Mobile Agents and XML for Distributed Simulation Support

Blaž Rodič, Miroljub Kljajić

Faculty of Organisational Sciences, University of Maribor, Kidričeva cesta 55, 4000 Kranj, Slovenia
blaz.rodic@fov.uni-mb.si

Podpora distribuirani simulaciji z mobilnimi agenti in XML

V članku je predstavljena uporaba programskih agentov in XML za povezovanje simulacijskih modelov in podatkovnih virov preko komunikacijskega omrežja. Razviti in preizkušeni sta dve vrsti programskih agentov: mobilni agent, ki deluje kot mobilni strežnik za izvajanje poizvedb v SQL (Structured Query Language) in pretvorbo rezultatov v XML (eXtensible Markup Language) format in stacionarni agent, ki deluje kot odjemalec za posredovanje poizvedb in pretvorbo podatkov iz XML formata v CSV (Comma Separated Values) datoteke. V raziskavi smo ugotovili, da je s programskimi agenti možno povezati različne geografsko ločene simulacijske modele in podatkovne baze in tako izboljšati povezljivost in uporabnost različnih simulacijskih modelov v distribuiranih informacijskih sistemih.

Ključne besede: programski agenti, Java, XML, middleware, distribuirana simulacija

1 Introduction

Recent years have seen an increase of research interest in distributed information systems. The main reason is probably the fast growth of the best-known distributed information system today - the Internet. The possibility of accessing a multitude of data and processing resources is also very attractive for simulation purposes, and so the area of distributed simulation systems is gaining attention. In the past, simulation was mostly used to develop stand-alone solutions with a limited scope and lifetime (Harrell & Hicks, 1998). However, the penetration of computer simulation into various areas of business processes has resulted in the need to connect the simulation models used in different parts of an organization (Kljajić et al., 2000). Also, the trend in simulation development has shifted from purely analytical and optimisation oriented models to integrating simulation models into decision support tools to be used recurrently.

However, setting up the connections between distributed simulation models and other data sources can be a demanding task, especially if there are both continuous and discrete event simulation (DES) models present or if the models run within dissimilar simulation tools or on different platforms. 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 that we would like to address:

- Lack of a common data format understandable to all simulation tools, decision support tools, databases, etc.
- High amount of data exchanged by the components of a simulation system.
- Security threats in non-private networks.
- Lacking control of remote components.

The demands of distributed simulation are different than those of conventional simulation. Distributed simulation is deployed on a broader scale and relies heavily on shared data. The developers focus is shifting from the "all in one" simulation environments with integrated graphical model building, animation and analyses tools towards "barebone" simulation engines with comprehensive application programming interfaces (API). Models can be embedded as a component of the distributed simulation systems and dynamically built by procedures using databases. However, due to less flexibility and user interaction in model building and use, embedded and dynamically built models are only appropriate for specific problem domains. One of the recent developments in the area of distributed simulation is web-based simulation. But its weaknesses are the lack of interoperability with conventional simulation tools, difficult handling and a lack of real advantages over more conventional methods of building distributed simulation systems (Kuljis & Paul 2001). HLA (High Level Architecture) is a well-publicised framework for the construction of distributed simulation systems (Dahmann et al. 1998). The main force behind the development of HLA is the US Department of Defense, thus

the development of HLA is closely tailored to the needs of military simulation (Carson 2000). HLA is very highly specified and standardized (IEEE Standard 1516); however it's also too complex for general use. Complex specifications mean difficult development of models and nearly impossible integration of non-HLA simulations (Szymanski & Chen 2000). Another problem is the large consumption of bandwidth due to system overhead (Wayne & Gerald 1999). The CORBA architecture (Common Object Request Broker Architecture) (Buss & Jackson 1998, OMG 2005) is similar to HLA, but more general in its scope. Its aim is the interoperability of distributed systems, however it is difficult to use with legacy applications that do not implement it and suffers from similar complexity problems as the HLA.

Mobile agents are a technology that has gained a lot of attention recently. One of the more popular uses for mobile agents is in the development of distributed systems (White 1996). 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). The term "software agents" 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 - either a human being or another piece of software. In the software community the term has come to stand for programmes that have a certain degree of intelligence and adaptability, being able to operate without constant supervision or reducing the necessary user input (e.g. 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.

