Integration of Intelligent and Mobile Agents for E-commerce – A Research Agenda

M.T. Tu, F. Griffel, W. Lamersdorf *
Distributed Systems Group, Computer Science Department,
University of Hamburg, Germany
Vogt–Kölln–Str. 30, 22527 Hamburg, Germany
[tu|griffel|lamersd]@informatik.uni-hamburg.de

Abstract

The challenge of realistic E-commerce application scenarios makes an integration of both mobile and intelligent technology – which have been traditionally treated separately as a result of the diversity of research and development work on agent technology – appear essential. This paper introduces into some relevant technology issues related to this integration task which all still need substantial research efforts. It also gives some hints on corresponding work done (and published elsewhere) by the authors.

Keywords: intelligent and mobile agents, dynamic software components, interaction patterns, electronic commerce.

1 Introduction

One of the most obvious motivations for developing agent technology is to provide agents which are able to perform commercial transactions on behalf of the people launching them (see, e.g., [Gen97]). Using software agents – including both intelligent and mobile ones – in electronic commerce is attractive for several reasons: disburdening people from routine transactions, handling (gathering, selecting) great amounts of information in a given time, supporting mobile device users by asynchronous communication etc. (see [CGH+95]).

However, a consequence of the diversity of research and development work on agent technology is that a strong distinction between "mobile" and "intelligent agents" has emerged which can also be regarded as a distinction between *location autonomy* and *decision autonomy*. This simple de-facto classification of agents at present seems unfortunately inaccurate because the features of mobility and intelligence are obviously by no means mutually exclusive or even only contrary. And although the integration of these two qualities is certainly desired in order to face the challenging requirements of *realistic* agent application scenarios modeling

*This work is supported, in part, by grant no. La1061/1-2 from the German Research Council (Deutsche Forschungsgemeinschaft, DFG)

the complexity of real-world business processes, there have been up to now very few concrete approaches in this direction.

In view of this situattion, this paper is intended to identify and introduce into relevant aspects of this integration problem. In order to limit the size of the paper, these aspects are first presented as independent base technologies to be developed or advanced in near future. Then, the challenge of realizing automated negotiations as a concrete application area that can profit from the base technologies is discussed by means of an agent framework developed at the University of Hamburg.

Before going into the specific aspects, some general requirements arising from the integration problem should be mentioned: One of the main difficulties a practical approach has to deal with is that the incorporation of "intelligent" capabilities into mobile software agents can become very expensive because reasoning mechanisms, for instance, are usually much more complex than a few "go" and "select" commands coded in simple agents roaming the network nowadays, i.e. the agents can become literally too "fat" and consequently, their mobility is reduced. This problem is even more severe if the concrete purpose or task of the agent is not known ahead (at compilation time), in which case either many agents for different tasks or very general-purpose agents, which are likely even bigger in size, have to be built. In order to cope with this problem, some important requirements have to be imposed on a corresponding system design:

- Role-specific functionality: A mobile agent should not be loaded with every kind of available functionality or intelligent capability at the same time (as it is usually the case with complex AI systems or human beings), but should rather carry with him only the functionality required to fill out the actual *role(s)* assigned to him at a given time, for instance "seller" or "notary" in the context of electronic commerce.

- On-the-fly loading: Moreover, the functionality of an agent should be able to be loaded "on demand", i.e. at (or short before) the moment it is really needed.

- Flexible configuration: The agent's functionality should also be flexibly and dynamically configurable so that it can be reused in many similar, but differently constrained situations without having to replace its corresponding implementation.

These requirements can be regarded as boundary conditions that always need to be kept in mind when discussing about mobile agents with intelligent capabilities.

2 Development of Base Technologies

In this section, a selection of the technologies contributing to the development of mobile and intelligent agents which can participate in practical E-commerce scenarios is introduced.

