Blaž Rodič

Mobile agents for distributed decision support systems

Article Info:

Management Information Systems,
Vol. 6 (2011), No. 1,
pp. 020-027

Received 28 September 2010
Accepted 6 December 2010

UDC 005.552.1 ; 004.65

Summary

This article focuses on the performance of Java based mobile agents using format translation via an intermediary XML based format. Our goal was to develop and verify the performance of a lightweight, mobile agent based solution that would allow strong security, portability and access to heterogeneous data resources from a mobile platform to facilitate exchange of data between simulation models and data resources. We have developed two types of agents: a mobile agent that functions as a server for queries in SQL and converts the query results into XML documents and a stationary agent acting as a client for query forwarding and conversion of received documents into text files readable by a client application. We have tested the performance of the agents in a distributed simulation scenario and established that the agents can be used to connect heterogeneous simulation models and other applications, improving their connectivity and usability.

Key words

mobile agents, XML, data filtering, data retrieval, distributed simulation, decision support systems

1. Introduction

Recently we have witnessed an increase of research interest in distributed information systems. New technologies in networking have improved the accessibility of information, whereas strong cryptography allows us to transfer information more safely. This has caused many changes in the way organizations do business and improved the mobility of people and jobs. The possibility of on-demand access to a multitude of data and processing resources is also very attractive for decision support. Better access to data can improve the accuracy and performance of business and manufacturing simulation models used in decision support systems (Kljajić, Bernik, & Škraba, 2000). In turn, faster and more accurate decision support can give a company an advantage over its competition.

In the past, simulation was mostly used to develop stand-alone solutions with a limited scope and lifetime (Harrell & Hicks, 1998). However, the use of computer simulation in various areas of business processes has resulted in the need to smoothly exchange data between simulation models and data resources used in different parts of an organization or in several organizations. Setting up the connections between distributed simulation models and other data sources can be a demanding task, especially if the models run within dissimilar simulation tools or on different platforms and there are both continuous and discrete event simulation (DES) models applied. There is a clear need for solutions that would simplify the exchange of data between simulations and other applications over the communication network. We have identified the following problems:

- Lack of a common data exchange method and format supported by all simulation tools, decision support tools, databases, etc.
- High amount of data exchanged by tightly coupled components of a simulation system.
- Security threats in public networks.
- Difficult control of remote components in distributed systems.

In order to address these problems, we have decided to develop a solution using Java mobile agents for data retrieval and filtering and to implement data format translation via an intermediary XML format. In this paper we have focused on the performance of the developed agents in a prototype scenario involving simulation models from real-life projects, and we aim to answer the following questions:

- What would be the impact of security mechanisms on system performance?
- What is the impact of data package size on system performance?
- How much processing time is required by different parts of code in the agents?
- What is the latency (minimum response time) of the system?

2. Literature review

We have examined several available technologies for development of distributed simulation systems and the related research. One of the recent developments in the area of distributed simulation is web-based simulation - the use of web technologies such as Web Services, Javascript, CGI and Adobe Flash to facilitate access to remote simulation models. Web-based simulation sacrifices performance and sometimes data security in exchange for accessibility and ease of use; however it is very well suited for educational use (Karagiannis, Markelis, Paparrizos, & Sifaleras, 2006). An example of a professional simulation system that is tailored for web-based simulation is Anylogic (XJ Technologies, 2009). Anylogic produces stand-alone models in Java and Java Applet models that can be stored in XML files. Anylogic supports several modeling paradigms, and thus allows the construction of agent based models, system dynamics models, and discrete event models.

A technology that has gained a lot of attention recently, especially in the field of distributed systems are mobile agents (MA), a subset of software agents. The term "agent" comes from the field of artificial intelligence, and in its broadest sense means an entity that engages in an activity in the name of another entity. In the software community the term "agent" is used for programmes that have a certain degree of intelligence and adaptability, being able to operate without constant supervision and less user input (e.g. software setup wizards). Mobile agents add another degree of autonomy – the ability to move between computer systems. Naturally, this requires an infrastructure that allows for transfer and execution of code. Agents can reduce network traffic, encapsulate protocols, execute asynchronously and autonomously, adapt to their environment and can be used to build robust, failure resistant systems (Lange & Oshima, 1999). For these reasons one of the more popular uses for mobile agents is in the development of distributed systems (White, 1996).

