Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 25 – 29

ISBN 978-83-60810-14-9
ISSN 1896-7094

Adaptation of E xtended Polymorphic Self-Slimming Agent Model Into e-Sourcing Platform

Konrad Fuks, Arkadiusz Kawa, Waldemar Wieczerzycki
Poznan University of Economics, al. Niepodległości 10,
60-967 Poznań, Poland
Email: {konrad.fuks, arkadiusz.kawa, w.wieczerzycki}@ae.poznan.pl

***Abstract*—There are two main contributions of the work presented in this paper. First, extended PSA agent model is described. It is based on two new properties of so called *bootstrap agent*: multiplication and parallelism. Second contribution is application of extended PSA model to e-sourcing platform. This duo (e-sourcing and extended PSA model) shows enterprises can significantly increase resources acquisition and potential suppliers search.**

I. Introduction

DURING last several years, the lowly, back-end sourcing process has been transformed into a strategic resource. Sourcing is now seen not only as a strategic player in the value chain, but as a major driver in the extended supply chain. To enhance their global operations, firms are seeking dynamic supply chain partnerships to increase the speed and intensity of their response to changes in customer demand and to lower costs and reduce risks.[4]

Traditionally, sourcing involves a number of communication mediums to facilitate all business processes between various parties. These include the use of mail, phone, fax, EDI and, more recently, email and the Internet. The current advances in information and communication technology have revolutionized procurement to turn it into a mechanism that enables diverse and geographically disperse companies to create alliances to meet a new form of Internet-oriented consumer demand.[2] Analysts believe that enormous cost savings and efficiencies can be achieved through the utilization of the electronic sourcing, a component of the B2B model. According to the Prime Consulting Group, global firms are optimistic on the level of savings that can be achieved through full implementation of e-sourcing strategies. The potential for savings is tremendous.[5] For instance, General Electric reports that it believes that the firm has saved over $US 10 billion annually through its e-sourcing activities.[4]

However, there are some limitations of e-sourcing. Firstly, there are thousands globally operating e-marketplaces, each with its own range of transaction mechanisms and additional facilities. The increasing number makes it impossible to browse them all by humans. Secondly, there are also differences between the requirements of a prospective participant for buying purposes and those organizations seeking participation as suppliers.[6] Thirdly, the environment of e-marketplaces is constantly changing—some e-marketplaces occasionally give much better offers than others, growing competitive and demand's changeability make the best offer finding difficult.

Those inconveniences can be satisfied by combining e-marketplace with agent technology.

Agent technology has been one of the most prominent and attractive technologies in computer science in the last decade. Software agents and their standards are being developed all the time by a lot of enterprises. Agents can be useful for e-supply chain configuration and management based on e-markets[1], which is strictly related to three particular properties of software agents. Firstly, agents are autonomous. A user can activate them, leave in the network and disconnect, provided agent mission is well defined. Secondly, such agent can be highly mobile. Agents enable dynamic supply chain configuration and reconfiguration in different environments. Thirdly, agents can be very intelligent, thus they are able to support efficient supply chain configuration. In case of e-procurement, agent technology is able to support efficient e-marketplaces, including partners who offer one another the best cooperation possibilities and conditions at a given time. There is a possibility to create temporal and dynamic supply chains aimed at executing a single transaction. This creates previously unknown opportunities for enterprises which can avoid stable supply chains, built on the basis of long-term contracts with rarely replaced business partners. [7][3]

The structure of the paper is as follows.

Section II presents the PSA agent model which is based on multi-dimensional versioning and segmentation of agent code, as well as on distribution of agent code and data over the network. The role of proxy agent and bootstrap agent is detailed. Finally, two extensions of the primary PSA model are discussed.

Section III shows application of the extended PSA model to e-sourcing platform. At the beginning environment of the approach is described. In the second part of this section features and functions of agents as well as exemplification of the approach are presented.

Section IV concludes the paper and shows the future work.