We have examined a number of articles in journals describing systems utilizing mobile agents in distributed simulations. A system described by Corbin (1998) utilizes agents to connect military simulation models and allows for remote control of the agents via Java applets in web browsers. The agents use a proprietary method for control and exchange of data, while no mention is made of security mechanisms. Szymanski and Chen (2000) have used the IBM Aglets toolkit to connect two simulations running on very different platforms. Their goal was to improve the usability of component based simulation models of complex systems. The authors compared the execution time of models running within the same shared memory space with a distributed system, where the models were linked over LAN and TCP/IP. They established that the additional networking overhead slowed down the simulation to a quarter of the shared memory system speed. Their conclusion is that the communication link between distributed simulation models is a serious potential bottleneck and effective distribution of components and implementation of data filtering is crucial to distributed simulation system performance. Another interesting example is the ABELS system (Mills-Tettey et al. 2002), which is based on stationary software agents and a broker application that is to allow the connection of various simulation models and other applications into the simulation system. According to available publications the ABELS system is still under development and lacks several necessary components, including the security mechanisms. Conversely, 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 limit access to the resources to communication with available "service agents". The focus of another research (Qi et. al. 2001) is the use of mobile agents for integration and filtering of data in a distributed sensor network. While 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%. Jen-Yen and Shih-Wei (2003) describe a system for the connection of different multi-agent systems (MAS). They propose an "Agent Gateway" to act as a MAS protocol converter. As the implementation of an ACL translator for every pair of different ACL's would be uneconomic, the authors propose the construction of an intermediary ACL, based on XML. With this approach, each new ACL would necessitate the development of just two translators - ACL to intermediary ACL and vice versa.

We feel that despite the wealth of research, there is still a need for a lightweight tool that would facilitate the connection of simulation models and data resources over the Internet and provide 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 (twodimensional tables) with local filtering, a secure (encrypted) transfer and mobile agent authentication. We decided to develop the software in Java to provide cross-platform mobility and 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 looked at several platforms, including Aglets, Odyssey, Voyager and Grasshopper (Mangina 2002), and decided to utilize the Grasshopper V2.2.4 platform by IKV++ (IKV++ 2003). We have also decided to implement data format translation using basic XML tables as an intermediary format. In contrast with heavyweight solutions such as the HLA or CORBA we decided not to provide explicit support for runtime interface connections, registration and search and synchronization of events. Instead we are aiming for a low cost solution that would connect the simulations at the data resource level, and not at the runtime level. Other advantages of our proposed solution over the existing methods are to include shorter and simpler connection setup procedures, an open-ended structure and the use of security mechanisms with a relatively small impact on adaptability and performance of the system.

2 Methodology

We have used the Grasshopper V2.2.4 agent development platform by IKV++ (IKV++ 2003) to develop the software agents for distributed simulation support. The Grasshopper platform was chosen as it is entirely built in Java and compatible with most computer platforms, because the source code is open and well documented, because of the good implementation of transport, control and security mechanisms, 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) (OMG 2005) and FIPA (Foundation for Intelligent Physical Agents) (FIPA 2004).