2.1 Dynamic Software Components as Agent Building Blocks

First, in order to satisfy the requirements of role-specific functionality, on-the-fly loading and flexible configuration mentioned above, the agents that make up an application (or represent the application part of a (E-commerce) system cannot be built in a static way (i.e. compiled once out of all classes/functions that will be needed for the whole life-time of the agent), but have to consist of functional parts which can be "glued" together in a dynamic way. The *componentware* paradigm [Gri98] aims exactly at providing technical support for this kind of applications. The basic idea is really simple: Instead of generating a new application by writing the neccessary program code, compiling it (together with other existing source code), installing and starting the application, it should be *composed* out of active components – each of which encapsulates a (non-trivial) functionality that can be utilized through well-defined interfaces – which do not need to be recompiled but only eventually reconfigured[1].

In particular, so-called *plug-in mechanisms* can be used as a generic method to add, delete or exchange components of an application at run-time. In [TGML98, TSL99], a plug-in mechanism is described that allows for the (orthogonal) plugging of *distributed* and *mobile* components and is therefore appropriate to support the task of embedding arbitrary functionalities which can be some intelligent capabilities into mobile agents at run-time. Even the meta-feature of being pluggable (in either direction) can be acquired by a component or agent of the corresponding framework dynamically.

However, there are still a lot of open issues concerning component-based software to be investigated including, for example, complete and sound aggregation models, versioning and updating issues, secure (*sealed*) components and semantically rich interfaces [Gri98, GTZL99].

[1] Such a reconfiguration can even be performed automatically by means of *adaption* mechanisms [YS95, Gri98].

2.2 Integration of Agents, Business Objects and Workflows

In order to use agent technology to solve real-world business problems efficiently, it is of utmost importance to integrate this technology with other business-oriented technologies and corresponding software architectures that have been developed quite independently. Currently, Business Objects and Workflow Management are certainly two of the most relevant technologies to be considered for a seamless integration with agent technology, especially with mobile agents, since mobility and distribution in general entails a lot of additional (technical) possibilities as well as problems. For example, modeling business-relevant information as mobile agents carrying such data allows for more application-level oriented system designs comprising data that *actively* takes part in cooperative scenarios (self-controlled data flows) [Gri98]. Further, the actors within workflow systems may themselves be represented as agents allowing for decentralized workflows that may be adjusted in more flexible ways to changes of a company's internal organization and also allow better support for *inter-organzational* workflows [MLMJL96].

2.3 Policy and Rule Management

The characteristic features of *autonomy* (which implies the possibility to completely *delegate* certain tasks to the agents) and *interaction* make the agent programming paradigm on the one hand attractive for a wide range of distributed applications – in particular those related to E-commerce – but on the other hand also pose a considerable semantical risk potential, since the decisions delegated to an agent could bring about results that are not desired by its principal. Therefore, a generic approach of imposing rules on the behavior of (mobile) agents in order to reduce this kind of risk – without impairing the agents' basic decision logic and interaction ability – needs to be developed. In particular, it is important to enable the use of (dynamic) rules that influence the agents' *interaction* behavior.

At the Distributed Systems Group of University of Hamburg, a generic rule management system for distributed applications and mobile agents has been developed [TGML97, TGML99] which enables both a central and decentral management of currently four concrete rule types including *invariants, action rules* and *state transition rules*, as specified by the *Business Object Component Architecture* (BOCA) of the OMG [OMG98]. In particular, this rule system, which is formally based on a special form of predicate logic called LDNF, supports *policies* representing interaction rules (of different agents) that can be automatically unified to enforce a

common cooperation basis between the agents.

Two important issues which should then be investigated are first the extension of such a rule management system into a *rule-sensitive middleware* as a platform that provides *integrated* technical support for business applications including such services as event, transaction and domain management. Secondly, the elaboration of an *application framework* for rule-sensitive E-commerce applications is required to explore the potential of dynamic rules as a mechanism to provide "semantical add-on", i.e. to extend the semantics of applications at run-time.