II. Extended PSA Agent Model

According to the basic PSA (Polymorphic Self-Slimming Agent) model, an agent is composed of a sort of agent head, called *bootstrap agent*, and agent body.[7] Bootstrap agent is relatively small, thus it can be highly mobile. Its goal is to

move over the network and to decide whether a newly visited environment is potentially interested, taking into account agent mission defined by the user. If it is, then the bootstrap agent recognizes the specificity of the environment, afterwards it communicates with so called *proxy agent*, residing on the origin computer (i.e. a computer in which agent has been created by the user), asking for sending to the bootstrap agent an appropriate part of the code of agent body. Bootstrap agent communicates directly with the proxy agent using the messages in an agent-communication-language (ACL).[8][9]

It may happen that after some time, if the results of agent (i.e. bootstrap agent extended by agent body) activity are satisfactory, the bootstrap agent again communicates the proxy agent asking for the next part of agent body. This situation will be explained later.

If agent mission in the currently visited environment is completed then the agent body is removed from the environment and the bootstrap agent migrates to a new environment or it returns to the origin computer in order to merge with the proxy agent.

For the sake of platform independence and security, we assume that the bootstrap agent is interpreted by a visited environment. In other words the bootstrap agent is a source code, rather than partially compiled (pre-compiled) code.

There are four basic functions provided by the stationary proxy agent:

- It serves as a communication channel between the user and bootstrap agent: it knows current location of bootstrap agent and can influence the path of its movement, as well as tasks performed by the bootstrap agent.
- It encompasses all variants of the agent code that model the variability of agent behavior. Depending on what environment is currently visited by the bootstrap agent, and according to bootstrap agent demands, proxy agent sends a relevant variant of agent code directly to the bootstrap agent, thus enriching it with the skills required in this new environment and, as a consequence, enabling it to continue the global agent mission. Notice, that code transmission redundancy is avoided, since the unnecessary code (not relevant agent variants) remains together with the stationary component.
- It assembles data items that are sent to it directly from the bootstrap agent, extended by a proper agent variant, which are not useful for a mobile code, while could be interesting to the user. The data assembled is stored in so called *knowledge repository*.
- Whenever required, the proxy agent responds to the user who can ask about mission results and data already collected (e.g. a percentage of data initially required). If the user is satisfied with the amount of data already available, proxy agent presents the data to the user and finishes its execution (together with bootstrap agent).

To summarize the aforementioned discussion, one can easily determine the behavior and functions of a moving component (code). When it migrates to next network node, only bootstrap agent is transmitted, while the agent variant is just automatically removed from the previous node. There is no need to carry it together with the bootstrap agent, since it is highly probable that a new environment requires different agent variant. When migration ends, bootstrap agent checks its specificity, and sends a request for a corresponding agent variant transmission, directly to the proxy agent. When code is completed, the mobile agent component restarts its execution.

During the inspection of consecutive network nodes only the information that enriches the intelligence of a moving agent is integrated with agent bootstrap (thus it can slightly grow over the time). Pieces of information that does not increase the agent intelligence are sent by the mobile component of the agent directly to the stationary part, in order to be stored in the data repository managed by it. This agents feature, namely getting rid of unnecessary data, is called self-slimming.

Now we focus on possibilities of the PSA agent versioning, i.e. on the content of the proxy agent that is always ready to select a relevant piece of agent code according to a demand of the bootstrap agent. We distinguish three orthogonal dimensions of agent versioning:

- agent segmentation,
- environmental versioning,
- platform versioning.

Agent segmentation. Typically agent mission can be achieved by performing a sequence of relatively autonomous tasks (or stages). Thus the agent code is divided into so called segments corresponding to consecutive tasks that have to be realized. If one task is finished successfully, then the next one can be initiated. Thus, next segment of agent code is received from the proxy agent and agent execution switches to this new segment. Depending on whether agent behavior is sequential or iterative, the previous segment is automatically deleted (in the former case) or it remains in the execution environment (in the latter case).

To summarize, this versioning dimension, namely segmentation, models multi-stage nature of PSA agents.

Environmental versioning. Bootstrap agent can visit different environments providing different services, e.g. e-marketplaces, auction services. Moreover, every service can be implemented in a different way. For example, the e-marketplace can be implemented as a web-site or database application. Thus, depending on the specificity of environment being visited different version of agent segment is required.

In other words, the proxy agent keeps and manages sets of versions for each agent segment and takes care on their consistency, i.e. knows which versions can “go together”.

To summarize, this versioning dimension, namely environmental, models polymorphic nature of PSA agents.

