PROCESSING REMOTE MEASUREMENT DATABASES BY THE MEANS OF MOBILE AGENTS

Steffen Kernchen, Dmytro Rud, Fritz Zbrog, Reiner R. Dumke
Department of Computer Science, Otto-von-Guericke-University of Magdeburg, Magdeburg, Germany

Keywords: Agent Technology, Measurement Database, Remote Processing.

Abstract: This paper discusses possible approaches for using mobile agents for accessing and processing of big remote databases containing measurement data.
There are several reasons that make it inappropriate to dispatch raw data over the network. (a) Those data may be confidential and therefore need to be anonymized. (b) The amount of the data can be too big. In this case it could be advisable to generalize the data on the database provider node and to transfer only these generalization results to the customer. (c) Furthermore, this special information processing can be not in line with qualifications of both the database holder and its clients. The usage of agents gives the possibility to outsource the development and installation of additional software to an external agent provider, which is then responsible for regular updates of the software considering aspects like new algorithms.
Application scenarios are outlined and certain kinds of service will be described and motivated with examples.

1 INTRODUCTION

Measurement databases can be adopted in many fields of human activities. Typical examples are weather forecasting, software quality assessment, and production control. A common trait of measurement databases is their potential for the organization doing measurements and collecting the data, and also for third-party users. The database-holding organization can profit from giving access to its database.

Therefore the question about mechanisms of such access arises. Many functional and non-functional aspects must be taken into account for development of these mechanisms e.g. performance, security, format conversion, data generalization, etc.

In this paper we propose our solution. The idea is to use mobile agents, which act on the side of the database on behalf of third-party database users.

To talk about agents and to use them it is necessary to understand what they are. Based on a classic definition of agents we define agents as software components with certain properties (Wooldridge and Jennings, 1994). Autonomy, social competencies, reactivity and pro-activity are indispensable. Additional properties can be mobility, collaboration and the ability to learn. Agents that will be described in this paper will have autonomous characteristics, because they will be able to perform their tasks without direct supervision of humans or other agents. An important aspect will be the assurance of applicability in differentiated environment through interactive adaptation mechanisms. Perceiving changes in the environment and performing reactions leads to a reactive agent. This aspect can become important because of potential changes of the data to be processed. Proactivity describes a goal-directed behaviour.

There are several reasons that make it inappropriate to dispatch raw data over the network:

- The data may be confidential and therefore need to be anonymized.
- The amount of the data can be too big. In this case it could be advisable to generalize the data on the database provider node and to transfer only these generalization results to the customer.
- Furthermore, this special information processing can be not in line with qualifications of both the database holder and its clients.

Kernchen S., Rud D., Zbrog F. and R. Dumke R. (2007).
PROCESSING REMOTE MEASUREMENT DATABASES BY THE MEANS OF MOBILE AGENTS.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 184-188
DOI: 10.5220/0001271601840188
Copyright © SciTePress

The usage of agents gives the possibility to outsource the development and installation of additional software to an external agent provider, which is then responsible for regular updates of the software considering new algorithms etc.

These ideas will be illustrated by a practical example. We will introduce an agent-based system for processing a database containing web service performance measurement results.

The rest of the paper is organized as follows: section 2 gives a review of available related work, section 3 represents the core of our paper and outlines concepts and potential use cases of integrating agents in existing distributed software infrastructures. Section 4 presents practical aspects of our implementation and section 5 is devoted to conclusions and proposals regarding subsequent work.

2 RELATED WORK

The following existing approaches resemble our ideas.

Using agents as mediators for human database access is a common idea. They act as proxies to create queries for database access ((Masuoka and Ohtani, 1999), (Elio et al., 2000)). That is motivated by assumptions like (a) highly complex database design so it is not possible for the human agent to specify a single, simple database query for which there is one single answer; (b) vague set of constraints when starting the search task, (c) multiple search goals, or (d) the interface is not visual or requires direct manipulation.

