PROSECO - A Framework Architecture to Provide Services for Economic Data

Miriam Lux, Ralf Stuckert, Stefan Müller

Fraunhofer Institute for Computer Graphics
Rundeturmstr.6, D-64283 Darmstadt, Germany
{mlux, stefanm}@igd.fhg.de

ABSTRACT

The Internet now provides access to a number of repositories from which large data volumes can be available. Due to the huge amount of memory needed to store the data and hardware requirements to visualize them, there is a need for interactive visualization services for online data exploration and analysis. Therefore, we have investigated the requirements for a user-friendly access to information. Based on the resulting WWWW model, we have developed the system PROSECO, an open scalable framework architecture to easily provide and access any services needed for a distributed visualization of abstract data. We demonstrate the capabilities of our concepts by integrationg services in-to PROSECO to provide the workflow of an economist.

Keywords: User-friendliness, distributed visualization, economic data.

1 INTRODUCTION

Nowadays, the Internet has become one of the most important means of communication. Every user -consumers, enterprises, even the government- benefits from this new medium and from the possibility of being independent of location and time for all applications. The Internet can be regarded as the virtual global market-place. This changed status of the Internet calls for an adaptation of all components relating to it. The problems to overcome in finding relevant information have shifted, the difficulty previously being that information was not easily accessible or searchable. In order to make full use of the Internet's potential, a user-friendly access to information needs to be developed, in particular for the limitation of the enormous amount of data and for the optimal distribution of information and knowledge.

A representative set of data where a user-friendly access is necessary can be seen in economic data [Lux00]. In general, economic data is built of a complex network interconnecting arbitrary data types. Besides the different data types, these interconnections are fundamental for the overall understanding of the domain. During the visualization process, not only the huge amount of data, but also its' multidimensionality create various problems. The same holds for the users of a visualization tool for this kind of data. No assumptions can be made concerning their background knowledge or their analysis goals, even the available infrastructure, especially the graphics hardware is not known in advance. These preconditions require a system to be built as a set of independent sub- systems, each targeting one or a few possible combinations. Given these requirements, the resulting system must not be a monolithically organized system based on a main central component. Instead, it should be a set of peripherally organized components interconnected in a loose manner.

This work defines an open, scalable framework to visualize economic data in a distributed environment. By defining a general model for a service, it is possible to distribute application or data sources in an abstract way. This level of abstraction enables their uniform integration in-to new systems. Available resources can be integrated at any time to enhance the performance of the whole system. This framework is the basis for enabling any user to analyze any information in an intuitive and effective way at any place or time.

2 RELATED WORK

The Visualization Web Server system [WoBrWr96] introduces a judicious compromise between the hardware requirements on the client side and the possibilities for interaction. The server allows client interactions with the filtering and mapping level and yields a VRML file. With this constellation the application on the client side can be realized as an HTML document including the VRML scene. Here, possibilities for load balancing through the server are missing. Consequently, the requests for visualization can not be distributed to different servers. Also the distribution of the visualization pipeline is fixed on variation. The security of such a system is another important aspect. Absent mechanisms for an access restriction reduce this system to work only with non-critical data.

The viscWeb system [LePi99] has some common

ideas regarding the above described system. Here, the user can submit HTML forms and receives images or VRML descriptions, that are visualized by the browser itself or an external application. The distribution of the servers among the machines is managed by a special program called General Manager. In addition, on every machine a Local Manager is running, that provides services. Communication is done directly between the CGI scripts executed on the machine supporting the Web service, and the visualization servers executed on remote machines. This architecture supportes the use of a pool of visualization servers, that is transparent to the user. The General Manager chooses a machine available for the requested services as a function of various parameters, including the machine charge and the number of users that are currently logged on. Therewith, the system is scalable, however, if the General Manager -as an exclusive resource- fails, the whole system fails. The system is able to perform user sessions, where at least one visualization server is dedicated to the user. However, this features is not sufficient to fulfill the requirements from various users. Possibilities to choose the distribution of the visualization pipeline are missing. Besides, there is no simple method to integrate the resources that are already available. Even an access restriction is not considered.

Weather on demand (WxoD) [BoHa99] is a system for Web- based meteorological products. Here, distributed visualization is used to deliver a variety of meteorological information to many recipients with various client platforms across the Internet. This requires a range of different solutions utilizing HTML, VRML, CGI, and Java technology. The system provides several variants to distribute the visualization process. Therewith, a range of distribution schemes for different tasks are available, e.g., low or medium bandwidth connection, computation on server or client. On the client side, depending on the requirements, HTML/JavaScript- and Java-Front-Ends are used for controlling and presentation. By using Java-Applets the necessary work according to the distribution of the visualization pipeline can be done by the client. On the server side, CGI scripts control the visualization system. Like the Visualization Web Server, this a client/-server application without the possibilities for load balancing, which restricts the scalability of such a system. WxoD is a collection of separate solutions for special applications and tasks, without a common framework. Thus, the consideration of all requirements concerning the visualization of economic data is not feasible.