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

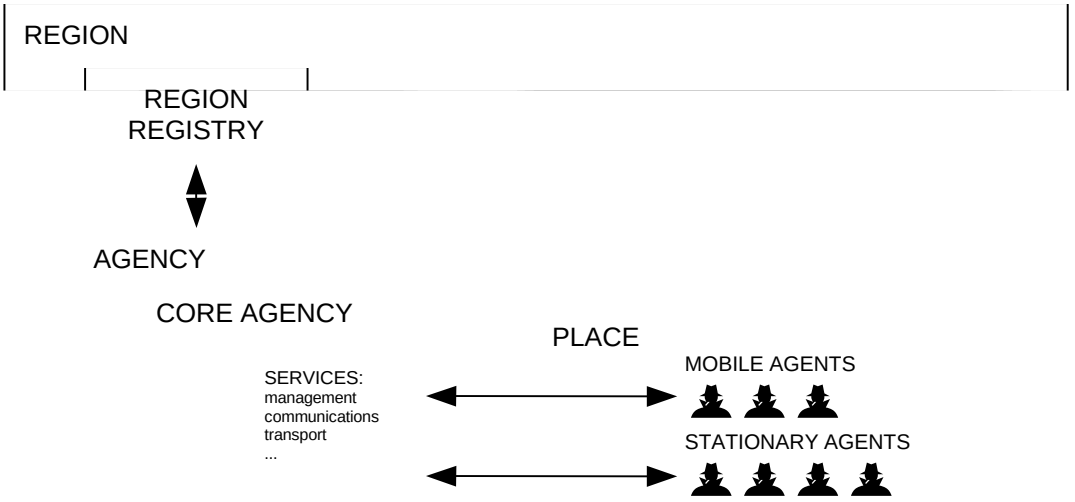

Figure 1: Structure of a Grasshopper based agent system

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 multiagent 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. Every agency has to contain at least one place on order to host agents. In contrast, the formation of regions is not mandatory, and serves only to simplify the communications between components of a distributed system. 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.

2.1 Prototype system

To test the software agents 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 application used simulation models derived from the models used in a production process reengineering project (Kljajić et al. 2000). In that project we have constructed several simulation models: a continuous simulation model for the financial analysis of investments and several DES models to represent the production line reengineering alternatives. In the prototype we gave the continuous simulation model running in Powersim Studio 2003 (Powersim AS 2004) the role of a data source, while a discrete event simulation model running in ProModel (ProModel Corporation 2002) had the role of a data consumer. 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. The only runtime data transfer option in Powersim is the Windows DDE (dynamic data exchange) link to MS Excel files, while the only easily accessible runtime data transfer option runtime data transfer option in ProModel is via text files, for example CSV (comma separated values) files. Therefore the only viable method of connecting Powersim and ProModel is to transfer the data from MS Excel workbooks to text-based CSV (comma se-

parated values) files. While it is possible to save data from MS Excel as CSV files, it is difficult to do remotely. Also, directly translating an MS Excel workbook into a CSV file or files would result in a large amount of poorly structured data that would be very difficult to use in ProModel, therefore data filtering is required. We have also decided to implement data format translation using an intermediary format based on XML to facilitate the addition of new data formats. Intermediary XML data is in the form of an XML table.

We have divided the distributed system into several components:

- Simulations,
- Data resources and
- Middleware.

The function of middleware is implemented by a multiagent system containing the following components:

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

The prototype of a distributed system contains three computers that host individual components of the system

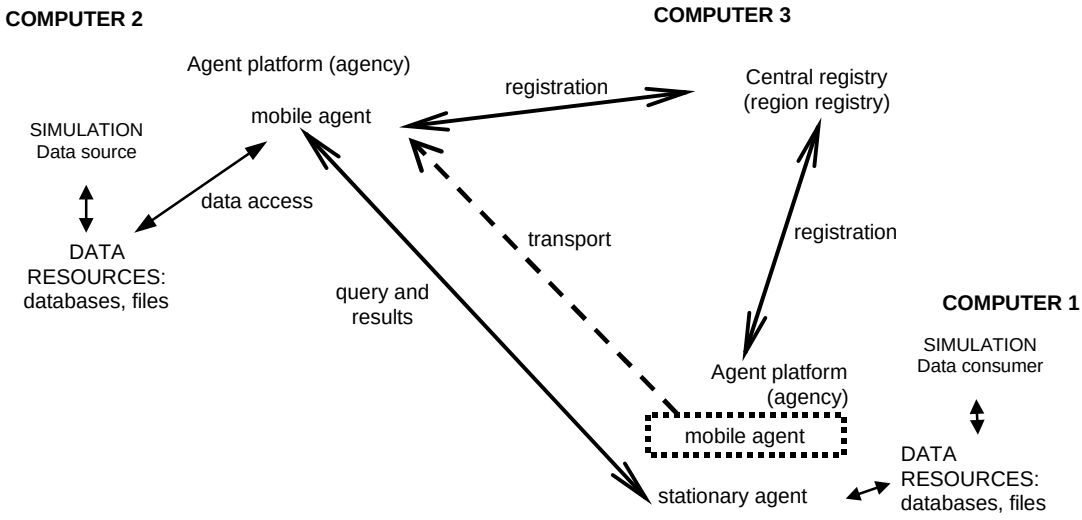

Figure 2: Distributed system prototype schematics

(**Figure 2**). The envisaged 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" 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, while 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. Every agency implements the following services:

- Creation, execution and removal of agents,
- Reception and execution of mobile agents,
- Control of active agents within the agency,
- Access to information on agents within the agency,
- Access to network and local resources,
- Registration of agents and the agency with the region registry,
- Inter-agent communications, and
- Security mechanisms for communications and agent transport.

