Performance Impact of Security Mechanisms on Java Mobile Agents for Data Retrieval

Blaž Rodič
Faculty of information studies
Sevno 13, 8000 Novo mesto, Slovenia
{blaz.rodic@fis.unm.si}

Abstract. *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.*

The main focus of this article is on the performance of Java based mobile agents using cryptography and data format translation via an intermediary XML (eXtensible Markup Language) format.

We have tested the performance of the agents in a distributed simulation scenario and established that while some data access methods may limit the performance of the system, the agents can be used to connect heterogeneous simulation models and other applications, improving their connectivity and usability.

Keywords. mobile agents, XML, data filtering, data retrieval, distributed simulation

1 Introduction: the problems of accessing remote data

In the last ten years we have witnessed an increase of research interest in distributed information systems. New technologies in networking have improved the accessibility of information, while strong cryptography allows us to transfer information more safely. Better access to data can improve the accuracy and performance of business and manufacturing simulation models used in decision support systems [7]. In turn, faster and more accurate decision support can give a company an advantage over its competition.

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. However, network connections can become a serious bottleneck of a distributed simulation [13], therefore it makes sense to plan the intensity of communications between simulation components and the placement of components according to the available network bandwidth and latency.

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 that we would like to address:

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

In order to solve or remedy these issues, 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, therefore 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 Technology review

We have examined several available technologies for development of distributed simulation systems and the related research. One of the better known and well publicized technologies for distributed simulation is the HLA (High Level Architecture) [14]. HLA is the result of advancements in high-performance computing that has allowed the military to focus on producing larger and more accurate simulations. But the weaknesses of HLA 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 [7]. The complexity of HLA leads to difficult development of models and nearly impossible integration of non-HLA simulations [13]. The evolution of HLA is however ongoing and many promising mechanisms such as fault tolerance [2] are being developed.

One of the recent developments in the area of distributed simulation is the use of web technologies such as Web Services, Javascript and CGI (Common Gateway Interface) 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 [5].

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. 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. 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 [8].

Maamar et. al. [9] describe a system intended to facilitate the use of e-commerce on mobile wireless devices. Issues involved include low bandwidth, high latency and security of transactions. In order to tackle these issues, software agents were used in the design and development of the system. Szymanski and Chen [13] have used the IBM Aglets toolkit to connect two simulations running on different platforms. 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. Their conclusion was 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. The focus of another research [12] 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%. A comprehensive introduction to data acquisition systems applying data filtering is available in [1] along with a presentation of an agent based system – the eDW (Enhanced Data Warehouse).

3 Methodology

Despite the developments in distributed systems 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 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 decided to utilize the Grasshopper V2.2.4 platform by IKV++ [3]. We have also decided to implement data format translation using basic XML tables as an intermediary format. We chose to connect the simulations at the data resource level, not at the runtime level. Runtime connections would limit the flexibility and would require a fast execution of code, something that Java still does not deliver. Advantages of data resource connections are shorter and simpler connection setup procedures and an open-ended structure.

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

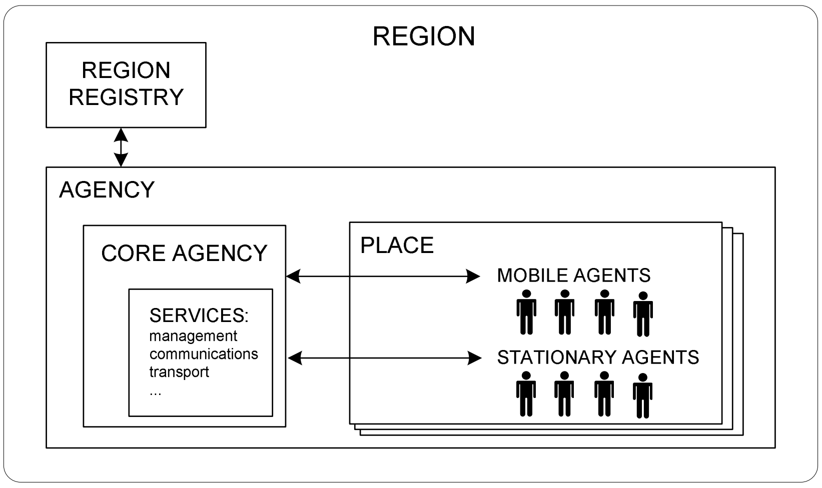

Fig. 1. Structure of a Grasshopper based agent system

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 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. More details about the platform can be found at the developer's website [3] and 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 [6]. 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 2003 [10] acted as the data server, while the DES model running in ProModel [11] 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. 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 in ProModel is via delimited text files, e.g. 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 separated values) files. While it is possible to save data from MS Excel as CSV files, it is very difficult to perform this from a remote mobile terminal. 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.

