Electronic Activity Interchange EAI – a new way of B2B cooperation

Jarogniew Rykowski
The Poznan University of Economics, Department of Information Technology
Mansfelda 4, 60-854 Poznan, Poland
rykowski@kti.ae.poznan.pl

Abstract. In this paper, a new Electronic Activity Interchange EAI approach is proposed to support inter-business cooperation. This idea is based on using tele-activities, being pre-programmed piece of software that may be moved to certain locations and executed there. In contrast to traditional approaches, e.g., Electronic Data Interchange EDI, we consider both program and data flow rather than the information flow only. We propose a reasonable trade-off between a need for remote execution of activities owned and controlled by different business partners, and overall system security and efficiency. Our approach is based on imperative programming of activity code, and two basic activity classes: private, "untrusted" activities, executed with special care and with certain security restrictions, and public activities, being "trusted" code from the point of view of the local environment the activity is executed in. A balance between private and public activities may be defined by the business partners according to the specific situation and the business case.

1 Introduction

Traditional business is human-oriented - these are the humans who control business processes and information flow. However, modern business continuously searches for faster and more reliable ways of cooperation and management. Among others, two basic trends are of growing importance: mass applying of new information and telecommunication (IT) technologies, and tele-working. Modern IT technologies, such as personal communication devices, portable computers, satellite services, etc., changes the traditional ways of work. Due to the recent progress in computer hardware and software, a natural need arose to postpone some business processes, traditionally performed by the humans, to the computers. This trend is particularly visible inside an enterprise, starting from simple CRM and financial systems, through integrated, centrally managed production lines, to integrated, complex applications of SAP and similar systems. At the same time, fixed working place and

working hours, caused by fixed, stationary devices and timetables, are being replaced by the tele-work, in order to deal with business problems at-the-place and just-in-time. A tele-worker has certain freedom to choose the place, time, and form of his/her work, in order to optimally fulfill the ordered tasks.

Growing importance of high level of computerization inside an enterprise and the tele-working (Shen, Norie, 2004) are not, however, necessary related with high level of integration of computer systems belonging to different business partners involved in a common business activity. There are several reasons for that, mainly related with overall safety and security, and a need of unification of different hardware and software systems owned by the business partners. Recently proposed technologies, as for example Web Services and Semantic Web, seem to optimize the inter-business cooperation from the point of view of the information owner. As a result, only information availability is taken into consideration, while further data processing is left to the business partners, to be performed separately in local partners' systems. One may see an analogy of such fixed places and tools of information processing with fixed, traditional working places of humans. Extending this analogy, one may see a need for a tele-work of computer software, with programs sent to different remote places to be executed there in the name of the program owner.

In this paper, we propose to extend the trend of mass applying of tele-work observed for human-based business and inter-business cooperation to tele-computing and tele-activities. Similar to human tele-workers, tele-activities are movable software programs, executed remotely in the name of the program owners, to realize certain business activities. A tele-activity is a standalone, completed, pre-programmed piece of software that may be instructed to move itself to a certain location to be executed there. All the tele-activities owned and executed in the name of different business partners form the system of Electronic Activity Interchange EAI. Comparing EAI with traditional approaches to business cooperation, like for example Electronic Data Interchange EDI, one may note the fact that rather than considering only the data flow, EAI deals with both program and data flow.

At the first view, it looks like the EAI idea may be implemented, in a natural way, by applying the technology of software agents. However, current agent-based systems and scientific prototypes are not flexible enough to deal with mass business cooperation. Wide applying of the agent technology is mainly limited by the trade-off between overall system security and a need for remote execution of the "alien" (from the local system point of view) agent code. More security means more administrative restrictions on user-defined code and code mobility that is hardly acceptable by the agent owners. On the other hand, more freedom in agent development and execution means limited control of agents' behavior, and limited control of the ways the locally gathered information is processed by the agents. Unrestricted and out-of-control access to local computer systems cannot be accepted by most of the business partners, unless the partners are really mutually trusted. As a result, nowadays software agents are used either in closed, mutually trusted environments with restricted number of users, or for some well-defined, fixed purposes and application areas, e.g., agent-based stock exchange, auctions, production lines, etc.