Maamar, Yahyaoui, and Mansoor (2004) describe a system intended to facilitate the use of ecommerce on mobile wireless devices. Problems involved include low bandwidth, high latency and security of transactions. In order to tackle these problems, software agents were used in the design and development of the system. The agents use a proprietary method for control and exchange of data, whereas the security mechanisms are not described. Agent security is the main focus of research by Chunlin and Layuan (2003). The authors have identified mobile agents as a threat to the security of local resources and propose the construction of a distributed system that would filter access to the resources via several "service agents". The focus of research by Qi and Chakrabarty (2001) is the use of mobile agents for integration and filtering of data in a distributed sensor network. Whereas the traditional approach would gather all available data at a central location, here the agents move from sensor to sensor and locally filter relevant data, reducing the data flow by up to 90%.

3. Methodology

Despite these developments, we believe that there is a niche for a lightweight, portable tool that would facilitate the connection of simulation models and data resources over the Internet and provide data filtering as well as security. We undertook the construction of a flexible, agent based middleware tool that would allow us to transfer and convert structured data (two-dimensional tables) with local filtering, a secure, encrypted data transfer and mobile agent authentication. We decided to develop the software in Java to provide portability and cross-platform mobility of agents and decided to use standard internet security mechanisms. As there are a number of agent development platforms already available, we tried to find a platform that would provide built-in support for important functionalities such as transport, control and secure communications between distributed components. We have also decided to implement data format translation using basic XML tables as an intermediary format.

3.1. Mobile Agent Platform

Our choice of methodology and technology was guided by the following goals: reuse of a tested, well documented and freely available technology, high portability of solutions, and support for mobile devices. We have looked at several MAS (multi agent system) platforms, including Aglets, Odyssey, Voyager and Grasshopper (Mangina, 2002), and decided to utilize the Grasshopper V2.2.4 platform by IKV++ (http://www.ikv.de, also available at: ftp://kibernetika.fov.uni-mb.si/SoftwareAgents/) to develop the software agents for distributed simulation support. The Grasshopper platform has the following technical

advantages over other MAS: it's entirely built in Java, it's compatible with most computer platforms, its source code is open, it has a good implementation of agent transport, it provides well developed control and security mechanisms, and finally, it includes excellent documentation and a free academic license. The central part of the Grasshopper platform is a distributed processing system, which integrates the conventional client/server architecture and the software agents' technology. The Grasshopper system is implemented in Java version 2 and is one of the first agent platforms to implement MAS interoperability standards such as MASIF (Mobile Agent System Interoperability Facility) (Object Management Group consortium, 2009) and FIPA (Foundation for Intelligent Physical Agents, 2010).

The Grasshopper platform builds on the concepts of region, place, agency and several types of agents (Figure 1).

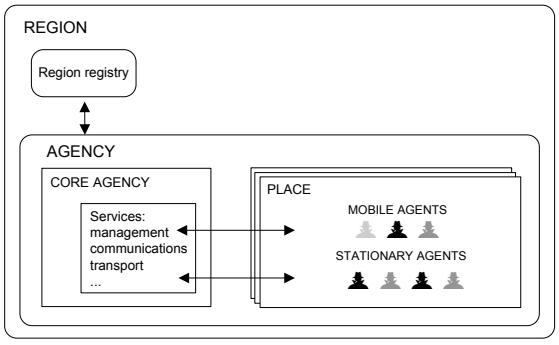

Figure 1 Structure of a Grasshopper based agent system

An agency is an instance of the Grasshopper application that hosts software agents and provides services such as communications, registration, data transfer, security, transport and archiving. Every computer that we want to connect to a distributed multi-agent system should be running at least one agency. Every agency contains the so-called core agency and several places where the agents can run. Agencies handle virtually all services related to the lifetime of agents. The concept of place aids the grouping of agents inside agencies according to their purpose or functionalities. A region registry keeps track of all agencies and agents within the region and enables communication with mobile agents regardless of their location. The information on agent states and events is reported to the registry by concerned agencies. More details about the platform can be found in the following sections.

4. Agent system prototype

To test the software agents in the context of distributed simulation we needed to develop a distributed system prototype. We have decided to use agents to connect two different simulation models via their data resources. The prototype scenario involved a laptop running a manufacturing simulation that requires access to the current results of a financial simulation running on a company server, accessible via a public network.