The role of the region registry is to enable a centralized overview and control of components in a distributed agent system, i.e. agencies and stationary and mobile agents. The region registry implements the following services:

- Registration of agencies and agents,
- Overview of agent locations,
- Overview of agency and agent properties,
- Removal of agencies and agents from the registry,
- Use of security mechanisms for all communications.

A mobile agent is an executable piece of Java code that can run within an agency and implements the following functionalities and services:

- Migration to the data source,
- Accepting user queries (via the stationary agent),
- Retrieval of data according to the user query (filtering),
- Conversion from original (MS Excel) format to intermediary (XML table) format,
- Transfer of data to the querying stationary agent
- Remote control of life-cycle and
- Use of security mechanisms for communications.

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. The initial state ("Awaiting activation") is marked bold. The agent can be started in any agency that is registered with the region registry. Initially, the agent is a passive object (not executing), and has to be activated (with a double-click or via agency GUI). Then the agent initializes itself and asks (via a graphical user interface) what agency the user wants to send it to and what data source it should access there. The user doesn't need to know the exact location of the agency such as the computer name or its IP address, as it is transparently provided by the region registry. The agent then creates a copy of itself and deactivates, becoming a passive object again. This is done to enable the remote control of agent's life cycle - after the user requests the removal of the original agent, the original sends its copy the pass phrase that was randomly generated at the agent's activation (a "shared secret"), and the remote copy stops executing at the first opportunity and removes itself. The copy moves to the target agency, using an encrypted protocol. The transport is performed by the agencies, moving both the execution code and the data (variables). The target agency accepts the mobile agent, provided that it is signed by a known and approved entity. This is done with standard SSL mechanisms and X.509 signatures, i.e. the agent creator's digital signature must be present in the target agency's signature storage. When the mobile agent is allowed to execute in the target agency, it assumes the role of a server and waits for incoming requests. The requests have to include an SQL query and the aforementioned pass phrase to prevent unauthorized data access. The mobile server agent can be removed either remotely by its owner or locally by the owner of the hosting agency. The region registry administrator can delete the agent (in fact any agent or agency) from the registry, thus making it inaccessible to other agents or agencies, but cannot physically remove or deactivate the agent. The xlSQL JDBC driver (NiLOSTEP 2004) was used to access data in MS Excel workbooks.

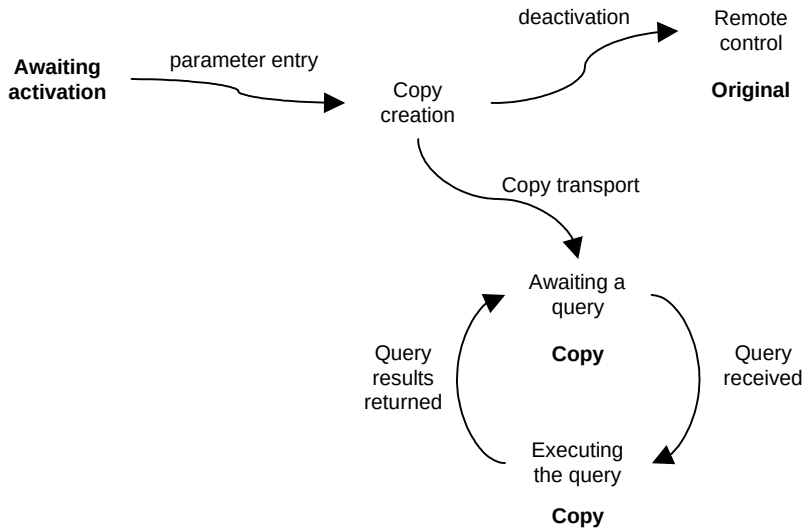

Figure 3: State transition diagram of the mobile agent

The stationary agent has the role of a intermediary between a mobile agent and the data consumer (a user or an application). The stationary agent's functionalities and services are:

- Accepting data queries,
- Forwarding a query to a mobile agent,
- Reception of results and their conversion from XML format to a CSV file, and
- Use of security mechanisms for communications.

