INFORMATICA, 2002, Vol. 13, No. 4, 441–454

An Agent-Based Architecture for Customer Services Management and Product Search

Thomas HEIBERG
BEKK Consulting AS
Skur 39, Vippetangen, N-0150 Oslo, Norway
e-mail: thomas.heiberg@bekk.no

Mihhail MATSKIN
Department of Computer and Information Science
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
e-mail: mihhail.matskin@idi.ntnu.no

Jøran PEDERSEN
Kelkoo Norge, Parkveien 60, N-0254 Oslo, Norway
e-mail: joran.pedersen@kelkoo.com

Received: August 2002

Abstract. The amount of products and services available over the Internet increases significantly and it soon becomes beyond users ability to analyze and compare them. At the same time the number of potential customers available via the Internet also increases dramatically and starts to be beyond the service providers ability to perform efficient targeted marketing. A possible way for relaxing the above-mentioned limitations could be in usage of electronic assistants, both for customers and providers. Such assistants may serve as mediators for commercial Internet-based activity. Software agents could play role of such mediators representing customers and providers in the network. In this paper we present our experience and a solution to using agent technology in customer services management for mobile users. The solution is intended to increase granularity and personalization in targeted advertising while ensuring customer privacy. The proposed solution has been implemented in a prototype system for providing services for users of mobile devices.

Key words: software agents, customer services, WAP devices, product search.

1. Introduction

Amount of products and services available now over the Internet increases dramatically and becomes beyond users ability to analyze them efficiently. At the same time the number of potential customers available via the Internet also increases significantly and starts to be beyond the product/service providers ability to perform targeted marketing. Additional important aspect of utilization of Internet-based services is their recent availability

via mobile devices such as cellular phones and PDAs. At the same time traditional marketing models (Amor, 2000; Bernstein, 1999; Straus and Frost, 1999) have their limitations when applied to Internet-based commerce. These limitations are related to advanced technical means for distribution of marketing information (emails, SMS-messages) and to limitations of users to analyze it as well as limitations of mobile devices to present it. If no new solutions will be applied the amount of potentially unseen offers and non-contacted potential customers will grow dramatically. A possible way for relaxing the above-mentioned limitations could be in usage of electronic assistants both for customers and providers. Such assistants could serve as mediators for commercial Internet-based activity. A role of such mediators may play software agents (Bradshaw, 1997; Nwana, 1996; Wooldridge and Jennigs, 1995) representing customers and providers in the network. This approach becomes popular among researchers last years. However, there are not too many widely accepted technical/practical agent-based solutions proposed till now.

In this paper we present our experience and a solution to using agent technology in customer services management for mobile users. The solution is intended to increase granularity and personalization in targeted advertising while ensuring customer privacy.

The rest of this paper is organized as follows. First we consider a problem of mobile services provision. Next we describe architecture of a system for support of such services. Then we discuss some implementation details and the interface to the system. Finally, we present an example and conclusions.

2. The Problem Description

The problem we would like to consider can be formulated as follows. A customer with mobile device would like participate in e-commerce process and have access to the following services:

- to search for some product which s/he is interested in;
- to be notified about valuable offers of products s/he is interested in.

There are several pre-conditions for the services which the customer requires to be satisfied:

- the information delivered both via product search and offer notification should be delivered to the customer's mobile device;
- the information about products delivered both by search engine and notification service should be as compact as possible but sufficient enough to be interesting to the customer (this requirement comes from restricted expressive capability of customer's mobile device);
- the delivered information should be as precise as possible (in other words it should not, as much as possible, contain non-relevant data or information which is out of the customers interest);
- the customer would like to keep privacy of some his/her personal data (s/he agrees to disclose only some (if none) of his/her personal data).

We also assume that the customer should be subscribed for a selected service. However, we remind that s/he is willing to disclose only minimal (if none) of his/her private information to the provider of such service.