The prototype application used simulation models derived from the models used in a project of manufacturing process reengineering. In that project we have constructed several simulation models: a continuous simulation model for the financial analysis of investments and several DES (Discrete Event Simulation) models to represent the reengineering scenarios. The continuous simulation model running in Powersim Studio (Powersim Software AS, 2010) acted as the data server, whereas the DES model running in ProModel (ProModel Corporation, 2009) had the role of a data client. Powersim and ProModel are both general purpose simulation tools and are designed for the Microsoft Windows operating system.

Powersim and ProModel cannot be directly connected, as they don't share a common data interface or data format that would allow runtime data transfer. The only viable method of connecting Powersim and ProModel is to transfer the data from Powersim to MS Excel workbooks and then to text-based CSV (comma separated values) files. We have decided to automate the process and implement data filtering and format translation using an intermediary format based on XML to facilitate the addition of new data formats. Our goal was to mask the complexity of the tasks necessary to fetch the desired range of data from a remote location and convert it into a desired format. After the agents are in place, a user should only have to enter an SQL query and specify the output file for CSV format data. SQL queries are a flexible and widespread method of querying databases and filtering large amounts of data, and were a natural choice for the data filtering method. Most data access drives that operate via the ODBC (Open Database Connectivity) or JDBC (Java Database Connectivity) allow the use of standard SQL for database queries.

We have implemented the translation of data formats used in the prototype using the Java XML API (application programmer interface). The xlSQL JDBC driver (JAVA.NET, 2007) was used

to access data in MS Excel (Microsoft Excel) workbooks accessed. We wanted to implement a translation method that would be compatible with mobile devices that do not run Windows or Microsoft Office, therefore we decided to use only Java at the client side to translate incoming data from XML into CSV files.

4.1. Prototype System Structure

We have divided the distributed system into several components, shown on Figure 2:

- Simulations,
- Data resources and
- Middleware.

The function of middleware is implemented by the multi-agent system containing the following components:

- Mobile agents,
- Stationary agents,
- Agent execution platforms (agencies),
- Central registry and control application (region registry).

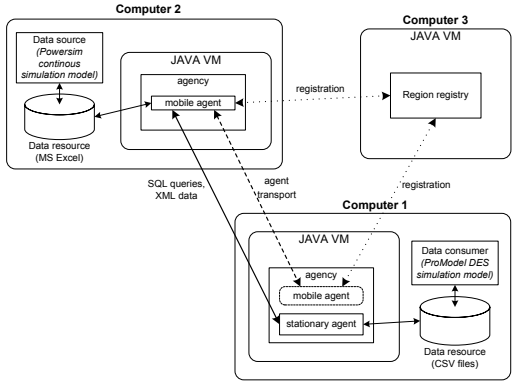

Figure 2 Prototype deployment diagram

The prototype of a distributed system contains three computers that host individual components of the system (Figure 2). The use case scenario has the user of "Computer 1" trying to obtain simulation data from "Computer 2" that is acting as a data source. "Computer 3" has the role of central registry and administration server. The "Computer 1" which runs the DES model contains the file used for data transfer (from continuous model to DES model) and an agency hosting a stationary and a mobile agent. The stationary agent is used to forward user queries to the mobile agent and then receive and convert the resulting data from the intermediate XML format to a CSV file. The mobile agent is used to fetch the data according to the user query (applying filtering), convert the data to intermediary XML format and send it to the querying stationary agent. The computer running the continuous simulation model (Computer 2) also contains an MS Excel file used to save and access simulation results and an agency that hosts the mobile agent. Finally, the computer marked "Computer 3" holds the region registry, which is used to control and administrate the agents and agencies. Figure 2 also displays the connections between components, where continuous lines show communication links, whereas the dashed line shows the path of mobile agent migration. The agencies are Java applications, running within local instances of Java VM (Java Virtual Machine). It is possible to run several agencies on the same computer, and several agents within every agency. Every agent runs in its separate thread, making the parallel execution of several agents possible without special mechanisms.