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) provided in Java 2 SDK version 1.4.2.03, downloaded from http://java.sun.com. The xlSQL JDBC driver [4] was used to access data in MS Excel (Microsoft Excel) workbooks accessed. The xlSQL JDBC driver is an open source project with a GPL license (GNU Public

License). An alternative was to use JDBC via the Microsoft ODBC, however that would require a Microsoft platform to run, limiting the portability, and would be a slow and cumbersome solution.

While it is possible to use xlSQL to write CSV files via MS Excel, xlSQL requires 4 Mb of memory to run and an installation of MS Excel to access the files, and thus cannot be used on many mobile devices. 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 to translate from XML to CSV files.

1.1 Prototype system structure

We have divided the distributed system into several components, shown on Fig. 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).

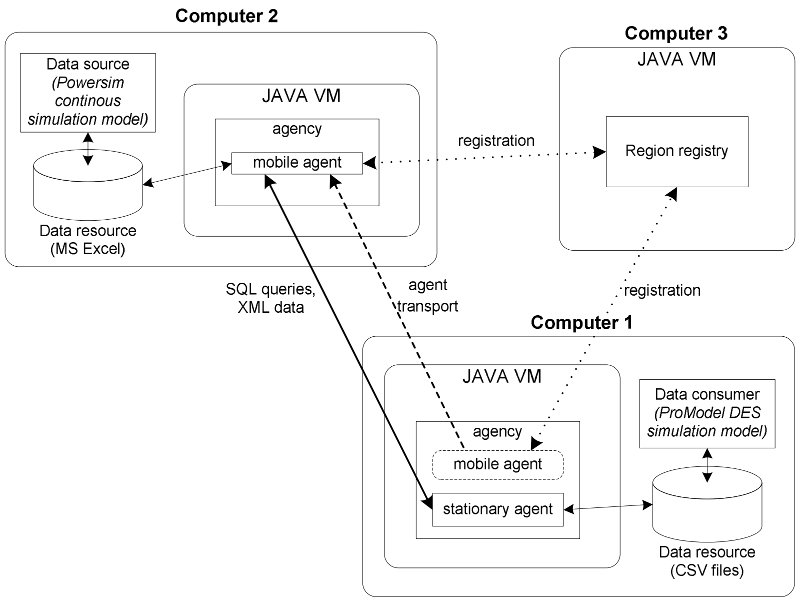

Fig. 2. Prototype deployment diagram

The prototype of a distributed system contains three computers that host individual components of the system (Fig. 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. 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, "Computer 3" holds the region registry, which is used to control and administrate the agents and agencies.

A mobile agent 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 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. 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.

5 Performance of the MA system

We have tested the operation of the prototype using different sizes of the query results and both with and without security mechanisms. Tests were executed 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 types of distributed simulation systems and hardware configurations.

The test environment contained three IBM PC compatible network workstations. The region registry was operating on a Windows 2000 SP3 system, 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 Fig. 2). The mobile and stationary agents were installed on a Windows XP SP2 system, a IBM Thinkpad r50p laptop with a Pentium M CPU running at 1.7GHz and a 60Gb, 7200RPM hard drive and 512Mb of 333Mhz DDR SDRAM (Computer 1 on Fig. 2). The role of remote data source (Computer 2 on Fig. 2) was handled by another Windows XP SP2 system, a desktop machine with an Athlon 64 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 via Fast Ethernet (100Mbps) network adapters and 3Com 100Mbps Ethernet switches.

The software used in the experiment was MS Excel 2000, xlSQL version Y7, Grasshopper V2.2.4, Powersim Studio 2003 and ProModel version 5.0.

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 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 the query to the completed transfer of results to the CSV file) depending on the size of the results.