The realization of distributed databases is another important role for agents in this context. While working towards a certain goal, they may exploit concurrency, parallelism and distribution to thereby bequeath those functionalities to databases (Kirchberg, 2006).

Agents themselves include databases. They must carry a view of their environment to be able to make decisions what can be seen as a local database. Obtaining a durable view over time and across disturbances is essential for agent success. Thereby multiagent systems form highly distributed databases (Lockemann and Witte, 2005).

3 PROPOSED APPROACH

3.1 Involved Parties

In our model the following parties exist. The data holder owns a large data amount that is to be processed. On the data holder side there are an interface for interaction with the agent provider, a service for configuration of arrived agents and the database mentioned above. The functionality necessary for processing the database can be obtained from the agent provider in form of an appropriate agent. Figure 1 shows interaction steps of the components in the architecture proposed by the authors. These components will be discussed in detail below.

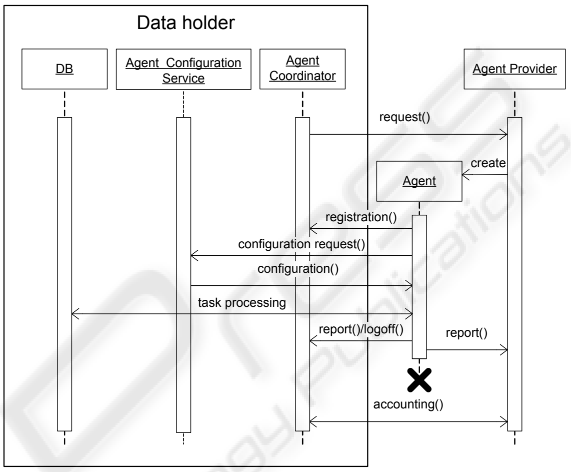


Figure 1: Interaction flow of involved parties.

3.1.1 Data Holder

The interface of the agent coordinator is dedicated for interaction with agent providers and their agents. It initializes the interaction, registers the generated agents and carries out management tasks. Special ontologies could be needed to solve possible interoperability issues. It is important to standardize relevant vocabulary and message exchange patterns used for request to agent provider, registration of agents and accounting process (Grütter, 2005). The Web Ontology Language (OWL) (W3, 2004) could be appropriate for that purpose.

Providing a uniform database accessing mechanism is the most important task of the agent configuration service, because there might be different databases and database schemes. This service supplies necessary information about access credentials and addresses of tables and views which need to be processed. Furthermore the agent gets in that way a possible location for storing intermediate results.

3.1.2 Agent Provider

The web service located on the side of the agent provider takes orders of the clients and instantiates appropriate agents for autonomous data processing.

Other tasks are the accepting of reports and accounting. Non-functional properties of both the provider's interface and the delivered agents could be described using special ontologies.

3.1.3 Agent

The agent is the main working component in the proposed architecture. It is instantiated by the agent provider and should support all requested features specified in the clients order. A modular construction system and a predefined skeleton constitute the foundation of the generation process. After migration the agent registers itself on the agent coordinator and is being configured for database access. Now it is ready to perform its tasks. Finally it sends reports to the agent coordinator and the agent provider as a basis for service accounting.

Security aspects such as safe code distribution and sandbox-based execution should be respected by implementing the infrastructure. To avoid espionage network interaction is minimized, because main communication takes place on the data holders side. Furthermore we propose usage of secure transport protocols and authentication mechanisms.

In the two following subsections we will present possible application scenarios.

3.2 Acquisition of Third-party Know-how

The example in the introduction can be extended to several fields of application. In the scenario presented here, the large amount of data on the data holder side needs knowledge and know-how for analysis. These algorithms can be subject to change or proprietary. The proposed architecture makes the data holder independent of procurement and update of monolithic resource-consuming software products or expensive in-house development. Time and effort for development, optimization and maintenance is outsourced to specialized providers. In that way service evolution is simplified, too.

The further advantages of using agents are minimized network load and increased security no confidential information must be dispatched.