The main goal of this paper is to propose a new agent-based technology for an efficient implementation of the EAI idea. We propose a reasonable trade-off between a need for remote execution of activities owned and controlled by different business partners and implemented by software agents, and overall system security and efficiency, both from the point of view of the "local" business partner, and the activity owners. Our approach is based on imperative programming of activity (i.e., agent) code, and two basic activity classes: private, "untrusted" activities, executed with special care and with certain restrictions, and public activities, being "trusted" code from the point of view of the local environment the activity is executed in. A balance between the private and the public activities may be defined by the business partners according to the specific situation and given business case.

The remainder of the paper is organized as follows. In Section 2, continuous evolution of inter-business cooperation is discussed, and the idea of applying telework for computer programs is justified. In Section 3, Electronic Activity Interchange EAI approach is presented, taking into account general system architecture, and two basic assumptions: user-defined distribution of EAI business activities, and different techniques for imperative programming of behavior of these activities, depending on security restrictions. Furthermore, some implementation issues are presented related with applying software agent technology within the scope of the Agent Computing Environment ACE. Section 4 concludes the paper and points some directions for the future work.

2 Continuous evolution of the ways of business cooperation

The ways of inter-business cooperation were continuously evolved, as technical and organizational possibilities grew. We may distinguish three basic stages of this evolution: cooperation based on direct communication among humans, with or without certain communication devices, cooperation based on remote, however, still human-based access to computer systems, and finally, cooperation based on direct connections among computer systems belonging to different business partners.

The first evolution step in facilitating inter-business contacts was taken by mass applying of telecommunication utilities. Direct human-human interaction was replaced by remote cooperation in human-device-human mode, e.g., a direct talk was replaced by a telephone conversation, traditional paper letters by e-mails and fax prints, etc. At the beginning, the humans were much more "movable" than the devices. Thus, the humans were forced to visit the physical places the devices were available in order to use these devices, e.g., a telephone box, a computer room, etc. With the recent progress in telecommunication and miniaturization, we observe a quite opposite trend of "moving" devices closer to humans, e.g., cordless and mobile phones, handheld computers with Internet connection, etc. Note that "moving closer" sometimes means "logically closer" rather than "physically", to mention remote access to computer systems, remote control systems, networking, etc. The final phase of this evolution step is to massively use personal, portable devices – mobile phones, handheld digital assistants, palmtops, etc. Note that the interesting trend for

improvement of this kind of devices is to put attention to different ways of the data transfer rather than simple human-2-human (i.e., voice) communication.

Mass applying of remote access to company's computer systems usually imposes some organizational changes of this company (Jagdev, 2001). However, as the business profits are high, such evolution is justified. Tele-work is an example of successful applying of modern telecommunication technologies and personal communication devices to traditional business processes (AMASE, 2005). From the tele-worker point of view, continuous computer&telecom support reduces the need for fixed working place and working hours. The work is performed at-the-place and just-in-time. Due to the on-line availability of all the information needed to complete the task does not matter the place and time, the worker efficiency is increasing. From the enterprise point of view, tele-work optimizes company's internal business processes and usage of the resources, including human staff.

Tele-work introduced a new quality for humans being distant representatives of a company. Even being physically and geographically away of the company, such representative is logically close to all the company's resources needed at the moment. Again, we may observe here the trend of "moving closer", especially if we take into account personal and portable communication devices. Note, however, that even the tele-work may speed up the process of uploading the information, the tele-working human staff must be physically present close to the information source.

The next evolution steps are forced by the need of replacement of the weakest entity in the inter-business communication process – the humans. As more and more business processes are automatized, more and more information is collected and stored by the computers rather than humans and traditional "paper" media. Thus, the communication process is extended to the computer-human-2-human-computer line. From the point of view of this process, humans at both sides of this process are additional "wrapping" activities. Human activity cannot be fully programmed, thus such activity is usually neither fixed nor fully repeatable. Human-provoked errors may lower the quality of the "wrapping" process to a great extent. Thus, from the point of view of the consumer of the information, eliminating human activity at the "source" side is justified. Such reduction may substantially improve not only the quality, but also the "bandwidth" of the communication. As a result, we observe mass applying of remote-access technologies, such as Web servers and server-side extensions, e.g., CGI scripts and servlets. Note that this trend is similar to the "moving closer" trend for personal and portable communication devices.