Figure 4 shows the life-cycle of the stationary agent. The agent is started by a user that needs access to remote data. The agent is activated during its initialization and as-

sumes 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.

3 Results

With the prototype we have managed to build a system that allows us to access a remote data resource, fetch a defined range of data and convert it into the desired format. 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, and

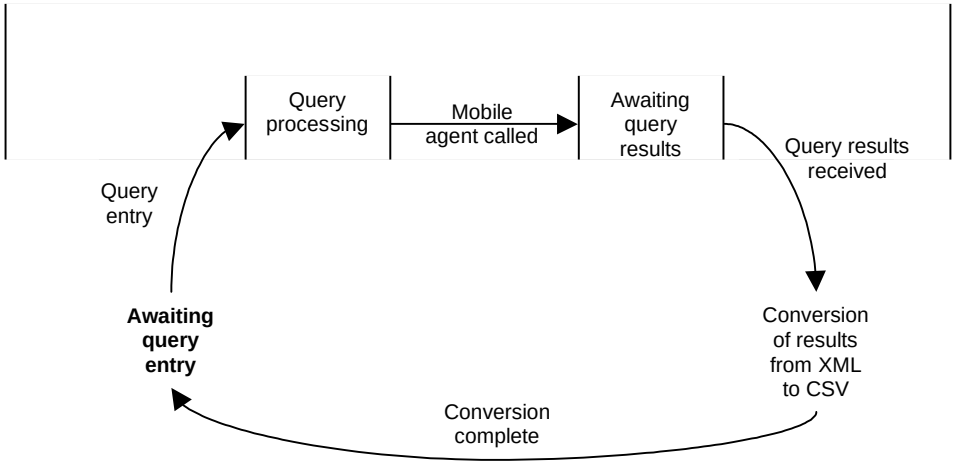

Figure 4: State transition diagram of the stationary agent

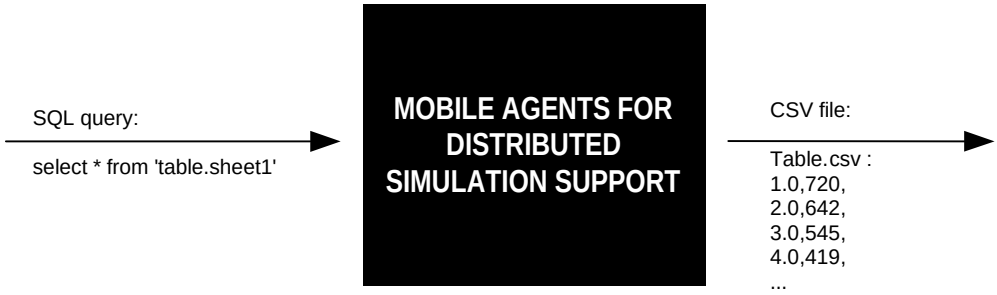

Figure 5: The software agents based system from the end-user's point of view

can be seen from the end-user's point of view as a "black box" (Figure 5) that accepts SQL queries and returns CSV format data.

We have tested the operation of the prototype using different sizes of the query results and both with and without security mechanisms. Before the testing took place, we have checked the operation of individual computers and their components: central processor units, hard drives and network adapters. Hard drive defragmentation was performed afterwards. All computers were rebooted between individual tests and the system page file initialized. As the prototype was operating on Faculty's computers the tests were conducted outside office hours to minimize undesired network traffic.

We aimed to find out how the system responds to different communication protocols (sockets and SSL) and data package sizes in order to establish the suitability of the system for different types of distributed simulation systems and hardware configurations. We were especially curious as to what would be the impact of security mechanisms on system performance.

The test environment contained three IBM PC compatible network workstations. The region registry was operating on a DELL Inspiron 8100 laptop with a Pentium 3 mobile CPU running at 1GHz and a 20Gb, 4200RPM hard drive and 512Mb of 133Mhz SDRAM (COMPUTER 3 on **Figure 2**). The mobile and stationary agents were installed on a IBM Thinkpad r50p laptop

with a Pentium M mobile CPU running at 1.7GHz and a 60Gb, 7200RPM hard drive and 512Mb of 333Mhz DDR SDRAM (COMPUTER 1 on **Figure 2**). The role of remote data source (COMPUTER 2 on **Figure 2**) was given to a machine with a Athlon 64 3000+ CPU running at 2GHz and a 160Gb, 7200RPM hard drive and 1Gb of 433Mhz DDR SDRAM. All computers were connected to the local area network at the Faculty of Organisational Sciences via Fast Ethernet (100Mbps) network adapters and 100Mbps switches.