Other possible fields of application of agents in this case are knowledge discovery in self-contained data volumes, automatic tax consultancy or the extension of the internal knowledge base of the agent or its providing service in a generalized and anonymized manner.

3.3 Case Study: Web Services Measurement Service

In this section we will outline another possible scenario of integrating software agents in a web service-based distributed infrastructure.

The web service measurement service Wesement, being operated by the authors of the present paper as part of their Web Service Trust Center research project ((Schmietendorf and Dumke, 2005), (Schmietendorf et al., 2004), (Rud, 2005)), is collecting performance statistics of third-party web services. Any web service provider can use Wesement for long-term availability, metadata stability and performance measurement of its services at no-cost. Collected statistics is being persisted in a relational database. The only available means go generalize the data in the web-based Wesement frontend is to represent it graphically in form of per-day or per-month diagrams. That is unambiguously insufficient, because:

- Another presentation form for example a web service provider, simply wants to get notified whether its web services are performing well on a given day (i.e. to get a boolean value instead of a PNG image) or to receive a list of services whose performance in this month has fallen below the preconfigured limit.
- Another time granularity some providers may want to get weekly or yearly reports in addition to daily or monthly ones.

A possible solution of this problem would be to change the measurement service frontend itself. But this way is tied with enormous maintenance effort, because the frontend will need to be customized almost every time a new provider comes and registers its services. Another possibility for the providers is to get raw statistics (e.g. in XML format) and process it locally.

In our opinion, the most appropriate solution is to use agents, which work on the Wesement's side and thus have local access to the statistics database. Agent technology provides the highest degree of productivity, because it is the only possibility for clients to extend the infrastructure of the server. By this they are able to remotely perform their tasks in direct adjacency to the possible source of events. Results of their calculations can be then sent to the corresponding web service providers. It is also advisable that agents work continuously and keep track of services' performance changes in "real-time" if a negative trend is detected, the provider can be notified early enough.

The next question is, where these agents can be taken from. Below we outline three possible answers and discuss their advantages and disadvantages:

- Agents are implemented and provided by the measurement service's operator, i.e. by us. In this case we have roughly the same development effort as if we were changing the Wesement frontend, but the approach differs instead, we develop some kind of plug-ins for the backend.
  - Advantages: (a) possibility to reuse such agents for many web service providers; (b) no security risks and low agents configuration effort.
  - Disadvantages: Wesement is a research project, and we have neither interest nor resources to develop the agents.
- Agents are implemented and sent to the measurement services side by the web service providers.
  - Advantages: the providers have the full control of how their agents behave.
  - Disadvantages: (a) reuse of agents between many providers is impossible; (b) same as in previous case, the provider might not have all necessary resources to develop and manage the agents.
- Full division of labour a third party works on behalf of web service providers and deals with development and management of agents.
  - Advantages: (a) possibility to reuse agents for many web service providers; (b) possibility for providers to select agents from many offerings.
  - Disadvantages: management efforts.

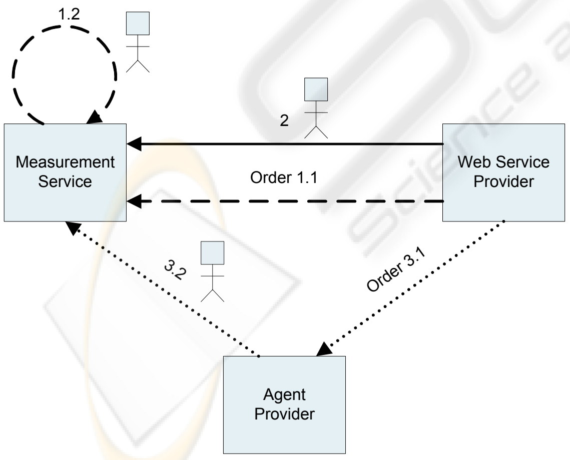


Figure 2: Agent deployment scenarios.