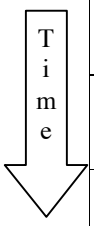
Mass applying of the direct access to remote computer systems by different business partners is limited by the conflict of interests of the "consumer" and the "producer" (i.e., the "source") of the information. The consumer controls the way of using the information, while the producer controls the way of accessing it. Business goals of the producer and the consumer are usually different. As long as both the producer and the consumer are humans, these goals are flexibly and efficiently negotiated. However, once a human is replaced by a computer, the negotiations are no more possible. Moreover, as the humans play active role in the communication process, the role of the automatized party is reduced. As a consequence, the responsibility for the activation and control of the communication process is shifted

to the still-human party. The automated party is then realized as a set of passive information sources, usually Web pages or services, waiting for the access. These information sources are polled by the humans in order to detect "crucial" information changes. In such a way it is quite hard to provide "active" information sources, automatically pushing the information changes to the consumers.

The automatized services are common for all the business partners and the business goals. Thus, usually a need arises to perform some additional data processing at the "consumer" side (Bonnet, 2001), to deal with given business case, situation, partner requirements, etc. Such processing is restricted by at least three reasons. First, client-side information processing may substantially increase network traffic in comparison with the server-side computations. Second, the just-collected information maybe useless; however, one has to complete the whole processing in order to detect that fact. For example, polling for a new business offer requires periodical access to all the offers, among them only a few are finally taken into consideration. Third, some business partners are not interested in providing "raw" information to be processed at the client-side, with limited control of the information owner. For example, a hospital will not provide outside a full list of its patients in order to compute some statistical data only.

As long as the humans control the way of information exchange, direct communication between computers is reserved for local usage only, inside an enterprise. However, with a continuous need for faster and more reliable business contacts, external computer-2-computer cooperation becomes a must. So far, such cooperation is quite rare, and typical information flow is brokered by human staff, mainly due to security problems and architectural differences of the computer networks belonging to the cooperating business partners. What is also quite important is the lack of common standards for data exchange – format, ways of communication, knowledge and ontology representation, etc. There are several attempts to improve the speed and the quality of the inter-business cooperation, to mention Web Services, Semantic Web, and several proposals for systems realizing the idea of a Virtual Enterprise (Oprea, 2003; Petri, 2003). All these proposals are based on common standards for global information exchange, resolving the problem of hardware/software heterogeneity. However, similar to the human-based remote access to the remotely-stored information described above, there is usually a business conflict between an owner of a service and the consumer of the information to be accessed via this service. The service owner tends to offer a fixed and long-term service common for everybody, passively waiting for incoming requests. The service consumer expects a personalized information source, actively informing about "critical" information changes, with no need for additional data processing at the client-side.

Table 1 summarizes the evolution of inter-business cooperation methods, taking into account both human- and computer-based cooperation. Note that, according to the above discussion, the evolution of the computer-2-computer direct cooperation seems to be non-completed yet, in comparison with the human-based cooperation methods. Thus, the question arises how to improve direct communication among computers belonging to different business partners?



H-2-H	*H-2-C*	*C-2-C*
Direct meeting	Local access	Data transfer under human management, computer-human-computer data flow
Remote contact	Remote access	Direct computer-computer data flow, client-side data processing
Tele-work	Personal communication devices, remote access and control of programs	**?**

Table 1. Evolution of inter-business cooperation methods

Prior to answering the above question, we have to discuss in more details current approaches to inter-business information exchange, based on the idea of Electronic Data Interchange EDI. Typical EDI process of an information exchange is the following. First, the information is produced by a business partner and made available using an e-mail or a Web page. Second, the letter is sent to (the page is requested by) another business partner. Third, the just-obtained information is locally processed, including necessary wrapping (both format and contents, if needed), ontology/schema verification and adjustment, etc. Once processed, the information is available for the business partner.