Platform versioning. Finally, every version of every agent segment is available in potentially many variants which are implemented for a particular target environment (i.e. for a particular hardware which runs agent environment: processor, operating system, network communication protocols etc.). There is one especial variant for each agent segment version, that is in a source form. It is sent to the environments which for the security reasons do not accept pre-com-

piled code. Besides this particular variant, the proxy agent keeps potentially unlimited number of partially compiled variants. If the environment accepts this sort of code, then the proxy agent delivers a variant matching hardware and system software parameters of this environment.

To summarize, this versioning dimension, namely platform, models platform independent nature of PSA agents.

Now we present two important extensions of the primary PSA model, which in our opinion, could be very useful in e-commerce applications (cf. section 3).

Firstly, it may happen that the chain of environments (nodes) visited and examined by a bootstrap agent is very long, before the relevant one is found. In order to increase the efficiency of examination, we allow for a single PSA software agent to have more than one copy of a bootstrap agent, behaving in the same way. In other words, it is possible for the proxy agent to send in parallel, say, *n* bootstrap agents addressed to different network locations. When allowed by visiting environments, they start to examine their specificity in asynchronous way. Their common goal is to find, as fast as possible, the most appropriate environment (e.g. e-marketplace) to fulfill the agent mission.

When all *n* bootstrap agents report the end of their activity and send back results, now it is up to the proxy agent to decide which one of the visited nodes will be examined by the PSA agent (i.e. consecutive variants of code segments). Then, only one of bootstrap agents (augmented by a code delivered to it on demand) continues its execution, while all the remaining *n-1* bootstrap agents automatically "dye" without further actions.

Secondly, we must allow the situation when more than one bootstrap agent reports to the proxy agent a success of examination, i.e. more than one potentially interesting network sites have been found. If they are just e-marketplaces, then perhaps instead of short visiting, they require continuous monitoring of announced offers, 24 hours per day in a long time period. To correctly deal with this situation, we assume that a single PSA agent may have many, so called *twin instances*, residing in different environments. They are fed with necessary code in-the-fly, by the same proxy agent, since they are just "twins" acting in a corresponding way, according to the same code repository kept by a single, shared proxy agent.

Twin instances are to some extent autonomous, thus in general they may work asynchronously. In this case the common proxy agent plays the role of a pure communication mean between them. It may happen, however, that there is a need for synchronicity between twin instances. For example, one instance can switch into next, say, third code segment only if the other one has already finished successfully the execution of first segment. In this case, bootstrap agent is responsible for coordination of instances execution, providing basic synchronization mechanisms.

III. Application of Extended PSA Agent Model to E-Sourcing Platform

Information systems that support e- sourcing can be classified into four major segments: buy-side applications, sell-side applications, content applications and e-marketplace applications.[4]

In this section we focus on the last segment and we present the adaptation of Polymorphic Self-Slimming Agent Model into it. This new proposal expands the present opportunities offered by e-marketplaces.

Environment of the new approach

- The foundation of resources/suppliers searching is the *sourcing cluster*.
- *Sourcing cluster* (type of business cluster) is a group of enterprises which are looking for the same type of resources (e.g. steel, plastic, packaging, transportation, etc.) created within specific e-marketplace. All sourcing cluster data is stored within e-marketplace database.
- E-marketplaces are based on service-oriented architecture (SOA) which provides system interoperability.
- E-marketplaces coopetite[1] with each other in order to provide wider offers range for their customers.
- Each e-marketplace can have its own company priority managing mechanism. Each company has its own priority level participating on every e-marketplace (*p-level*) which can depend on: purchase/sell volume, purchase/sell value, length of e-marketplace participation period, company size, annual income, country of origin, etc. Characteristics of the prioritization mechanism can be individually set on each e-marketplace.
- E-sourcing platform is a set of all potential interoperable e-marketplaces that can exchange information. Each e-marketplace receives interoperability level status. We distinguished three levels: full interoperability (F), quasi interoperability (Q), base interoperability (B).
- *F status* of e-marketplace must provide: offer browsing, *sourcing cluster* creation and management, agent negotiation, contract signing, partner evaluating (p-level changing), message exchange, and external systems (i.e. ERP, WMS) integration.
- *Q status* of e-marketplace must provide: offer browsing (obligatory), any combination of other *F status* features.
- *B status* of e-marketplace must provide: offer browsing.
- Each enterprise is represented by group of software agents (cf. section II).
- Only enterprises (their software agent representatives) can create or join sourcing clusters.
- Type of the resource in each sourcing cluster is the same.
- The more enterprises in sourcing cluster the bigger possibility to find needed resources.
- At the same time each e-marketplace is searched