A mobile agent's life cycle (Figure 3) is started by a user that would like to access data on a remote computer that hosts a data source and can accept mobile agents. On Figure 3 each box represents a state of the agent, with text inside a box describing a state, and text beside the arrow describing the event leading to a state. The agent can be started in any agency that is registered with the region registry. Immediately following the start, the agent is a passive object (not executing), and has to be activated via agency GUI (graphical user interface). The initial state of a mobile agent in Figure 3 is thus "Awaiting activation". After the activation the agent initializes itself and asks the user (via a GUI) what agency the user wants to send it to and what data source to access at the destination. The user doesn't need to know the exact location of the agency such as the computer name or its IP (internet protocol) address, as it is transparently provided by the region registry. The mobile agent has a mobile and a stationary part. The stationary part remains at the source agency and exists only to allow the user to control the life cycle of the mobile agent. After the agent is ordered to move to the destination agency, it creates a copy of itself. It is the copy that then moves to the remote system, whereas the original becomes a passive object again. Security mechanisms for remote agent control and data access are implemented using a shared secret. Note that "COPY" and "ORIGINAL" are two instances of the same agent, with identical code - the difference between them is in the part of the code (method) executed and the data state of the agent.

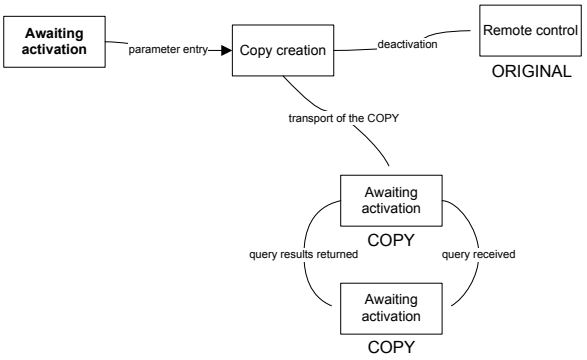

Figure 3 State transition diagram of the mobile agent

After the mobile agent has moved to target agency, it connect with the data source, assumes the role of a server and waits for incoming SQL queries. The mobile server agent can be removed either remotely by its owner or locally by the owner of the hosting agency.

Figure 4 shows the life-cycle of the stationary agent. Again, each box represents a state of the agent, with text inside a box describing a state, and text beside an arrow describing the event leading to a state. The agent is started by a user that needs access to remote data. The agent is activated during its initialization and assumes the state "Awaiting query" (marked bold in Figure 4). The agent then displays a GUI dialogue requesting the remote mobile agent address, pass phrase, the SQL query and the destination file. The stationary agent can be removed by its creator or the administrator of the hosting agency. The region registry administrator can delete the agent from the registry, thus making it inaccessible to other agents or agencies, but cannot physically remove or deactivate the agent.

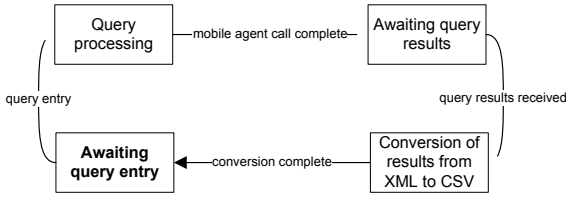

Figure 4 State transition diagram of the stationary agent

5. Performance of the MA system

The prototype allows us to access a remote data resource, fetch a defined range of data and convert it into the desired format and save the results in a local file. The software agent based system masks many operations that are necessary to fetch the desired range of data from a remote location and convert it into desired format.

We wanted to analyze the impact of security mechanisms on system performance. We have executed the tests using the plain sockets protocol and compared the results with the performance of the system using the Secure Socket Layer protocol. We also measured system performance using different data package sizes in order to establish the suitability of the system for different applications. As we have used a third party data access driver (xlSQL) we also wanted to test its performance and influence on the performance of the entire system.

The test environment contained three IBM PC compatible network workstations. The region registry was operating on a Windows XP system, a DELL Inspiron 8100 laptop (COMPUTER 3 on Figure 2). The mobile and stationary agents were installed on a Wind ows XP SP3 system, a IBM Thinkpad r50p laptop (COMPUTER 1 on Figure 2). The role of remote data source (COMPUTER 2 on Figure 2) was handled by another Windows XP SP3 system, a desktop machine with an Athlon 64 3200+ CPU. All computers were connected to the local area network via Fast Ethernet (100 Mbps) network adapters and 100 Mbps Ethernet switches. The software used in the experiment was MS Excel 2007, xlSQL version Y7, Grasshopper V2.2.4, Powersim Studio 2007 and ProModel version 5.0.