The above information-exchange process implies common ontology/schema and fixed interfaces (formats) at both sides, in order to (1) establish a connection, (2) formulate a request understandable for the information owner, (3) generate a response understandable to the consumer. While more and more business partners are involved in the process, more difficult is to maintain common ontology and interfaces. As one cannot foresee all the possible expectations of all the possible business partners, no individualization is possible of the above communication steps at the server-side. Thus, all the post-processing and content/format wrapping must be performed by the business partners locally. Moreover, inter-business communication usually involves several human-based administration activities, as with the automatic services one cannot fix all the reactions to all the possible errors and unexpected cases. As a consequence, EDI-based cooperation is usually provided only for long-term, stable business connections. There is no support for ad-hoc and temporal cooperation, becoming crucial for nowadays business. In addition, connection maintenance and control is partially manual and thus expensive and error-prone.

As already stated, the process of information exchange is usually realized in passive and polling way. First, the information passively waits to be accessed by anyone interested in this information. As long as there is no request for the information, the information is useless. Thus, a need arose for continuous polling for changes of the information, usually generating huge network traffic from the place the information is accessible to the place the information is about to be used. The polling process needs substantial data treatment at the client-side, at least for the comparison and detection of information changes. Such treatment is usually time-

and resource-consuming, and often needs a human-based coordination and maintenance.

One may see an analogy of the above-described inter-business communication process – fixed information sources, business partners, data format and ontology, local information processing – to the traditional, fixed, local working places of humans. Extending this analogy, one may see a need for applying the idea of tele-work, being natural consequence of modern IT technologies, to automatic cooperation of computer systems belonging to different business partners. Tele-working computer programs are sent to different remote places, to be executed at-the-place in the name of the program owners, similar to tele-working human staff.

3 Electronic Activity Interchange

In this section we propose to extend the trend of mass applying of tele-work observed for human-based business and inter-business cooperation to tele-computing and tele-activities. As already mentioned in the Introduction, similar to human tele-workers, tele-activities are movable software programs, executed remotely in the name of the program owners, to realize certain business activities. A tele-activity (an activity for short) is a standalone, completed, pre-programmed piece of software that may be instructed to move itself to a certain location and execute there. Each activity carries its business state and an authority of its owner – a human or another activity. The state, together with the information gathered at the execution place, may be used by the activity to undertake some business processes. All the activities owned and executed in the name of different business partners form the system of Electronic Activity Interchange EAI.

Below basic strategies are presented for: (1) activity distribution and maintenance, and (2) activity programming. The basic assumption of the EAI approach is that these are the activity owners who have full control over the place an activity is migrated to, and over detailed activity behavior at-the-place. Thus, the presented strategies are owner-oriented, however, taking into account generic security and safety requirements.

First we discuss the *global strategy for effective distribution and maintenance of the activities*. We assume that there are three basic classes of the hosts an activity may be sent to and executed: private hosts, generic network hosts, and server-side hosts. According to these host classes, we distinguish three basic activity pools: client-side pool, composed of the hosts controlled by the activity owners (i.e., business partners), middle-side pool, composed by some general-usage hosts, and source-side pool, composed by the hosts controlled by the service owners (i.e., business partners offering some services and access to business information). The pools are characterized by different methods for migrating, storing, searching for, and executing the activities. Below, a general characteristic is given of each pool, together with a description of purpose and functionality of sample activities belonging to these pools.

A functionality of a host from the source-side pool is optimized towards reliable and efficient access to selected data sources, from the point of view of the

information owner (i.e., given business party). Activities operating in source-side hosts are usually owned by the information owner. For security reasons, storing and executing "alien" activities belonging to external business partners is substantially limited. A typical source-side host is reduced to a set of gateways, able to standardize an access to the data source(s) connected, with limited support for public telecommunication facilities (WWW access, SMS/MMS/e-mail asynchronous messaging, etc.), the latter described in more details later in this section. The gateways are equipped with several mechanisms supporting efficient, parallel, multi-user access to the data sources, as for example cache memories, proxies, synchronizers, semaphores, locks, query optimizers and serializers, etc.

Accessing activities from source-side hosts is similar to accessing public Web servers and services. The difference is the activities provide some additional communication, wrapping and brokering functionality, requested by the business partners, as well as some uniformity of the external access to several information sources. However, nevertheless the business partners have limited control over source-side activities – usually such activities are used as "black boxes", with limited possibilities of individualization of their behavior as well as the mode of operation.