3. Agents for Mobile Services

Taking into account the requirements from the previous Section we propose that each customer participating in mobile commerce activity is provided with a software assistant agent representing him in the Internet. Main goal of these agents is to keep the customers profile, analyze available information autonomously and notify customer about events which are important to him. The main advantages of the agent-based approach to mobile services problem solving we see as follows:

- operating autonomously and off-line the agent can analyze much more information than the human customer;
- agents keep customer profile and allows very personalized processing of available information;
- agent keeps privacy of the customer profile (it encapsulates the profile and makes secure analysis). Implementation of the privacy issues is not a simple task, however, we assume that customer agents operate in a secure environment – which can be a customer's PC or another trusted host. The agent doesn't disclose to service provider how the provider's information is analyzed (it is done privately by the agent) and the agent communicates with customer only by results of analysis but not by the criteria for analysis.

General issues related to usage of agents in mobile commerce services were presented in (Matskin and Tveit, 2001; Matskin, 2000; Matskin, 2001). In this paper we are mostly focused on some implementation details of the approach rather than on its general discussion.

4. General System Overview

4.1. *System Architecture*

Applying the agent-based approach to our problem solving we can present a general architecture of the agent-based system as it is shown in Fig. 1.

The system design is divided into two main parts referred to as client and server side. In addition, external online stores are included in Fig. 1 as the system incorporates interfaces to communicate with such entities.

The server side. The *database* contains an event list with product/service offers from the *participating stores*. Participating stores are stores that contribute to the event list and cooperate with the agent system. Access to their product databases is not assumed to be by a search engine on their web site (as opposed to external online stores). Instead of sending

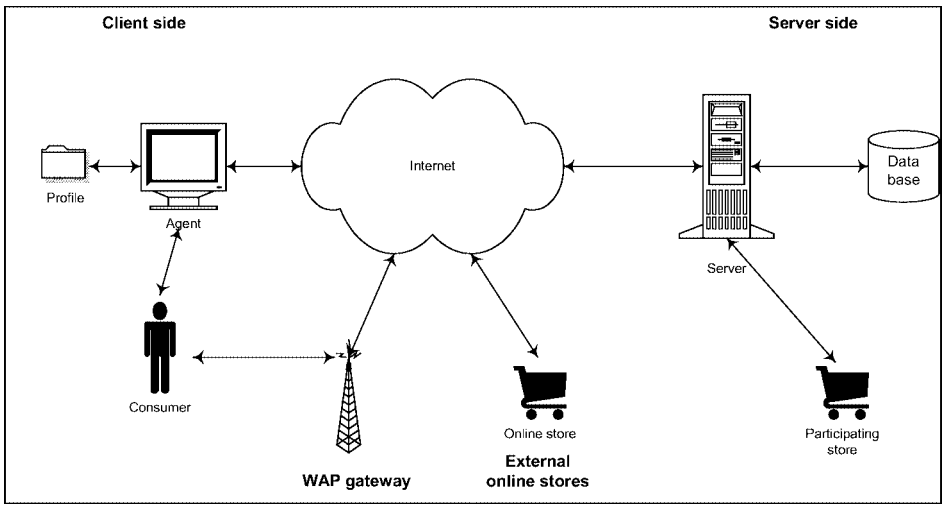

Fig. 1. General system architecture.

the offers from a customer database directly to the agents, the offers are just registered in the event list for the agents access and their private analysis. In addition to the offer list the database may also contain a product register which can be used for several reasons. First, products referred by agents are registered in the database for providing service providers with anonymous information about customer's requests. Second, product databases from participating or external online stores may be mirrored in the server side database for faster product search (this is called "pre-fetching"). For product search requests to non-prefetched stores the requests are passed both to participating and external stores.

The *server* manages requests from the agents and from the users via the WAP gateway. The server acts mostly as an interface to the database, however, in some cases it handles requests itself.

The client side. In order to use the system, the *consumer* has to create an *agent* on his/her personal computer (an access to the Internet is required, and preferably via a permanent connection). The consumer's profile is stored on his/her PC without disclosing personal information - data from the profile are not sent anywhere. Agents regularly access the event list to download offers that have arrived since the last check and analyze them using the consumer's profile. When the agent finds an offer to be of user's interest a notification about the offer is sent to the user. The user may then access more information about the offer via his/her WAP device.