We have measured the following parameters:

- Time needed for the transfer of a mobile agent between two agencies, depending on the communication protocol and sequence of transfer,
- Time needed to establish the connection between xlSQL JDBC driver and the data source (MS Excel workbook) depending on the MS Excel workbook size,
- xlSQL JDBC driver's usage of memory depending on MS Excel workbook size,
- Time needed for xlSQL to return a data range depending on the query result (data range) size,
- Time needed for the conversion of a returned data range to XML data depending on the query result (cell range) size,
- Time needed for the conversion of received XML data to a CSV file depending on the query result (cell range) size,
- Time needed for the completion of a SQL query (from the entry of query to the writing of a CSV file) depending on the query result (cell range) size.

The MS Excel workbook contained a table with three columns, containing the record index, decimal value and the time of record creation. We have used tables that ranged in size from 100 to 65.500 rows (maximum supported size in the MS Excel in MS Office 2000). The stationary agent used an SQL query that returned a range of cells at a randomly selected position in the table. The queries were one hundred times to provide better accuracy of results. An example of a query that returned one hundred records is as follows:

```
select idx,val from 'table.sheet1' where idx>100 and idx<201
```

An individual record returned contained two pieces of data: the record index (64 bits) and record value (64 bits). The Y7 version of the xlSQL JDBC driver was used. The times were measured using the system clock in the Java Virtual Machine.

In the following part of the paper we present some of the more interesting results of prototype testing. The results of all tests are available in the doctoral thesis (Rodič 2004). Figure 10 shows the dependence of system performance on the size of cell range that a query returns. The performance is expressed as throughput - the average number of records processed per second. We have measured the performance at the client agent side by measuring the duration of processing from the entry of an SQL query to the writing of a CSV file. The processing 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 conversion to a CSV file. We have established that the system performance is proportional to the query result size and grows in a logarithmic fashion, and that the use of secure mechanisms for communications reduced the system performance by approximately 20 percent.

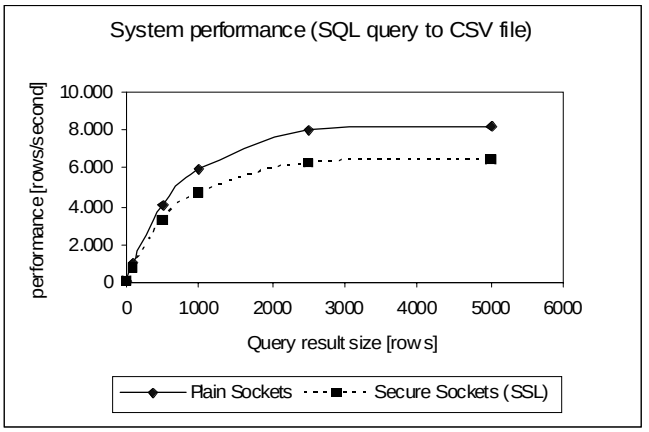

Figure 6: System performance depending on the query result size

Figure 7 shows the average duration of tasks of the mobile server agent, i.e. the execution of the received SQL query and the conversion of the resulting data range into an XML table. The accuracy of results was limited due to the limited resolution of the system clock (10 ms). It is evident that the duration of SQL queries is not seriously affected by the query result size; however the duration of conversion of the data set to XML seems to be linearly dependent on the size of query results. The query response time or latency of the mobile agent is limited by the duration of SQL query using the xlSQL JDBC driver, which was at least 60 ms during our tests.