Figure 2 shows the proposed approach. Interaction steps of the three possible scenarios discussed above are numbered appropriately.

The described approach can increase flexibility of the service measurement infrastructure.

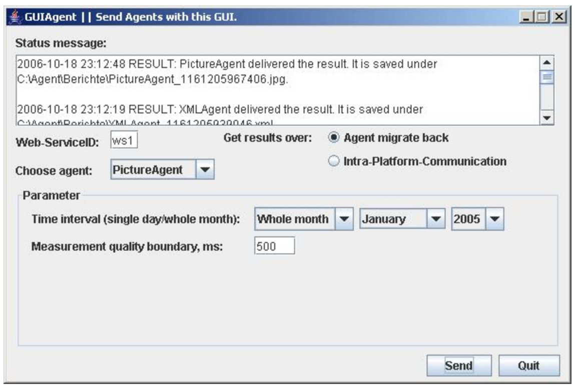
4 AGENT-BASED PROCESSING OF MEASUREMENT DATABASES

In this chapter we present the most important aspects regarding the implementation of agents for remote processing of measurement databases. As aforementioned we used the database of our existing service Wesement containing performance statistics of web services.

For the execution of agents an infrastructure is needed. We have choosen JADE as a java-based agent platform to provide normative services like life cycle management, white pages service, yellow pages service and message transport service as defined by the according FIPA standard for agent platforms. Additional services are agent-software integration, an ontology service and human agent interaction.

Based on this technology we created our multi-agent system. Therefore we launched one JADE platform with several containers. They are used to logically separate agents and are often referred as "agent cities" in literature. In JADE they can be distributed on several computers. In our case one container resides on the same computer as the database to allow its remote processing and to guarantee the aforementioned advantages. The others are distributed.

Entry point for agent creation is the ClientAgent. It is launched with the JADE platform and presents a Graphical User Interface (GUI) to the user. The main objectives of this agent are the selection of an appropriate processing agent type as well as the definition of required parameters. The ClientAgent is able to detect available processing agent definitions and presents parts of their corresponding GUI's. Thereby separation of ClientAgent and processing agent functionalities is guaranteed. Each of the later agents provides an own GUI component for the definition of necessary parameters. Figure 3 visualizes the ClientAgent with a graphical component of a picture agent.


GUIAgent | | Send Agents with this GUI.

Status message:

```
2006-10-18 23:12:48 RESULT: PictureAgent delivered the result. It is saved under
C:\Agent\Berichte\PictureAgent_1161205967406.jpg.

2006-10-18 23:12:19 RESULT: XMLAgent delivered the result. It is saved under
```

Web-ServiceID: ws1

Get results over: [✓ Agent migrate back] Intra-Platform-Communication

Choose agent: PictureAgent

Parameter

Time interval (single day/whole month): Whole month January 2005

Measurement quality boundary, ms: 500

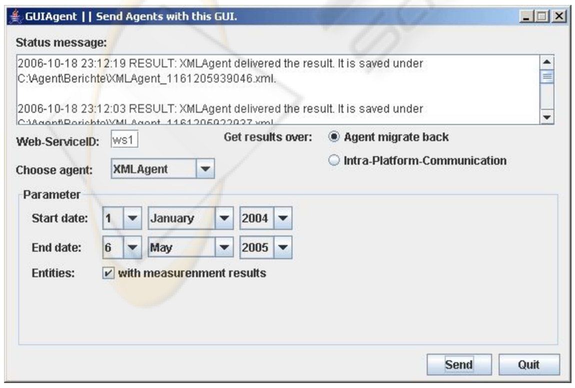
Figure 3: Sending a PictureAgent for remote processing.

Parameters common to all agents are the web service ID, the result delivering definition and the selection of the processing agent. Therefore these aspects are integrated into the ClientAgent in addition to a window component presenting status information and locations where the results of the processing agents are locally stored. By the web service ID the data of the web service to be analyzed can be clearly identified. Our approach takes benefits from the advantages of mobile agents. So it is possible to define result delivering by intra-platform communication as well as by agent migration.