WAP gateway. *WAP gateways* are normally operated by telecom providers and act as the link between WAP devices and the Internet. In order to view the offer details, users have to dial up a WAP gateway and to get connection to the system's server.

External online stores. *Online stores* that are not considered as participating stores can also be included into product search. This is done by creating "wrappers" – programs that post search requests to the stores' web sites and parse the generated results. Creating wrappers, however, involves a lot of work and several companies have product search as

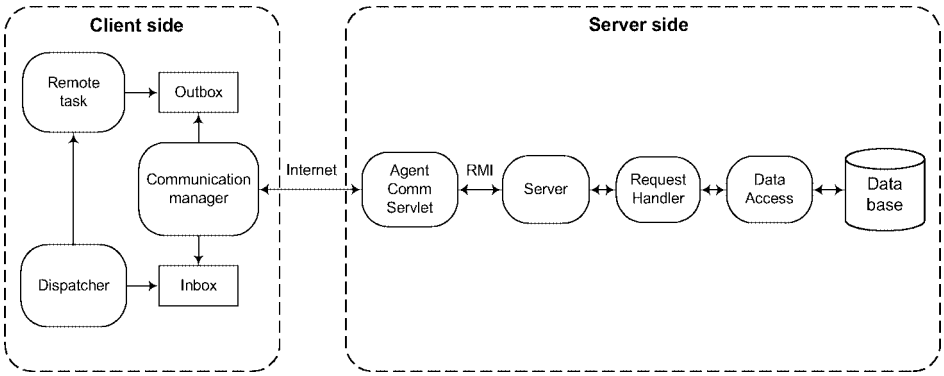

Fig. 2. Agent–server communication.

their sole business (for example, see (Kelkoo, 2002)).

4.2. *Communication Components*

Agent–server communication. Communication between server and client is asynchronous. Apart from some core functions, everything the agent performs is divided into *tasks*. Most tasks communicate with the remote server and they are called *remote tasks*.

Fig. 2 shows what happens when a remote task sends a request to the server (for communication between the agent tasks and the server, *message objects* are used). When the remote task creates a request message the message is put into the *outbox* message queue. The *communication manager* creates connection to the *agent communication servlet* (if there was no connection before) and sends the message. The servlet passes the request message to the *server* process via a Remote Method Invocation (RMI) connection. For each remote task on the client side, there is a *request handler* on the server side. Depending on the kind of request the server invokes appropriate request handler. Most request handlers need to access the database in order to fulfill the request and they create a *data access* object (or SQL object) to query the database. Then the request handler creates a response message that is returned to the agent.

At the client side the communication handler receives the response message and puts it into the *inbox* message queue. The dispatcher reads the response from the queue, checks the task identifier of the message and sends it to the corresponding task.

WAP browser–server communication. The communication between WAP devices and server side is shown in Fig. 3.

WAP devices send requests to the *web communication servlet*. Depending on request the servlet may or may not need access the *database*. In any cases the servlet forwards the request to the appropriate Java Server Page (JSP) (Avedal *et al*., 2000). The JSP pages usually need access to the database and this is done using *cache* objects - objects that retrieve data from the database when they are created and make the data available via corresponding methods. The output from the JSP pages is in the form of Wireless Markup Language (WML) pages that are returned to the WAP browser.

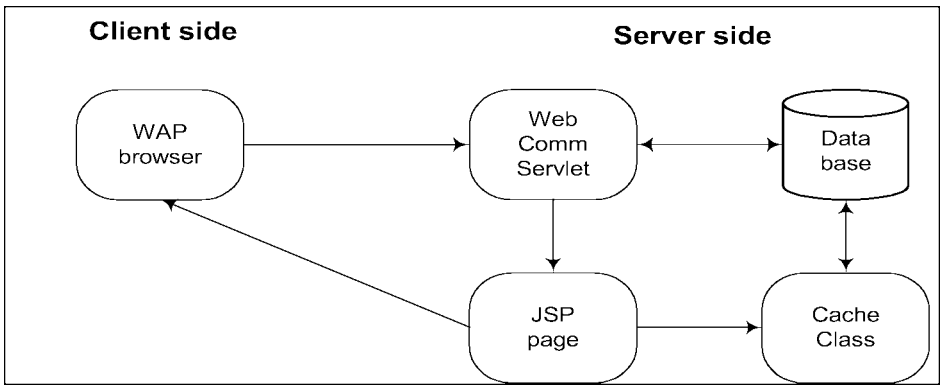

Fig. 3. WAP browser–server communication.