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* format (double accuracy floating point number) 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 natural science simulation models or applica-

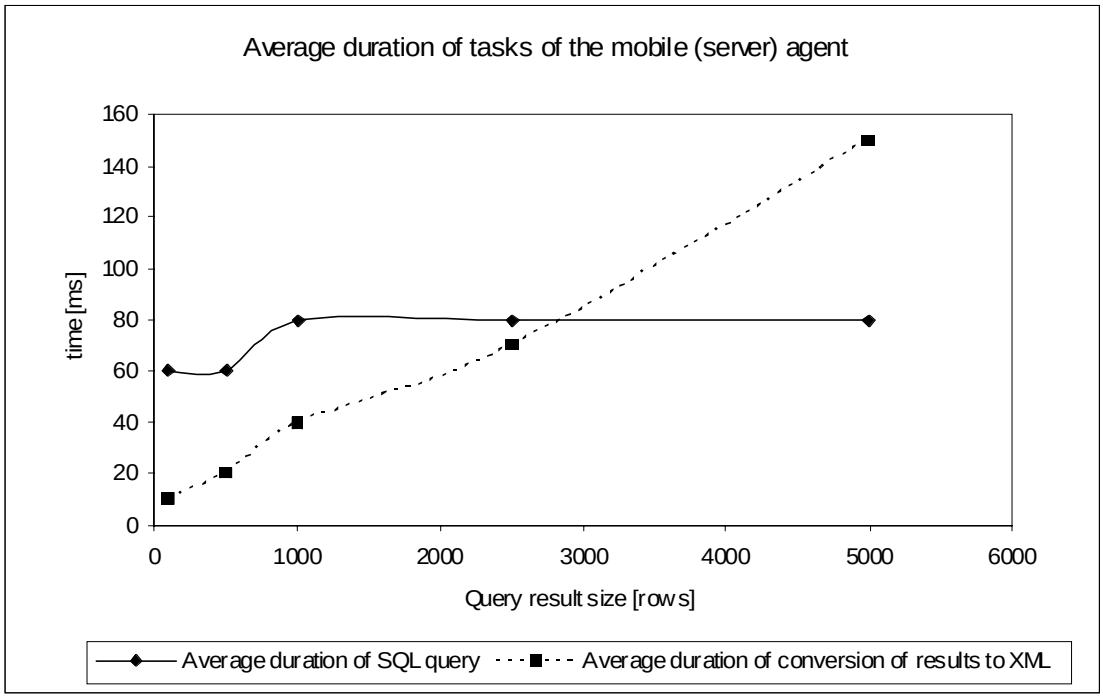

Figure 7: Average duration of tasks of the mobile agent

tions 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 20 to 25 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.

4 Conclusion

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.

We have developed two types of agents: a mobile agent that functions as a mobile server for on-demand queries in SQL and transformation of results into XML compliant documents and a stationary agent functioning as a client for query forwarding and conversion of resulting XML documents into CSV files. The mobile agent implements the methods for migration between agencies, remote control, accepting and processing of queries from remote client agents and conversion of query results to intermediary XML format (a two-dimensional table). The stationary agent acting as a client implements the methods for connecting to a mobile server agent, query forwarding and conversion of results from the XML format to CSV files. The use of an intermediary XML format for data conversion facilitates the addition of new data formats and integration with modern business information systems. By using an intermediary format we only need to develop two converters for every new data format, i.e. converters for conversion between different data formats. Without an intermediary format, converters are necessary. The use of Java, XML and a well documented agent platform facilitated the development of an open ended and expandable system. The use of XML also facilitates data processing, as XML documents can be converted to Java software objects and then modified and processed with Java code. We have used this feature to convert the XML documents to a text string and write it into a CSV file.

While 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 (therefore 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.

Acknowledgments

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.

Literature

Buss, A. & Jackson, L. (1998). Distributed simulation modelling: A comparison of HLA, CORBA, and RMI, *Proceedings of Multi-agent systems and agent based simulation: first international workshop MABS '98* (Sichman, J.S., Conte, R. & Gilbert, N., Editors), Paris July 2nd-8th 1998, pp. 819-825, Springer, Berlin Heidelberg.

Carson, J. (2000). Simulation in the future, panel discussion, *Proceedings of the 2000 Winter simulation conference* (Joines, J.A., Barton, R.R., Kang, K., & Fishwick, P.A., Editors), Orlando December 10th-13th 2000, pp. 1568-1576, IEEE, Piscataway.

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

Corbin, M. (1998). Applications of mobile agents and ambassadors in distributed simulation, *Simulation Practice and Theory*, **6**(6), pp. 505-532.