Figure 5 shows the dependence of system performance on the size of a cell range that a query returns. To gauge the system performance, we have measured processing time from the entry of an SQL query to the writing of a CSV file. We have used several different sizes of query results, from 1 to 5000 rows of data, each row containing two 64 bit numbers. The processing time includes the transfer of the SQL query to the mobile agent acting as a server, querying using the JDBC driver, conversion of the resulting data range to XML, sending XML back to the client agent and transfer of results to the CSV file. We have established that processing time is proportional to the query result size, it seems to grow exponentially, and that the use of secure mechanisms for communications reduced the system performance by approximately 20 percent. Processing time does not drop much below 100 ms, even for very small query sizes. We have found out that the processing time is limited by the latency of the xlSQL driver, as is explained in the following paragraphs.

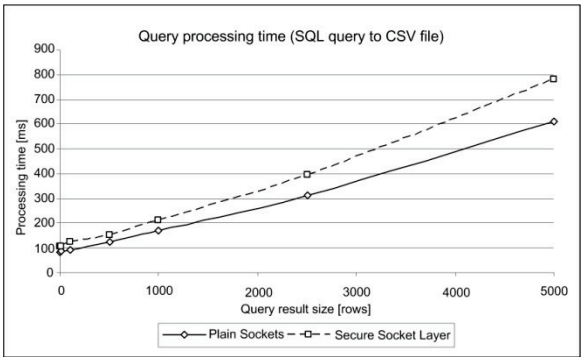

Figure 5 Query processing time in relation to query result size

Figure 6 shows the same data as Figure 5, but this time from the perspective of system throughput - the average number of records processed per second. The dramatic drop in system performance as the query size gets smaller is much more evident in this type of graph. Again, the limiting factor is limited by the latency of the xlSQL driver. Throughput for queries that return one row is 12 rows per second. In contrast, for queries that return 5000 rows, throughput is as high as 8193 rows per second.

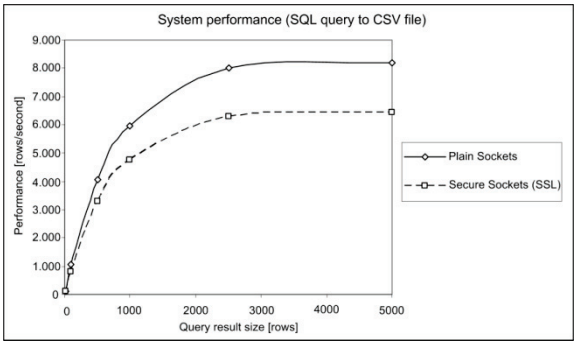

Figure 6 System throughput depending on the query result size

The process of data retrieval and filtering has several distinct parts, however we have decided to take a closer look at the tasks specific to our agents. We wanted to know what share of the total processing time is taken by each of the agent's tasks. We have measured the average duration of processing from the submission of a SQL query to receipt of results in XML format and the average duration of processing from the receipt of XML data to the completed conversion of results into a CSV file. The results of our measurements are shown in Figure 7. It seems that the processing share of client agent tasks (conversion from XML to CSV) grows with the size of query results. Conversion from XML to CSV format becomes relatively slower with bigger data sets, but it's still much faster than the execution of the SQL query and subsequent conversion of data to XML.