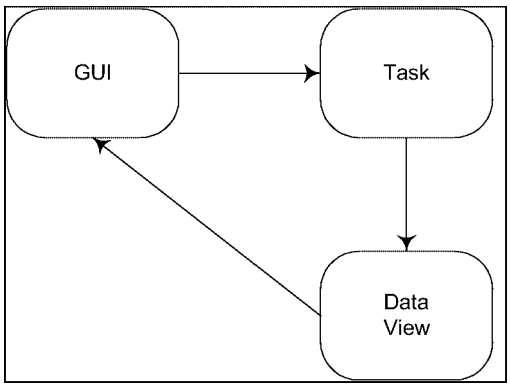

Fig. 4. GUI interaction.

GUI interaction. Some of the tasks interact with the GUI either for displaying or for reading data. The main goal of the GUI design was to make the GUI and the business logic (i.e., tasks) as much independent as possible. The solution is to use *data view* classes as a link between tasks and the GUI (this is shown in Fig. 4).

When data has to be displayed, for example, when the user would like editing his/her profile, the GUI makes a call to a task of appropriate type. The task object collects necessary data and passes it to data view object. The data view object (contrary to the task object) knows the details of the GUI and how to display the data.

Reading data from the GUI happens in a similar manner. When data has been entered into the GUI, for example, when the user is finished editing his/her profile, the GUI makes a call to a task of appropriate type. The task object calls its data view object, which reads the data from the GUI and returns it to the task object.