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 SQL query returned a range of cells at a randomly selected position in the table.

An individual record returned contained two pieces of data: the record index (64 bits of data) and record value (64 bits of data). The times were measured using the system clock in the Java Virtual Machine. The accuracy of time measurements was limited due to the 10 ms resolution of the system clock.

Fig. 3 shows the dependence of system performance on the size of a cell range that a query returns. 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.

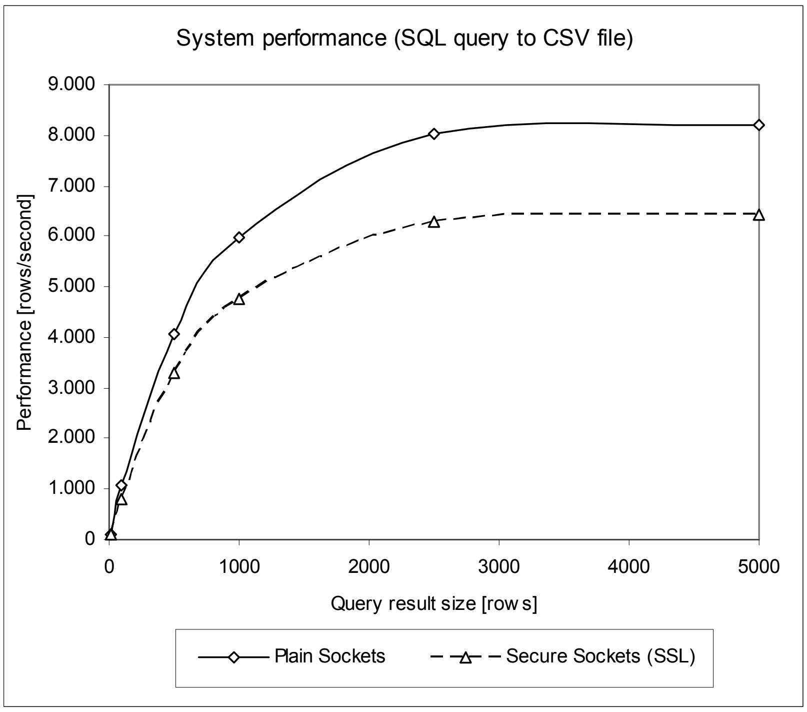

Fig. 3: System throughput depending on the query result size

We have divided the tasks of agents in two parts: first, the execution of an SQL query and the conversion of query results into XML and second, transforming XML data into a CSV file. The former is performed at the remote location by the mobile server agent, while the latter is performed by the client agent. 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 on Fig. 4. 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. We should note that the system clock resolution (10 ms) limited the accuracy of results, as the conversion from XML to CSV took less than 10 ms for one-row queries and about 100 ms for 5000-row queries.

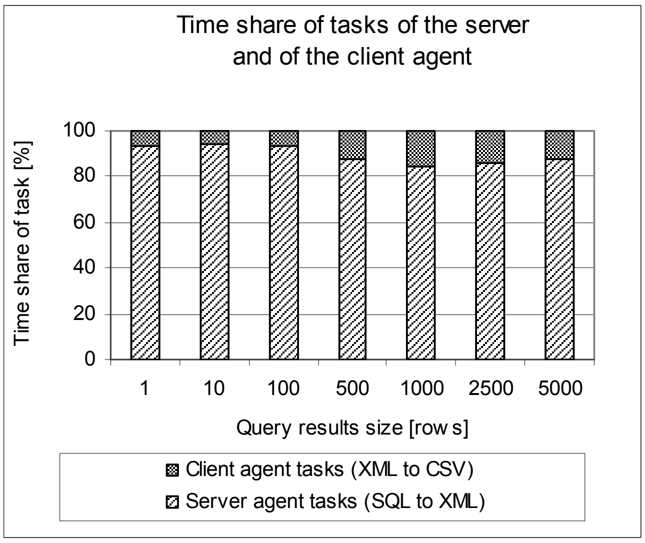

Fig. 4: Time shares of the tasks of a server agent and a client agent

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. Fig. 5 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 Fig. 5, the duration of SQL queries is affected by the query result size only for query result sizes of under 1000 rows, while 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.