Hosts from the middle-side pool are located in arbitrary chosen parts of the global network. In contrast to the source-side pool, middle-side hosts store and execute activities belonging to different business partners. A typical task list for the activities covers: brokering among source-side and private activities, wrapping and formatting messages exchanged by the population of activities, providing access via different telecommunication means and protocols. A stress is put on efficient access to the activities by the humans, using popular telecommunication channels and standards (WWW/WAP, SMS/MMS, e-mail, etc.). Activities executed in the scope of the middle-side pool are usually devoted to the tasks related with network-side monitoring – comparing information and detecting changes that are "interesting" for the activity owners. As already stated, what is "interesting" is programmed by the activity owner in the activity code.

Architecture and usage of a host from the client-side pool strongly depends on technical and communicational possibilities of an end-user hardware/software the activity owner posses at the moment. Private activities may be executed for example in the scope of a stationary PC, mobile equipment (a PDA, a notebook, or even an intelligent mobile phone). It is up to the activity owner to locate his/her activities either in a host from the middle-side pool, or in the private (i.e., client-side) host. In the first case, the network traffic may be substantially reduced; however, remotely executed user activities are less secured (from the user point of view) and less efficient (mainly because of additional security checks). In the second case, all the user activities are executed in a trusted (still, only from the activity owner point of view) environment, however, a lot of information must be transferred among distributed hosts.

For the activities executed at a portable/mobile device, a stress is put on fast and user-friendly activity-to-human communication. The technical capabilities of the device strongly limit the possibilities of executing the activities (small memory, limited battery time, difficult management, etc.). Thus, usually only a few private

activities are located in a mobile host capable of performing some simple tasks, as for example final formatting of an alert message, filtering incoming messages, generating sound alerts, etc.

As already mentioned above, a stress is put on efficient communication among the activities, and between an activity and its human owner, the latter mainly for the administration and management purposes. To the first goal, XML-based messaging is applied. Choosing the detailed mode of communication, i.e., communication standard/language (e.g., SOAP), knowledge representation and exchange method (e.g., KQML, RDF), ontology synchronization (e.g., OWL), etc., depends on the activity owner. Due to the large offer of standards and protocols that may be used, a more detailed discussion on the methods for activity-2-activity communication is out-of-the-scope of this paper. As for the activity-2-human cooperation, we propose to use standard telecommunication facilities, based on natural-language conversation, and Web-like access, based on predefined pages with different forms, menus, etc. To facilitate the goal of efficient communication with activities, we propose to provide some ready-to-use activities specialized for a conversation via given telecommunication channel (Rykowski, 2005C). Such a communication activity stands for a wrapper between an activity and human-owned communication device/software, like for example mobile phone or an Internet browser. The communication activities are usually located in middle-side hosts, to be effectively used by large population of human users (Rykowski, Juszkiewicz, 2003).

3.2 Programming activity behavior

Similar to the real world, longer is the distance from the place of origin to the place of execution, less an activity is "trusted". Locally (i.e., in a host owned by the activity owner) executed, an activity is treated as trusted code, as it is developed by the exclusive owner of the local environment. Remotely executed, an activity is usually treated as an alien code, thus additional security checking should be applied prior and during the execution of this activity. Such checking substantially reduces privacy and execution efficiency, resulting in a trade-off between overall system security (from the local point of view) and privacy/efficiency (from the activity owner point of view). Setting up the balance of this trade-off, one may use several techniques for code development and execution, among others skeleton-based, declarative, and imperative programming are the most popular ones.

Skeleton-based approach is based on some predefined pieces of code being "patterns" for automatic creation of the programs to be executed in the name of the program owners. This approach maybe used in the case the overall system security is much more important than user privacy and code individualization. The main disadvantage of the skeleton-based approach is that the total software functionality must be known in advance, before the development of the skeletons. Thus, this approach is unacceptable for ad-hoc and temporal business cooperation (at least). The second above-mentioned programming technique – declarative programming – is usually based on a specialized programming language, originated from logical programming, artificial intelligence, goal-oriented programming (e.g., database programming languages), etc. Even if declarative programming is much more

flexible than program skeletons, we think that it still cannot be applied for unrestricted user-defined design of the EAI activities. First, note that each declaration is automatically changed at run-time to an equivalent piece of software code, imperatively programmed by the system designers. Thus, similar to the skeleton-based technique, all the possible declarations (and thus total system functionality) must be known in advance. Each user has a choice of using or not certain declarations, however, he/she is not able to fully program details of the activity behavior. Second, providing run-time "compilation" of a declaratively-programmed activity certainly limits overall efficiency, both from the system (less throughput) and the activity owner (greater execution time, more resources consumed – memory, CPU, etc.) points of view.