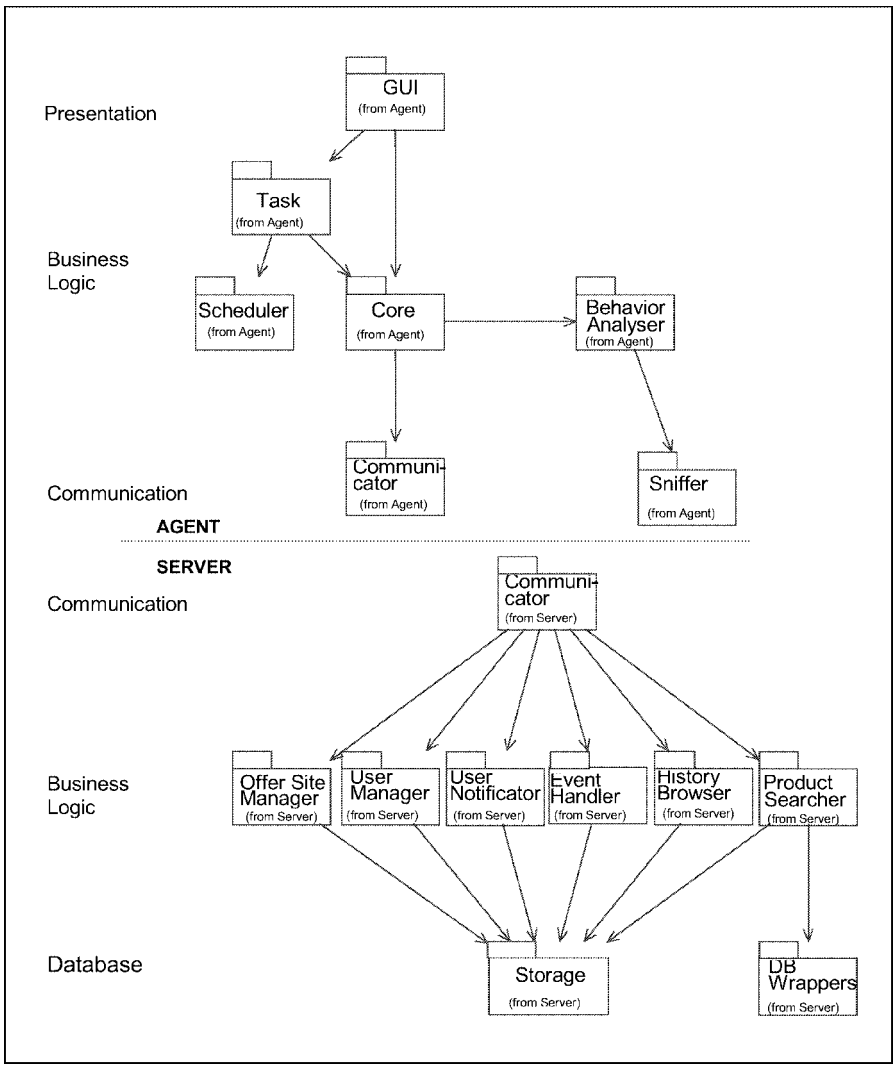

Fig. 5. Agent–server overview.

4.3. *System Packages*

The system design contains the following four packages: *Agent, Server, Message* and *Profile*.

The *agent* and *server* packages contain the logical build-up of the client side and the server side respectively. As a result, they are quite complex. The other two packages are less complex and contain mainly objects for (temporary) storage of data. Fig. 5 shows an overview of the agent and server packages, other packages are described in (Heiberg and Pedersen, 2001).

4.4. *Product Searcher Component*

When the server receives a request message of the ProductSearchRequest type a new SearchDispatcher object is created to handle this request. A product search can be performed both locally and externally.

For finding of stores which offer products in the requested category a query to the local database is made. This query uses the incoming category type as a parameter. As a result of the query processing, pointers to all stores offering products in the specified category will be returned. If any of the stores are prefetched by the system then a local search is made. For all non-prefetched stores (if any) the search is made externally.

Local search is made in the server's own database. The *isPrefetched* attribute assigned to every store indicates whether to make a local search or not. If the attribute is "set on", the store and all its products are prefetched and stored in the server's database (the search can be made locally in this case). Results of a local search are in the system format and they don't need further processing.

External search is made for shops that are not prefetched by the server (*isPrefetched* attribute is "set off"). These shops can be a part of the system (participating shops) or they can be integrated into the system for product search purposes only. Unlike the local search the external search must be processed both prior and after the execution. Prior to the search the server has to deal with differences in category structure. A search which is based on the server's own representation of categories, probably, will not be understood by the shop. Hence the server must map its category structure into the category structure of the shop.

When the server finishes mapping categories and properties it is ready to perform search. Results of the search will need further processing upon arrival before they can be used by the server. In particular, the server must perform filtering the search results and translation of the results in order to fit them to the local database format.

5. Implementation Tools

The implementation is done using Java and Java based utilities.

On the client side, the core Java 2 platform was used.

On the server side, a few other tools are needed. In addition to the core Java 2 platform, the Java 2 Enterprise Edition (J2EE) Reference Implementation libraries (Pawlan, 2000) are required by two servlets as well as for e-mail sending. In addition to the libraries a relational database and an application server (that is able to host JSP and servlets) are needed. For the relational database, we used the limited version of Cloudscape that is shipped with the J2EE software development kit. For the application server we ended up with Orion version (Orion, 2002), which is free for development and non-commercial purposes. Due to the limitations in the supplied version of Cloudscape the database, the application server and the Java server process have to execute on the same physical computer.

Fig. 6. Configuration screen.

In order to demonstrate how to search external online stores, we implemented a wrapper for the Norwegian *Telehuset* (Telehuset, 2001) – this store is specialized in computer and telecommunication equipment.

As the focus of design has been more on the customer part rather than on the service provider part, no interface has been implemented for the providers to add offers and products into the system's database. Currently this has to be done via SQL tool.

6. Interface to the System

In order to use the system the customer should create and register an agent who presents his interests in the mobile commerce environment. Next step is presenting customer profile to the agent. This is done via a configuration screen as shown in Fig. 6.