The remote container accommodates an additional agent with proxy functionalities. It provides database accessing information for the processing agents and is intended to be extended by safety mechanisms to restrict database access to secure agents. All agent communication acts are ontology-based. Thereby the possible conversation content is defined and the architecture is extendible towards the usage of additional web service measurement services. From a technical point of view this ontology needs to be mapped to Java. Classes realize the underlying communication in the multiagent system.

The presented PictureAgent needs additional parameters about the time interval to be observed and a measurement quality boundary. As a result it creates a graphical visualization presenting performance statistics of third-party web services.

We exemplary implemented two more processing agent types. The ObserverAgent periodically (e.g. weekly, monthly or self-defined) checks the state of the web service for a predefined time and sends back alert messages in case of state changes. The XMLAgent creates XML documents containing the performance data of the selected web service. They can be used for further processing. Its graphical component is shown in figure 4.



Status message:

```
2006-10-18 23:12:19 RESULT: XMLAgent delivered the result. It is saved under
C:\Agent\Berichte\XMLAgent_1161205939046.xml.

2006-10-18 23:12:03 RESULT: XMLAgent delivered the result. It is saved under
```

Web-ServiceID: ws1

Get results over: [✓ Agent migrate back] Intra-Platform-Communication

Choose agent: XMLAgent

Parameter

Start date: 1 January 2004

End date: 6 May 2005

Entities: ✓ with measurenment results

Figure 4: Sending a XMLAgent for remote processing.

5 CONCLUSIONS AND FURTHER WORK

We have analyzed possible scenarios of accessing remote measurement databases using mobile agents. We outlined also different implementation approaches and discussed their advantages and disadvantages. Security issues were mentioned too. An example illustrated the effectiveness of the proposed approach.

We plan to continue our research and to extend the infrastructure functionality for automated interaction in a web service-oriented manner by the definition of appropriated access interfaces. Another aspect of our further work is the extension towards accessing multiple databases.

REFERENCES

Elio, R., Haddadi, A., and Singh, A. (2000). Task models, intentions, and agent conversation policies. In *Proceedings of the Pacific Rim Conference on AI (PRICAI-2000)*, Lecture Notes in Artificial Intelligence 1886, pages 394–403. Springer Verlag.

Grütter, R. (2005). Software Agents in Semantic Web. (in German). *Informatik Spektrum*, (6):1–11.

Kirchberg, M. (2006). Dbaa/acl - a database agent architecture and communication language. In Isaas, P., McPherson, M., and Bannister, F., editors, *Proceedings of the IADIS International Conference e-Society 2006*, pages 244–249. IADIS Press.

Lockemann, P. C. and Witte, R. (2005). Agents and databases: Friends or foes? In *Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS05)*, pages 137–147.

Masuoka, R. and Ohtani, T. (1999). Agent description ontology. Proposal to the 13th meeting of FIPA. Seoul, Korea.

Rud, D. (2005). Methods for Quality Assessment of Web Service Offerings (Diploma thesis, in German). Master's thesis, Otto-von-Guericke-University of Magdeburg.

Schmietendorf, A. and Dumke, R. (2005). A Measurement Service for Monitoring the Quality Behaviour of Web Services offered within the Internet. In *Proceedings of IWSM/MetriKon 2004*, pages 381–390. Shaker-Verlag Aachen.

Schmietendorf, A., Dumke, R., and Reitz, D. (2004). SLA Management Challenges in the Context of Web-Service-Based Infrastructures. In *Proceedings of the IEEE International Conference on Web Services (ICWS 2004)*, pages 606–613, San Diego, USA.

W3 (2004). OWL Web Ontology Language.

Wooldridge, M. and Jennings, N. R. (1994). Agent Theories, Architectures, and Languages: A Survey. In *ECAI Workshop on Agent Theories, Architectures, and Languages*, pages 1–39.