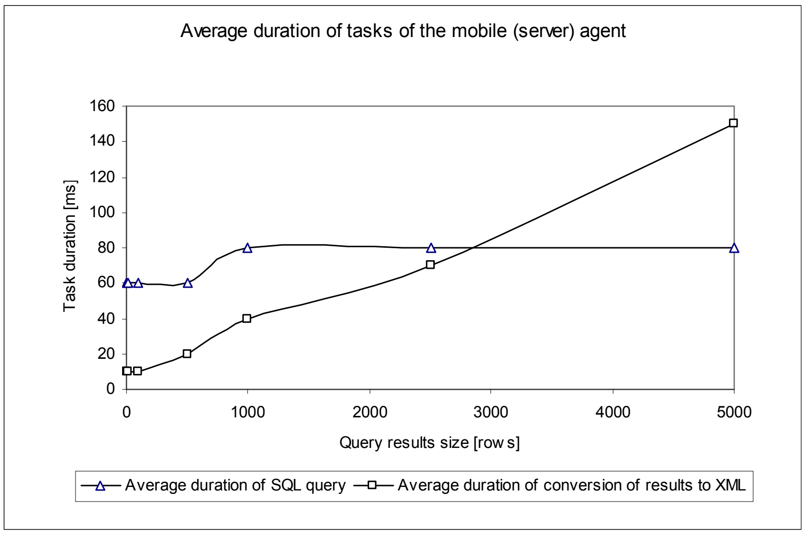

Fig. 5: Average duration of tasks of the mobile agent

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.

6 Conclusions and future research

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.

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.

However, several limitations exist within this system. The selected data access driver (xlSQL) introduces latency that limits the real-time application of the agents. Also, data retrieval could be automated with the introduction of data-broker 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 handheld devices running Java.

8 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

[1.] Abramowicz W., Kalczynski P., Wecel K.: Filtering the Web to Feed Data Warehouses, Springer-Verlag, Berlin Heidelberg New York (2002)

[2.] Chen D., Turner J.T., Cai W.: A Framework for Robust HLA-based Distributed Simulations, In: Turner J.T., Luethi J. (eds.): Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation - PADS 2006, IEEE Computer Society, Washington D.C. (2006) 183-192

[3.] IKV++: http://www.grasshopper.de, 20.10.2004

[4.] JAVA.NET: https://xlsql.dev.java.net/, 3.5.2006

[5.] Karagiannis P., Markelis I., Paparrizos K., Samaras N., Sifaleras A.: E-learning technologies: employing Matlab web server to facilitate the education of mathematical programming, International Journal of Mathematical Education in Science and Technology, Vol. 37, No. 7/15 (2006) 765-782

[6.] Kljajic, M., Bernik, I., Skraba, A.: Simulation Approach to Decision assessment in Enterprises, Simulation, Vol. 75, No. 4 (2000) 199-210

[7.] Kuljis, J., Paul, R.J.: An appraisal of web-based simulation: whither we wander?, Simulation Practice and Theory, Vol. 9, Nr. 1-2 (2001) 37-54

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

[9.] Maamar Z., Yahyaoui H., Mansoor W.: Design and Development of an M-Commerce Environment: The E-CWE Project, Journal of Organizational Computing and Electronic Commerce, Vol. 14, No. 4 (2004) 285-303

[10.] Powersim Software AS: http://www.powersim.com, 10.03.2004

[11.] ProModel Corporation: http://www.promodel.com/, 16.02.2002

[12.] Qi, H., Iyengar, S., Chakrabarty, K.: Distributed multiresolution data integration using mobile agents, In: Robert P.W. (ed.): IEEE Aerospace Conference Proceedings, Vol. 3, IEEE Service Center, Piscataway, NJ. (2001) 1133-1143

[13.] Szymanski, B.K., Chen, G.: Linking spatially explicit parallel continuous and discrete models, In: Joines J.A., Barton R.R., Kang K., Fishwick P.A. (eds.): Proceedings of the 2000 Winter simulation conference, The Society for Computer Simulation International, IEEE, Piscataway NJ (2000) 1705-1712

[14.] U.S. Department of Defense, Defense Modeling and Simulation Office, https://www.dmso.mil/public/transition/hla/, 1.12.2006