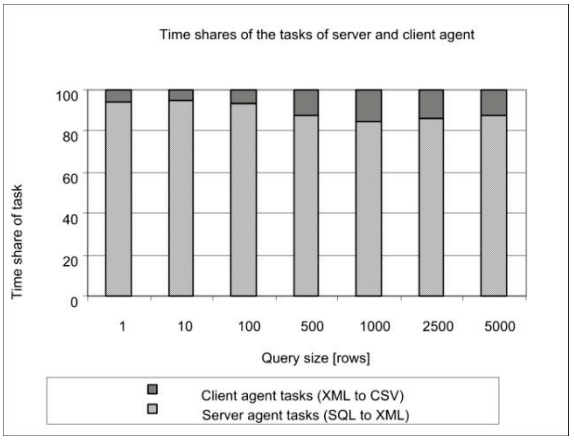

Figure 7 Time shares of the tasks of a server agent and a client agent

As the mobile server agent relies on the xlSQL driver to do its job, we decided to gauge the performance of the xlSQL driver, by separately measuring the duration of the agent's two distinct tasks: the execution of the received SQL query and the conversion of the resulting data range into an XML table. Figure 8 shows the average duration of these tasks. These tasks involve the methods implemented in the mobile agent, methods implemented by the local JDBC driver xlSQL accessing MS Excel files, but do not include the transfer of data over the network. Network performance in different mobile agent scenarios may vary a lot, therefore we decided to exclude this variable from our performance measurements. The accuracy of results was limited due to the limited resolution of the system clock (10 ms). Judging by Figure 8, the duration of SQL queries is affected by the query result size only for query result sizes of under 1000 rows, whereas the duration of conversion of the data set to XML increases proportionally with the size of query results. As we have found, the duration of an SQL query is the limiting factor for the latency of mobile agents. The shortest achieved duration of a query and thus agent latency in our tests was 60 ms. Such latency significantly limits the usability of the system for real-time access to data, however it should be noted that this latency is due to the xlSQL JDBC driver and not the agents. With a different data source and access method, the latency would change.

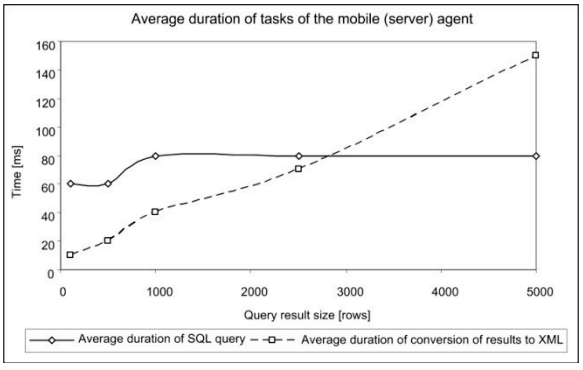

Figure 8 Average duration of tasks of the mobile agent

Our results show that the system performance is affected by both query result size and the use of security mechanisms, as we have expected. The system throughput was highest when the size of query results was several thousand rows. The latency of the mobile server agent is affected by the duration of SQL query. The minimal latency we have achieved with the mobile agent during our test was in the order of 60 ms and the smallest total time of service (SQL query to CSV file) achieved was approximately 100 ms, which translates to about 10 transactions per second. That speed is unsatisfactory for a real-time application of the system but may be adequate for decision support systems.

The highest achieved throughput is approximately eight thousand records per second, where each record contained two numbers in the "double accuracy floating point" format with the size of 64 bits each. This speed is in our opinion adequate to link business simulation models and other applications, but not appropriate to conduct real-time data transfer between complex simulation models or applications with intensive communication between components. That can be expected as Java applications still tend to be relatively slow compared to compiled native applications. Also, MS Excel workbooks are not intended for the storage of large amounts of data and cannot compete with relational databases for speed of access. Given these limitations, we conclude that the achieved throughput is satisfactory.

The use of security mechanisms in data transfer has a notable negative effect on the system performance due to increased communication setup and data transfer overhead of secure protocols. Establishing a connection using SSL has several additional steps compared to plain Sockets, and some of these steps are computationally intensive (encryption and key generation). SSL also requires some additional resources on the computer (key storage). As all transferred data is encrypted using strong encryption, the overhead is significant during the entire communication. Our tests show that the use of security mechanisms slows the system performance down by approximately 20 percent. We believe that although significant, the security-performance trade-off is acceptable as the use of SSL makes the system considerably more resistant to eaves-dropping, impersonation, unauthorized modification of data and "rogue" (malicious) agents.

6. Conclusions

We have developed two types of agents: a mobile agent that functions as a server for remote queries in SQL (Structured Query Language) and converts the query results into XML documents and a stationary agent acting as a client for query forwarding and conversion of received documents into text files readable by a client application. The results of our research show that software agents can be used to connect distributed simulation models, developed with different general purpose simulation tools and databases, thus improving the connectivity and usability of simulation models in distributed information systems. The use of standard security mechanisms provide authentication, confidentiality and integrity of information and contribute to the safety of the entire distributed system without a major negative impact on system performance. As the developed agents automate data filtering and format conversion, as well as a part of data retrieval, the agents also reduce the time necessary to link the simulation models and significantly simplify this process. By using an intermediary format we only need to develop two converters for every new data format, i.e. $2*n$ converters for conversion between n different data formats. Without an intermediary format, $n*(n+1)$ converters are necessary.