[1] Coopetition is a neologism which matches cooperating and competition. Examples of coopetition include Apple and Microsoft building closer ties on software development and the cooperation between Peugeot and Toyota on a new city car for Europe in 2005.

only by one enterprise representative (software agent) from each sourcing cluster.

- E-marketplaces can be accessed by an arbitrary number of software agents (e.g. representing many suppliers) which work independently, or collaborate with other software agents.[3]
- The facilitates offered by the e-marketplace may extend to full execution, including financial and logistical services. [6]

Features and functions of agents

The main c haracteristics of the agents used in our approach are:

- robust ness (even when one or more individuals fail, the group of agents can still perform its tasks),
- self-organization (agents need relatively little supervision or top-down control because they are autonomous; agents have capabilities of task selection, prioritization, goal-directed behavior, decision-making without human intervention)
- flexib ility (agents can quickly adapt to a changing environment because they are very intelligent and reasoning).

Additionally, agents are mobile, so they can relocate their execution onto different places in the network.

We distinguished some groups of features and functions which are characteristic for a proxy and bootstrap agents active on e-sourcing platform.

- Features and functions of proxy agent:
  - It represents particular entities (e.g. purchaser, supplier, logistics service provider) which operate on e-marketplace.
  - It communicates not only with bootstrap agents, but with delegating users (human or IT system of enterprise (e.g. ERP system)).
  - It acts according to guidelines and entitlement granted by a delegating user.
  - It informs about the progress and the results of performed tasks.
  - It solves problems that appear during offers searching.
  - It co opetites with agents from competitive companies.
- Features and functions of bootstrap agent
  - It browses offers available at e-marketplaces.
  - It is responsible for checking and comparing offers of products and services.
  - It notifies proxy agent when a new offer appears and informs about the progress of the realization of the offer.
  - It is responsible for negotiation of terms of cooperation with the agent presenting the offer.
  - It is engaged in business contract monitoring and finally signing it.
  - It is obliged to inform proxy agent about problems and failures.

Exemplification of the approach

Let 's imagine that one company is not satisfied with its past relations with suppliers. Additionally company' transaction volume is still rising and customers want more customized products. Problems with suppliers mainly relate to shipment delays and rigid volumes of orders. This situation forces company to look for new partners to preserve its market position. Task of finding new, reliable suppliers is very difficult and time consuming, especially when company acts alone. Combination of e-sourcing and agent technology (extended PSA model) can inconceivably boost up this process. To start looking for new suppliers our company creates profile on one of available e-marketplaces. Each profile on our future e-marketplace is equipped with group of PSA agents (cf. section II). Then company finds a sourcing cluster on the e-marketplace. If there is no such cluster company creates one, but it still acts alone in searching potential suppliers. When company wants to be part of a bigger sourcing cluster it delegates proxy agent to find one on other e-marketplaces. Proxy agents send bootstrap agents to look through the e-sourcing platform (all e-marketplaces) to find proper sourcing cluster for the company. Company can determine conditions that sourcing cluster must meet (i.e. number of enterprises, size of enterprises, their locations, etc.). Of course for bigger scope of potential suppliers searching, company can participate in many clusters on many e-marketplaces.

But how these sourcing clusters help to find suppliers? Each company in sourcing cluster has its own proxy agent (company representative) and related group of bootstrap agents. Proxy agents of each company associated in the sourcing cluster delegate its bootstrap agents to search for precisely described resources (quantity, quality, price, shipping terms, etc.). Searching conditions (Cn) are determined individually by each company. Cn is a set of i conditions. $Cn = \{Cn_1,...,Cn_i\}$. Individual condition (Cn_i) can have precise value (number or text) or can be described by text list or number range. When in Cn are conditions that can be negotiated or omitted proxy agents creates additional Cnx set of information where Cnx_i represents possible state of condition i. Each Cnx_i can have one of four states: *ob* for obligatory, *om* for omitted, *n* for negotiable and *n/om* for negotiable or omitted. The last state allows proxy agent to decide what to do with specific condition when resource is founded. Additionally Cn set is supplemented with the value of company's total demand for the resource (D).