3 USER-FRIENDLY ACCESS TO INFORMATION

The contemporary situation in the age of information can be described as a huge data universe containing information that the user wants to access. To describe this situation, the *WWWW model* has been developed [Lux00].

The WWWW Model

On the one side, you have suppliers providing information, on the other side, there are users demanding information. To characterize this situation we have developed the WWWW model (**w**here, ho**w**, **w**ho, **w**hat for). This model includes all questions concerning information (preparing, obtaining, understanding it).

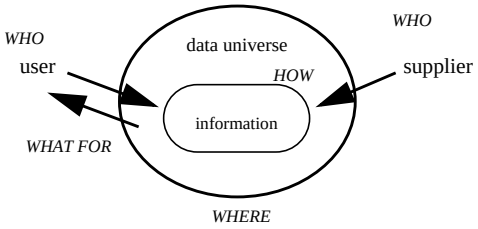

Figure 1: The WWWW model

The first question will be *where* to find data and information. In order to gain information, the supplier needs to know *how* data is structured. The preparation of information depends also on the user *who* is looking for it and on the purpose he/she is having in mind (*what* for).

In this situation, in order to establish a user-friendly access to information, access per se has to be analyzed. For the supplier as well as for the user, this access can be seen from two different points of view: on the one hand, the technical aspect and on the other hand, the aspect regarding the content. In view of the WWWW model, the technical side is influenced by the where and who, the content side by the how, who and what for.

Technical aspects

The assumptions made above can be transferred to a simple model used in economics. On a particular market (the data universe), there is supply (by suppliers) and demand (by users) for certain goods (information). Following this analogy, mechanisms for coordinating supply and demand in the information market will have to be developed, i.e., methods and procedures for an easy provision and distribution of information will have to be explored. In doing so, the following aspects have to be considered in particular.

Use of resources that are already available

Suppliers of information will only accept the market model mentioned above if they also get user-friendly access. In other words, in this model it has to be possible to provide information fast and easily. It is indispensable that all resources already available for obtaining information can easily be integrated. However, it has to be considered that not every information is provided free of charge.

Minimal hardware requirements

From the user's point of view, the available power of his/her own technical equipment should not be relevant when searching for information. If, for example, information is provided by knowledge tools or data mining tools, this software should absolutely be platform-independent. In the case of visual representation of information, local or distributed visualization must be considered depending on the user's hardware configuration.

Utilization of information obtained

The spreading of gained insights can be regarded as the final aspect of the technical user-friendliness. Assuming that a specialist has processed the information necessary for a particular objective and, while doing so, has generated new knowledge, he/she wants to make it available within this model. This means that we have to consider the case that a user becomes a supplier of information.

Technical access can be summarized in the following chart (see fig. 2).

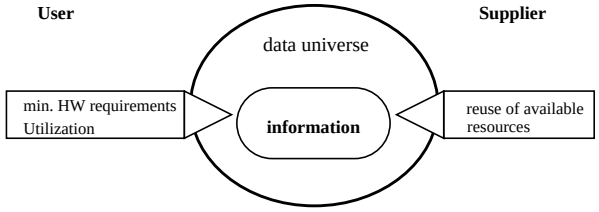

Figure 2: Technical user-friendly access to information

In order to realize this technical access, a service model satisfying all these requirements has been developed and implemented in a framework architecture.

Access to information contained in data

This aspect is more complex. In order to arrive at a user-friendly access to information, the problem has first to be regarded from the supplier's point of view. First of all, the question arises how information can be obtained in the data universe. As soon as the supplier has generated the information from the data universe, the question is how to present it to the user. This can be in the form of a simple text or visual representations. After all, the user receives the information in that format. This leads to the question, what kind of representation of information the user needs for his/her own use or for further spreading. The access described above can be summarized in the following chart (see fig. 3).

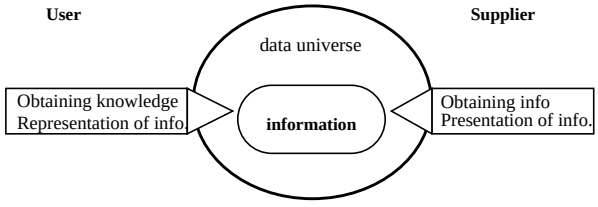

Figure 3: User-friendly access to the information contained in data

Obtaining knowledge

A more detailed presentation of the question of discovering knowledge related to the problems concerning abstract data is offered in [CaShEi99]. The *knowledge crystallization task* Card developed describes the process of obtaining knowledge with a predefined goal. The course of obtaining knowledge that is described there makes it clear, that the way to the goal plays an important role and can be significantly assisted by the information supplier. But even if the exploration of data is not aimed at a particular goal, the user should be aided in obtaining knowledge. The formation of knowledge has to be viewed in the context of user-friendly access.

Looking at the entire interaction between supplier (obtaining and presenting information) and user (obtaining and presenting knowledge) shows the following temporal sequence (see fig. 4).