In the interests screen users can select interests from a "tree" of categories (left-subwindow)

Product search. The prototype can perform search for products that are registered in the system's database (from prefetched stores) as well as for products located in one external (non-prefetched) store (Telehuset, 2001).

Some product categories may have subcategories. For example, *Products* in the *Software* category have three subcategories (*Model, Make*, and *Platform*) which the customer can use in addition to the general string that searches product titles. Some properties have a limited number of allowed values and they are displayed as combo boxes, while the others have text fields.

Fig. 7. Login page.

Fig. 8. Offer overview.

Events check. New events check task can be created and performed immediately or it can be scheduled to run later. The events check tasks can be scheduled to run only once or they can be scheduled to run regularly (recurring tasks).

Once the computer's clock reaches the task's scheduled execution time, it sends the request message to the server. The request contains an attribute that tells the server when the events check was done last time.

At the server side, an event checker queries the event list for valid offers (that appear after the previous check time) and returns them to the agent. The events check task retrieves the offers and sends them to filter manager. For each matched offers a notification task is created which sends a notification request to the server with information of where to send the notification (both e-mail notification and SMS messages could be used for this purpose). At the server side, a notificator creates e-mail and sends it to the given e-mail address. It is also registered that the user has not seen this offer yet but would like to do that next time s/he logs on to the WAP site.

When all offers have been filtered and the necessary notification requests have been sent, a browsing history task is created to collect information about the user's access to the offers via WAP devices since the previous check. At the server side a history browser object queries the database for offers that the user has seen (or followed links to) as well as how many times the user has accessed the WAP site and the previous access date.

WAP browsing. User can access more information about offers using a WAP (Mann, 2000; WAP, 2001) enabled cellular phone (or other WAP device). After entering the URL to the server in the WAP device the login page will appear. The user enters the username and password in the corresponding fields of the page (see Fig. 7 – here and further all screenshots have been taken by means of TTemulator (TTemulator, 2001).

Fig. 9. Offer details.

If the authentication was successful, the user is directed to an offer overview page. Otherwise an error page with a link to the login page is displayed.

In the offer overview page, the offers are divided into two categories (see Fig. 8). New (unseen) offers are listed first, followed by offers which have had a chance to be seen before. More information about individual offers can be seen after clicking their links (see Fig. 9).

7. Conclusions and Future Works

Application of agent technology to mobile customer services management seems to be very fruitful. It allows to keep a reasonable privacy of customer's data, provides personalization and better scalability of services because of asynchronous and autonomous processing. The future work is planned on extension of agent functionality by developing agent learning module allowing creation more accurate profiles and filters. Also the provider side of the system has been given too little attention in the design. Possible extensions we see in implementing a GUI for the providers allowing more convenient specification of their offers.

We used WAP protocol as a means for communication of mobile device with the Internet, however, the design is modular and any other protocol developed for this reason could be easily adopted by the system.

Acknowledgements

This work is partially supported by the Norwegian Research Foundation in the framework of the Information and Communication Technology program (IKT-2010) and the ADIS project.

References

Amor, D. (2000). *The E-business (R)evolution.* Hewlett-Packard Professional Books, Prentice Hall PTR.

Avedal K., *et al.* (2000). *Professional JSP: Using JavaServer Pages, Servlets, EJB, JNDI, JDBC, XML, XSLT, and WML*. Wrox Publishing.

Bernstein, M. (1999). *Enterprise Management Update: Web Personalization – Possibilities, Problems and Pitfalls.* Inside Gartner Group.

Bradshaw, J.M. (Ed.). (1997). *Software Agents*. Menlo Park, CA: AAAI Press/The MIT Press.

Heiberg, T., J. Pedersen (2001). Agents for Support of E-Commerce Services, *Diploma Thesis*, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim.

Kelkoo (2002). `http://www.zoomit.com`

Mann, S. (2000). *Programming Applications with the Wireless Application Protocol: The Complete Developer's Guide*. John Wiley& Sons.