Whereas the server side of the system (the computer hosting the data resource and mobile server agents) has to provide an agent platform and an appropriate JDBC driver to access data, the client side needs only the agent platform for full functionality. Therefore any computer that can run the Grasshopper agent platform (most systems that support Java) can be used to easily access remote MS Excel data with simple yet powerful SQL queries. This significantly facilitates the integration of different simulation models and applications from various platforms into a distributed information system.

However, several limitations exist within this system. The selected data access driver (xlSQL) introduces latency that limits the real-time application of the agents. In the future we intend to implement conversion to and from several other data formats and verify the system performance with different data formats, automate data retrieval, and verify the performance of the system using mobile devices running Java.

Acknowledgment

The research was financially supported by the Slovene Ministry of education, science and sports within the "Decision Systems in a Global e-Economy" programme, code: PP-0586-501 and the Young Researcher programme.

References

Chunlin, L., & Layuan, L. (2003). Combine concept of agent and service to build distributed object-oriented system. *Future Generation Computer Systems , 19* (2), 161-171.

Cooper, A., & Reimann, R. (2003). *About Face 2.0: The essentials of interaction design.* New York: John Wiley & Sons.

Đurković, J., & Tumbas, P. (2000). *Metodološki prilazi, metodi i tehnike razvoja informacionih sistema.* Subotica: Ekonomski fakultet Subotica.

Foundation for Intelligent Physical Agents. (2010). Retrieved July 3, 2010, from http://www.fipa.org

Galitz, O. (2002). *The Essential Guide to User Interface Desing.* New York: John Wiley & Sons.

Harrell, C. R., & Hicks, D. (1998). Simulation software component architecture for simulation-based enterprise applications. *Proceedings of the 1998 Winter Simulation Conference* (pp. 1717-1721). Piscataway: The Society for Computer Simulation International, IEEE.

JAVA.NET. (2007). Retrieved May 3, 2007, from https://xlsql.dev.java.net

Jošanov, B., & Tumbas, P. (2002). *Softverski inžinjering.* Novi Sad: Viša poslovna škola.

Karagiannis, P., Markelis, I., Paparrizos, K., & Sifaleras, A. (2006). E-learning technologies: employing Matlab web server to facilitate the education of mathematical programming. *International Journal of Mathematical Education in Science and Technology , 37* (7/15), 765-782.

Kljajić, M., Bernik, I., & Škraba, A. (2000). Simulation Approach to Decision assessment in Enterprises. *Simulation , 75* (4), 199-210.

Lange, D. B., & Oshima, M. (1999). Seven good reasons for mobile agents. *Communications of ACM , 42* (3), 88-89.

Maamar, Z., Yahyaoui, H., & Mansoor, W. (2004). Design and Development of an M-Commerce Environment: The E-CWE Project. *Journal of Organizational Computing and Electronic Commerce , 14* (4), 285-303.

Mangina, E. (2002, June). *Review of Software Product for Multi-Agent Systems.* Retrieved February 7, 2010, from AgentLink: www.agentlink.org/admin/docs/2002/2002-47.pdf

Object Management Group consortium. (2009). Retrieved September 1, 2009 , from http://www.omg.org/

Powersim Software AS. (2010). Retrieved February 27, 2010, from http://www.powersim.com

ProModel Corporation. (2009). Retrieved February 27, 2009, from http://www.promodel.com

Qi, H., Iyengar, S., & Chakrabarty, K. (2001). Distributed multiresolution data integration using mobile agents. *In Proceedings of IEEE Aerospace Conference. 3*, pp. 1133-1143. Piscataway: IEEE Service Center.

White, J. (1996). Telescript technology: mobile agents. In J. Bradshaw, *Software Agents.* Cambridge: AAAI Press/MIT Press.

XJ Technologies. (2009). Retrieved December 3, 2009, from http://www.xjtek.com

Blaž Rodič
University of Maribor, Faculty of organizational sciences
Faculty of Information Studies
Novi trg 5
SI 8000 Novo mesto
Slovenia
Email: Blaz.Rodic@fis.unm.si