Dahmann, J.S., Fujimoto, R.M. & Weatherly, R.M. (1998). Simulation software component architecture for simulation-based enterprise applications, *Proceedings of the 1998 Winter simulation conference* (Medeiros, D.J., Watson, E.F., Carson, J.S., & Manivannan, M.S., Editors), Washington D.C. 13th-16th December 1998, pp. 797-804, IEEE, Piscataway.

Davis, W. J. & Moeller, G. L. (1999). The high level architecture: is there a better way?, *Proceedings of the 1999 Winter simulation conference* (Farrington, P.A., Nembhard, H.B., Sturrock, D.T., & Evans, G.W., Editors), Squaw Peak 5th-8th December 1999, pp. 1595-1601, IEEE, Piscataway.

FIPA (2004). http://www.fipa.org, (Accessed 01.09.2004).

Harrell, C.R. & Hicks, D. (1998). Simulation software component architecture for simulation-based enterprise applications, *Proceedings of the 1998 Winter simulation conference,* (Medeiros, D.J., Watson, E.F., Carson, J.S., & Manivannan, M.S., Editors), Washington D.C. 13th-16th December 1998, pp. 1717-1721, IEEE, Piscataway.

IKV++ (2004). http://www.grasshopper.de, (Accessed 20.10.2004).

Jen-Yen, C.J. & Shih-Wei, S. (2003). AgentGateway: a communication tool for multi-agent systems, *Information Sciences - Informatics and Computer Science: An International Journal*, **150**(3-4), pp. 153-164.

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

Kuljis, J. & Paul, R.J. (2001). An appraisal of web-based simulation: whither we wander?, *Simulation Practice and Theory*, **9**(1-2), pp. 37-54.

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

Mangina, E. (2002). The 'draft' review of software products for multi-agent systems, Report for the European Network of Excellence for Agent Based Computing, Applied Intelligence (UK) Ltd., Kenstone, Available from http://www.agentlink.org/admin/docs/2002/2002-47.pdf (15.09.2004).

Mills-Tettey, G.A., Johnston, G., Wilson, L. F., Kimpel, J.M. & Xie, B. (2002). The ABELS System: Designing An Adaptable Interface For Linking Simulations, *Proceedings of the 2002 Winter simulation conference* (Yucesan, E., Chen, C.H., Snowdon, J. L., & Charnes, J. M., Editors), San Diego 8th-11th December 2002, pp. 832-840, IEEE, Piscataway.

NiLOSTEP Information Sciences (2004). http://nilostep.com, (Accessed 20.10.2004).

OMG (2005). http://www.omg.org, (Accessed 01.09.2005).

Powersim Software AS (2004). http://www.powersim.com, (Accessed 10.03.2004).

ProModel Corporation (2002). http://www.promodel.com, (Accessed 16.02.2002).

Qi, H., Iyengar, S. & Chakrabarty, K. (2001). Distributed multiresolution data integration using mobile agents, *IEEE Aerospace Conference Proceedings. Vol. 3,* (Robert, P.W., Editor), Big Sky 10th-17th March 2001, pp. 1133-1143, IEEE Service Center, Piscataway.

Rodič, B. (2004). Distributed simulation support using mobile agents and XML, Doctoral Dissertation, University of Maribor, Faculty of Organisational Sciences.

Szymanski, B.K. & Chen, G. (2000). Linking spatially explicit parallel continuous and discrete models, *Proceedings of the 2000 Winter simulation conference* (Joines, J.A., Barton, R.R., Kang, K., & Fishwick, P.A., Editors), Orlando December 10th-13th 2000, pp. 1705-1712, IEEE, Piscataway.

White, J. (1996). Telescript technology: mobile agents, General Magic white paper, Software Agents, MIT Press.

Blaž Rodič received his B.Sc. degree in electronics and telecommunications engineering from the University of Ljubljana, Faculty of Electrical Engineering in 1996 and the M.Sc. and Sc.D. degree in information systems management from the University of Maribor, Faculty of Organizational Sciences in 2002 and 2004, respectively. Currently he is working as a senior researcher at the Faculty of Organizational Sciences. His research interests include distributed simulation and application of artificial intelligence methods in simulation models.

Miroljub Kljajić - rec biography on page 518.