2.4 Genetic Algorithms

GAs are inspired by the the evolution taking place in nature. Evolution in nature discloses an unmatched variety of different species each optimized for its ecological niche. This is achieved by very simple principles: selection together with reproduction, crossover and mutation. Selection means that only the fittest individuals survive. Reproduction is the ability to breed new individuals and mutations are deviations during this reproduction process. Crossover is the ability to take two individuals (parents) to breed one new individual that shares some attributes with each parent.

In GAs, the basic principles of evolution (reproduction, mutation and crossover) are used to create objects that are optimized for a certain function. To carry out this process, a set of objects (the *population*) is evaluated at discrete points in time (between the *generations*). Each individual has a certain probability to be taken into the next generation. This probability depends on its quality (*fitness*) measured by the function that should be optimized. The individual can be propagated into the next generation either unchanged (reproduction), mutated or as resulted from a crossover with another individual. The evolutionary approach can be used for the optimization of numerical problems (*Genetic Algorithms*) as well as for the automatic generation of programs (*Genetic Programming*). The main benefit of evolutionary techniques is that they make few assumptions about the environment they are applied to. All that is needed is the definition of the reproduction, mutation and crossover operations for the data structures used and a fitness function. Evolutionary techniques are very often used successfully in environments where little knowledge of their structure exists.

Therefore, GAs seem to be a very powerful approach to implement agents' tasks that require some kind of *adaptation* or flexible *strategy* in general. In [TWL99], an approach to implement GAs which can be utilized for automated negotiations

is presented. These GAs are based on Finite State Machines (FSM) which provide a greater expressiveness than linear data structures like vectors or tuples. However, substantially more research is required to ensure the quality and especially performance of this technology in real-world situations.

3 Automated Negotiations as a Sample Application Area

Each of the base technologies briefly introduced above is by itself an important research area that can make significant contributions to the goal of providing mobile intelligent agents which are capable of solving real-world E-commerce problems in an open distributed environment such as the Internet. For proof-of-concept purposes, however, it would be desirable to have an application area that is wide enough to profit from all these technologies and nevertheless poses a set of goals to be achieved which can be clearly defined. In our view, the task of enabling automated negotiations, especially in E-commerce scenarios, is exactly one of such application areas. Moreover, the challenge of automating negotiations can of course be considered an independent research goal that requires a firm conceptual foundation of its own [Rai82].

Basically, enabling software agents to perform automated negotiations requires that they be equipped with two capabilities: acting conforming to a public negotiation *protocol* and implementing a concrete *strategy* to achieve their private negotiation goals. Since the number of possible protocols and strategies is principally unlimited, it is important to be able to switch between different implementations of them in a flexible manner. Therefore, we have proposed a modular and very dynamic framework, in which the required negotiation capabilities can be subsequently loaded into a mobile agent or can be exchanged at run-time and which also allows explicit control of the mobility of an agent's components [TGML98, TSL99].

According to this framework, which is depicted in Figure 1, a negotiation enabled agent consists of three main components that can be dynamically aggregated (by means of generic plug-in mechanisms mentioned above): a *communication*, a *protocol* and a *strategy* module. Such an agent can make use of several third-party support services such as a *broker* and a *protocol engine* that helps coordinate and checking the compliance of the participants' negotiation actions [TLGL99].

However, there is still a lot to be undertaken to improve the practical usability

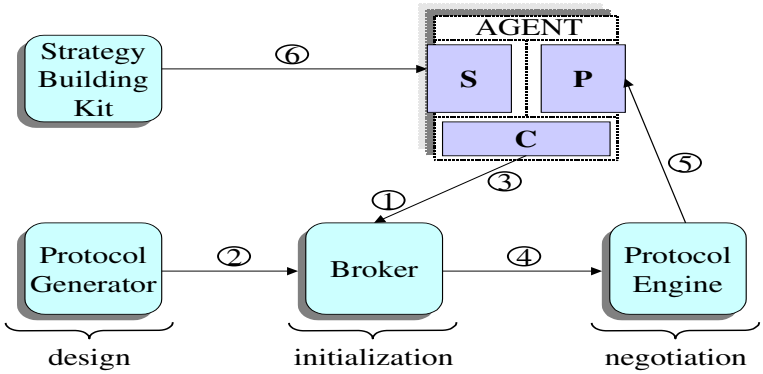

Figure 1: Structure of negotiation enabled agents and support services

of such a framework. Currently, an actor-based framework to develop distributed strategy modules is being implemented [TSL99] and the deployment of genetic algorithms for such modules is investigated by experimental work [TWL99].