Due to the following restrictions of the skeleton-based and declarative programming techniques, we propose to use imperative programming for setting up activity behavior, directly by the activity owners. In order to provide reasonable level of portability (migration) of the activities, and reasonable level of overall system security, we propose to apply a standard activity interface, and two primary programming techniques: interpretation connected with run-time code inspection for "untrusted" activities, and compilation for the "trusted" ones.

The main problem related with unrestricted usage of imperative code is limited system security. Remotely executed, imperatively programmed activities are treated as an "alien" code, potentially dangerous for the local environment. Such anxiety may be justified by insufficient level of code verification, or simply by pure psychological reaction of local system administrators. Even if from the "technical" point of view several security mechanisms are applied for the external code verification (i.e., code encryption and signing by digital certificates, built-in security verification for the compilers and kernels of the operating systems, etc.), the psychological fear maybe a serious obstacle for wide acceptance of user-defined, not-known in advance activities. However, note that similar problem has been already successfully resolved in the domain of the operating systems, by introducing two basic programming techniques: a shell language, used for example for preparing batch programs and desktop icons, and different programming languages, used for design of the application programs, further compiled to "executables". Most operating-system users are entitled only to manipulate shell scripts, and only some (usually system administrators) are able to install and control executables. Note, however, that the ordinary users are able to use given executables, even if such users have no possibility to change them. In contrast, the administrators rarely execute the applications, even if such users have full control over the application code, location, invocation parameters, etc.

Similar to the above division, we propose two basic techniques of setting up safe and secured activity behavior: the dedicated shell language, and full compilation. The shell language is used for programming mobile, remotely executed, user-defined activities (Rykowski, 2003). The language is based on the XML standard, and its syntax computational power is similar to the widely known shell programming languages. Note that we were not able to adapt any existing XML query language, as well as any generic XML transformation language, as these languages are

specialized for node processing, generating a set of XML nodes as a result of processing of queries/other nodes. Instead, we adopted typical shell syntax, adjusting it to the framework of the XML documents. There are also some security restrictions for the execution of XML-encoded activity code. First, total execution time is limited (depending on the activity owner and current system load). Thus, never-ending loops and procedures are no more executed forever. Second, total number of activity's internal variables is restricted. Thus, any activity is not able to block the system due to continuous, still growing memory demands. Note that none of current shell programming languages/interpreters is equipped with such run-time checking.

The EAI idea is being implemented in the scope of the Agent Computing Environment ACE project. ACE system consists of a distributed set of Agent Servers, characterized by different functionality of system agents, and different communication means. Taking into account a specialization of an Agent Server for certain tasks and localizations, we may distinguish three basic classes of Agent Servers: source-side servers, connected directly to data sources and located at the same host (local area network) as a given data source connected, middle-side servers located in the network, not directly linked with any particular data source, and client-side servers for personal usage of a given agent owner. The servers are connected to each other, and they are able to transfer the agents among each other as well as remotely execute the agents previously transferred to a given place. More information about the ACE framework may be found in (Rykowski, 2006B; Rykowski, 2005D). Several ACE applications are now being tested in a real environment, with commercial services and different users. Early proposals – individualized bank access and telecom information services – proved the usefulness of the idea of using user-defined software agents for mass personalization purposes. Other our proposals cover e-office support (Rykowski, 2005B), individual, targeted marketing in supermarkets, support for temporal and ad-hoc Virtual Enterprises (Rykowski, 2006A), and many others.

