provide a summarized (perhaps imperfect) history of another peer's transactions. Users use this information to decide to what extent they should trust an unknown peer before they themselves have interacted with it [7]. The initial

calculation of a peer's reputation is based on criteria adopted by schools to evaluate students: calculating the arithmetic mean of grades received by their evaluators.

Thus, each executor peer, after it has participated of a workflow, is evaluated and receives a grade for its participation. A new average will be calculated whenever a new grade is received. The publisher peer receives a grade calculated from the grades given by the executors. Historical grades (grades given and received) are stored by the executor as well by the publisher, and are composed by the workflow identification and the peer grade.

3 Discussion and Future Work

The possibility of task assignment and renegotiation provides new opportunities for handling events in E-science workflows. Revising a workflow after execution has begun is important for dynamic workflow control.

More efficient structures to handle the contract and its negotiation need to be defined. A good definition of the relevant contract metadata (e.g., time to execution, reliability of results, time to provision of results, etc.) enables appropriate task distribution and workflow adjustment. The definition of rules to handle events is also very important: if a result comes in that is not what was expected, how should the rest of the workflow be handled? Should the process be aborted? Can alternative results be obtained? Can these results be verified by other sources? Should the experiment be rerun? These actions will depend on the situation, but need to be addressed.

References

1. Singh, M. P.; Vouk, M. A.: Scientific Workflows: Scientific Computing Meets Transactional Workflows. Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, Univ. Georgia, Athens, GA, USA; 1996, pp.SUPL28-34.
2. Fakasa, G.J.; Karakostasb, B., 2004. A peer to peer (P2P) architecture for dynamic workflow management. In Information and Software Technology, Vol. 46, No. 6, pp. 423-431.
3. Goble, C., Wroe, C., Stevens, R., and the myGrid consortium, "The myGrid Project: Services, Architecture and Demonstrator", Proceedings UK e-Science All Hands Meeting 2003 Editors - Simon J Cox, p. 595-603, 2003.
4. Kepler , Kepler: Distribueted Kepler Visited July 27, 2006 http://www.kepler-project.org/Wiki.jsp?page=DistributedKepler
5. Taylor, I , Shields, M. , Philip, R. 2002, GridOneD: Peer to Peer visualization using Triana: A Galaxy formation Test Case In *UK eScience All Hands Meeting*, September 2-4.
6. Miranda, M.; Xexeo, G. B.; Souza, J. M, 2006. Building Tools for Emergent Design with COPPER. Proceedings of 10th International Conference on Computer Supported Cooperative Work in Design, Nanjing, v. I. p. 550-555.
7. Marti, S., 2005. Trust And Reputation In Peer-To-Peer Networks. A dissertation submitted to the Department of Computer Science and the Committee on Graduate Studies of Stanford University.