Proxy agent delegates bootstrap agents to look for particular resource that meets Cn and satisfies D. We distinguished four possible scenarios of the resource searching process:

1. All Cn are met and D can be fully satisfied.
2. All Cn are met but D can be partially satisfied.
3. Not all Cn are met but D can be fully satisfied.
4. Not all Cn are met and D can be partially satisfied.

In cases (1) and (2) proxy agent just sends additional parts of the bootstrap agent source code that is responsible for next sourcing phase (i.e. contract signing). Scenarios (3) and (4) allow proxy agent to change not met Cn_i, to start negotiation phase or just to discard the supply offer. All above operations depend on e-marketplace status (F, Q or B). When proper features are supported by e-marketplace bootstrap agent source code can be extended with: contract signing procedures, negotiation procedures or message exchange procedures. If D is fully satisfied proxy agent in-

forms other proxy agents within its sourcing cluster about founded resource. Information includes basic conditions for the purchase (i.e. supplier location, resource description, shipping information, currency, needed documents, etc.) and updated volume of the resource. All company-specific conditions are omitted in this information.

Above exemplification contains only mechanism for resource searching on external to sourcing cluster e-marketplaces. Implementation of the mechanism and its extension to purchasing, negotiation and contract signing are foundations for future work (cf. section IV).

IV. Conclusions and Future Work

To summarize, research results reported in this paper mainly relate to well-known li mitations of e-sourcing, i.e. difficulties and restrictions in browsing e-marketplaces by humans, differences between requirements of a potential participant for buying purposes and those organizations seeking participation as suppliers, growing competitive and demands changeability at e-marketplaces that make the best offer finding difficult.

As proposed and explained in the paper, those inconveniences can be solved by software platforms implemented according to very promising and challenging for business – agent technology.

Future work will concern mainly further extensions of PSA model (cf. section II) towards its application in the area of widely understood e-sourcing activities, as well as prototyping experiences, based on the assumptions referred in the previous section (cf. section III). Authors find it also very interesting extending presented model with swarm intelligence algorithms (i.e. ant colony optimization (ACO), particle swarm optimization (PSO)).

References

[1]. Denkena B., Zwick M., Woelk P.O., *Multiagent-Based process Planning and Scheduling in Context of Supply Chains*, 1st International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2003, Springer-Verlag, LNAI 2003.

[2]. Folinas, V. Manthau, M. Sigala, M. Vlachopoulou, *E-volution of a supply chain: cases and best practices*, Internet Research, vol. 14, no. 4, 2004.

[3]. Fuks K., Kawa A., Wieczerzycki W., *Dynamic Configuration and Management of e-Supply Chains Based on Internet Public Registries Visited by Clusters of Software Agents*, 3rd International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2007, Springer-Verlag, LNAI 2007.

[4]. Hawking P., Stein A., Wyld D. C., Foster S., *E-Procurement: Is the Ugly Duckling Actually a Swan Down Under?,* Asia Pacific Journal of Marketing and Logistics, Vol. 16, No. 1, 2004.

[5]. Prime Consulting Group, *Reverse Auctions: An Industry White Paper,* International Housewares Association Reverse Auction Task Force, 2002.

[6]. Stockdale R., Standing C., *A framework for the selection of electronic marketplaces: a content analysis approach*, Internet Research: Electronic Networking Applications and Policy, Vol. 12, No. 3, 2002.

[7]. Wieczerzycki W., *Polymorphic Agent Clusters – The Concept to Design Multi-agent Environments Supporting Business Activities*, 2nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2005, Springer-Verlag, LNAI 2005.

[8]. http://www.fipa.org/specs/fipaSC00001L/, *Foundation for Intelligent Physical Agents*, FIPA Abstract Architecture Specification.

[9]. http://www.fipa.org/specs/fipa00061/, *Foundation for Intelligent Physical Agents*, FIPA ACL Message Structure Specification.