4 Conclusions

Summarizing, the main goal of this paper was to propose a new agent-based technology for an efficient implementation of the Electronic Activity Interchange EAI idea. This idea is based on applying tele-activities to support inter-business contacts, with an analogy to tele-work of humans. A tele-activity is a standalone, completed, pre-programmed piece of software that may be instructed to move itself to a certain location to be executed there. Comparing EAI activities with traditional approaches to business cooperation, like for example Electronic Data Interchange EDI, we consider not only the data flow, but also the software (program) exchange and remote execution. We propose a reasonable trade-off between a need for remote execution of activities owned and controlled by different business partners and implemented by software agents, and overall system security and efficiency, both from the point of view of the "local" business partner, and the activity owners. Our approach is based on imperative programming of activity (i.e., agent) code, and two

basic activity classes: private, "untrusted" activities, executed with special care and with certain restrictions, and public activities, being "trusted" code from the point of view of the local environment the activity is executed in. A balance between private and public activities may be defined by the business partners according to the current situation, business case, temporal and ad-hoc requirements, etc.

The EAI approach makes it possible to define an individual, distributed brokerage for existing information sources, as for example Web Services, and thus to personalize the behavior of the services to the maximum extent. In today's applications, the personalization level is restricted, due to potential service complexity and decreasing level of the overall system security. Our EAI idea fills this gap, as we allow to use user-defined, however safe code in different places of the network.

To our best knowledge, there is not a single proposal up to now to use imperative, mobile, user-defined software agents for supporting inter-business relations. Thus, we cannot deeply compare here our approach with a similar work. However, in (Rykowski, 2004) we included a comparison of the EAI/ACE approach with several strategies used for some existing agent-based systems, proposed for different reasons and purposes. A discussion on choosing either imperative or declarative way of programming the agents may be found in (Rykowski, 2006B), with similar conclusions to the ones stated in Section 3 of this paper.

References

1. AMASE: Agent-based Mobile Access to Information Services, ACTS homepage, http://www.cordis.lu/infowin/acts/analysys/products/thematic/agents/ch3/amase.htm.

2. M. Bonett, Personalization of Web Services, Opportunities and Challenges, 2001, Ariadne **Issue 28**, http://www.ariadne.ac.uk /issue28/personalization/intro.html.

3. H.S. Jagdev, K.D. Thoben, Anatomy of enterprise collaborations, International Journal of Production Planning and Control, **12**(5), 2001, pp. 437-451.

4. Oprea M., The Agent-Based Virtual Enterprise, Journal of Economy Informatics, **Vol. 3/1**, 2003, pp. 15-20.

5. Petrie C., Bussler C., Service Agents and Virtual Enterprises: a Survey, IEEE Internet Computing, **Jul.-Aug. 2003**, pp. 2-12.

6. J.Rykowski, A Juszkiewicz., Personalization of Information Delivery by the Use of Agents, IADIS Int. Conf. WWW/Internet 2003, Algarve, Portugal, 2003, pp. 1056-1059.

7. J.Rykowski, W. Cellary, Virtual Web Services - Application of Software Agents to Personalization of Web Services, 6th Int. Conference on Electronic Commerce ICEC 2004, Delft (The Netherlands), 2004 ; ACM Publishers; pp. 409-418.

8. J.Rykowski, Using software agents to personalize access to e-Offices, in *Proceedings of the 5th IFIP Conference on e-Commerce, e-Business, and e-Government I3E'2005*, October 2005 (B), Poznan, Poland, pp. 95-110.

9. J.Rykowski, Using software agents to personalize natural-language access to Internet services in a chatterbot manner, *2nd International Conference Language And Technology L&T'05, Poznan, Poland*, April 2005 (C).

10. J.Rykowski, ACE Agents - mass personalized software assistance, in Multi Agent Systems and Applications, eds. J.G. Carbonell et al., Lecture Notes in Artificial Intelligence 3690, ISSN 0302-9743, 2005 (D), pp. 587-591.

11. J, Rykowski, *Information management by software agents in ad-hoc virtual enterprises, to appear in "Knowledge and Technology Management in Virtual Organizations: Issues, Trends, Opportunities and Solutions",* eds. G.D. Putnik, M. M. Cucha, (2006 (A)).

12. J, Rykowski, Management of information changes by the use of software agents, Cybernetics and Systems, Taylor & Francis Publishing, Philadelphia (US), ISSN 0196-9722, **vol. 37**, no 2-3, March-May 2006, pp. 229-260, IF=0,768.

13. W Shen, D.H, Norie, An agent-based approach for information and knowledge sharing in manufacturing enterprise networks, Int. J. Networking and Virtual Organizations, **Vol. 2/2**, 2004, pp.173-190.