Matskin, M. (2001). Intelligent Agents for Mobile Commerce Services. In Ning Zhong *et al.* (Eds.) *Intelligent Agent Technology. Research and Development. Proceedings of the Second Asia-Pacific Conference on Intelligent Agent Technology (IAT-2001)*, Maebashi Terrsa, Maebashi City, Japan, World Scientific, pp. 129–133.

Matskin, M. (2000). Agents in telecommunication-based services. Invited talk. In *Proceedings of Networking'2000 Conference, Mini-Conference "IATE - Intelligent Agents for Telecommunication Environments"*, Paris, pp. 77–90.

Matskin, M., A. Tveit (2001). Mobile commerce agents in WAP-based services. *Journal of Database Management*, **12**(3), 27–35.

Nwana, H.S. (1996). Software agents: an overview. *The Knowledge Engineering Review*. **11**(3), 1–40.

Orion (2002). `http://www.orionserver.com`

Pawlan, M. (2000). Writing enterprise applications with Java^{TM}2 platform, Enterprise Edition. `http://developer.java.sun.com/developer/onlineTraining/J2EE/Intro/`

Straus, J., R. Frost. (1999). *Marketing on the Internet.* Prentice Hall.

Telehuset (2001). `http://www.telehuset.no`

TTemulator (2001). `http://ttemulator.com`

WAP (2001). Wireless application protocol (WAP). `http://www.wapforum.org`

Wooldridge, M., N. Jennings (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, **10**(2), 115–152.

T. Heiberg graduated from the Norwegian University of Science and Technology (NTNU) in 2001 with a M.Sc. in computer science from the Department of Computer and Information Science. Parts of his master studies he performed at the Queensland University of Technology (QUT) in Australia. His work experience includes software development at Accenture and his current employer BEKK Consulting, as well as a pre-sales role at the latter. Recent work has been related to portal and collaborative solutions development. His research interests include agents-based simulations and architectures.

M. Matskin, professor at the Department of Computer and Information Science of the Norwegian University of Science and Technology. He received his Ph.D. in computer science from the Institute of Cybernetics (Tallinn, Estonia) in 1984 and his M.Sc. in electrical engineering from the Tallinn Technical University in 1978. He was a head of Department of the Soviet New Generation Computer Project START (1985–1988) and a leading researcher and head of Department of the Institute of Cybernetics (1989–1995). Dr. Matskin has works in agent technology, information systems, formal methods for software design and knowledge-based programming. His current research interests include telecommunication-based services, multi-agent system environments and agent-mediated e-commerce.

J. Pedersen graduated from the Norwegian University of Science and Technology (NTNU) 2001 with an M.Sc. in computer science from the Department of Computer and Information Science. He has agent development experience from Zoomit, a shopping comparison portal that later merged with the Kelkoo company (September 2000). After receiving his M.Sc. in February 2001 he re-joined the Kelkoo company to continue work with agent development (he is currently their CTO for Norway, Sweden and Denmark).

Agentais pagrįsta architektūra klientų aptarnavimui valdyti ir produkto paieškai atlikti

Thomas HEIBERG, Mihhail MATSKIN, Jøran PEDERSEN

Produktų ir paslaugų skaičius internete nuolat didėja ir vartotojams tampa per sunku juos analizuoti bei lyginti. Iš kitos pusės potencialių klientų skaičius internete stulbinamai didėja ir paslaugų tiekėjams tampa per sunku vykdyti efektyvią prekybą. Galimas būdas sumažinti minėtus ribojimus galėtų būti elektroninių padėjėjų naudojimas klientams bei tiekėjams. Tokie padėjėjai gali būti tarpininkais komercinei internetinei veiklai. Tokių tarpininkų vaidmenį galėtų atlikti programinės įrangos agentai, atstovaujantys klientus ir tiekėjus tinkle. Šiame straipsnyje pateikiama autorių patirtis ir metodika naudojant agentų technologiją klientų aptarnavimui valdyti mobiliems vartotojams. Metodika nukreipta didinti individo asmenines galimybes bei užtikrinti kliento privatumą. Pateiktas sprendimas įgyvendintas prototipinėje sistemoje teikiant paslaugas mobiliųjų priemonių vartotojams.