4 Concluding Remarks

In this paper, we have briefly introduced several technology issues related to the goal of integrating mobile and intelligent agents to tackle the challenge of real-world E-commerce scenarios. These issues, which were divided into basic research topics such as componentware and rule management and more specific aspects related to automated negotiations, represent topical research areas which in our view bear a great potential for the practical deployment of agent technology in general and are therefore worth thorough investigation and further development.

References

[CGH+95] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, and G. Tsudik. Itinerant Agents for Mobile Computing. Technical Report RC 20010, IBM Research Division, T.J. Watson Research Center, 1995.

[Gen97] General Magic. Odyssey, 1997. `www.genmagic.com/agents/`.

[Gri98] F. Griffel. *Componentware.* dpunkt–Verlag, 1998.

[GTZL99] F. Griffel, M.T. Tu, C. Zirpins, and W. Lamersdorf. Towards Policy-mediated Component Composition. Technical report, Distributed Systems Group, University of Hamburg, 1999.

[MLMJL96] M. Merz, B. Liberman, K. Müller-Jones, and W. Lamersdorf. Interorganizational Workflow Management with Mobile Agents in COSM. In *Proc. PAAM96 Conference on the Pracical Application of Agents and Multiagent Systems, London*, 1996.

[OMG98] Business Object Component Architecture Revision 1.2. OMG Document 98-07-01. http://www.omg.org, 1998.

[Rai82] H. Raiffa. *The art and science of negotiation.* Harvard University Press, 1982.

[TGML97] M.T. Tu, F. Griffel, M. Merz, and W. Lamersdorf. Generic Policy Management for Open Service Markets. In H. König and K. Geihs, editors, *Proc. of the Int. Working Conference on Distributed Applications and Interoperable Systems (DAIS'97), Cottbus, Germany*, Cottbus, Germany, September 1997. Chapman & Hall.

[TGML98] M.T. Tu, F. Griffel, M. Merz, and W. Lamersdorf. A Plug-In Architecture Providing Dynamic Negotiation Capabilities for Mobile Agents. In K. Rothermel and F. Hohl, editors, *Proc. 2. Intl. Workshop on Mobile Agents, MA'98, Stuttgart.* Springer LNCS, September 1998.

[TGML99] M.T. Tu, F. Griffel, M. Merz, and W. Lamersdorf. Interaction-Oriented Rule Management for Mobile Agent Applications. In *Proc. of the Second Int. Working Conference on Distributed Applications and Interoperable Systems (DAIS'99).* Kluwer Academic Publisher, June 1999.

[TLGL99] M.T. Tu, C. Langmann, F. Griffel, and W. Lamersdorf. Dynamische Generierung von Protokollen zur Steuerung automatisierter Verhandlungen. In *Proc. 29. Jahrestagung der Gesellschaft für Informatik (Informatik'99).* Springer LNCS, 1999. (In German).

[TSL99] M.T. Tu, C. Seebode, and W. Lamersdorf. A Dynamic Negotiation Framework for Mobile Agents. In *Proc. 3. Intl. Symposium on Mobile Agents, MA'99, Palm Springs, California.* IEEE, October 1999.

[TWL99] M.T. Tu, E. Wolff, and W. Lamersdorf. Genetic Algorithms for Automated Negotiations: A FSM-based Application Approach. Technical report, Distributed Systems Group, University of Hamburg, 1999.

[YS95] D. M. Yellin and R. E. Strom. Collaboration Specifications and Component Adapters. Technical report, IBM T.J. Watson Research Center, 1995.