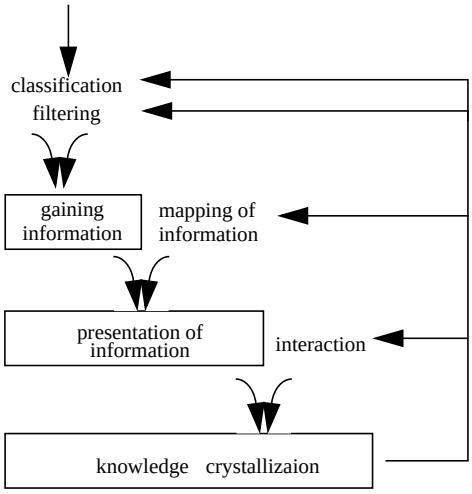

Figure 4: Knowledge crystallization for abstract data

Classifying abstract data is a basic prerequisite. The data-inherent structures and contexts have to be known, otherwise it is not possible to get useful visualization that assists the user in his/her analysis. Then the information obtained can be mapped in visual structures in view of the user, his/her knowledge and goals of analysis. Only through interaction with the information presented, the user can arrive at knowledge crystallization.

Obtaining information

At first data has to be analyzed and classified in order to obtain information from it. A detailed discusson can be found in [Lux98]. The relevant data for obtaining information has to be filtered out of the huge amount of data. In this process pieces of information from different data sources have to be combined. In preparing data, common data bases have to be found in order to represent various data at the same time. Thus, any values derived from data can either be regarded in relationships or be compared. It has to be taken into consideration that the data may be irregular in time and space.

The preprocessing of data will possibly have to be executed several times, for example, if new data is loaded into the system. This can be the case when the user starts off with a particular aim and, in the course of his analysis, finds out that he/she needs additional information, or changes the aim of analysis altogether. Thus, data preprocessing has to be very flexible and to support user require-

ments.

Presentation of information

Now, we have to discuss the way of graphically presenting the information. The data is mapped to visual structures that augment a spatial substrate with marks and graphical properties to encode information. Visual representation has to be found for the diverse user community, so that they are easily and intuitively comprehensible and can provide decision finding. Information visualization is clearly dependent upon the properties of human perception. In order to make full and correct use of everything that display, graphics and visualization technologies have to offer us, we have to take these aspects into account. In the following, we will shortly discuss the different kinds of visual structures, especially visual metaphors.

Various analysis tasks require different visual metaphors and visualization techniques. In some cases, it can be necessary to use more than one technique for the same data representation, because this offers different views on the data. A good overview of existing visualization techniques can be found in [CaShEi99]. A visual metaphor describes an idea of a graphical representation in such a way that the visual representation contains an association to the data. A visual metaphor is a symbolical and analogous description of contents, and not a representation of the reality. However, often realistic elements are used. The wide range of visual metaphors varies from simple graphical primitives to complex scenes.

We have introduced a detailed analysis of the information contained in data. In the following, we will be concentrating on the technical aspects.

4 THE SERVICE MODEL

It was defined in the requirements that the user can request data from several sources, use various output media, and manipulate or process them with the assistance of different applications. As these resources may be distributed over a number of machines, the client has to know where a server is to be found or how it can be accessed. This approach has a few disadvantages:

Should the server have to be migrated to another machine, e.g., for performance reasons, then all potential clients would have to be informed (if this was possible at all). Often there are duplicates or counterparts of a server in order to enhance availability. If the server fails, the client can switch to the duplicate, but he/she has to know where it is and how it can be accessed in order to switch over explicitly. Should the server fail altogether or should a new server be added, all clients have to be informed of that. In order to avoid these difficulties, another level, acting as a mediator, has to be inserted between client and server: *the broker*.

To be able to integrate all resources into the system and to avoid the discussed problems we have developed the *service model* with which applications, data sources, etc. can generally be described as abstract service.

Integration of arbitrary resources

In this model, one or several services are offered by a service provider. The service provider does not offer them directly but via a broker. Just as an agent represents artists, a broker arranges services for a client while the client is not aware of the location and the number of available instances of the service providers. The client requests a service from the broker and receives it. This way, the client only has to know the broker's "address", and a service provider's migration becomes transparent to him/her. If several instances of a service provider exist, the broker can -in case of the service provider’s failure- switch to a duplicate without the client becoming aware of this. Furthermore, the broker is in a position to spread the load out to various instances. During operation, new services can be added or existing ones withdrawn. Service providers register with the broker under the name of the service they are offering. Then, the broker is able to connect the server to interested clients (see fig. 5). It has to be noted that as soon as a client has chosen a service, all communication takes place between client and server only.

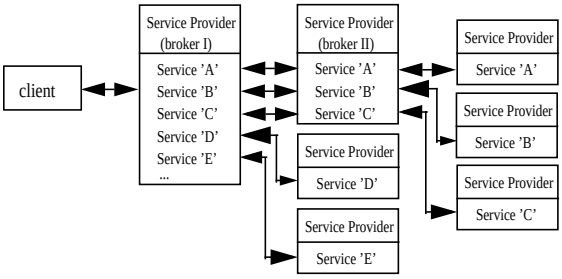

Figure 5: The service model

As explained above, the system allows for services to be added or withdrawn during operation. Such changes in services offered can be of interest not only for many clients, but also for service providers, as they in turn can make use of services. Continually requesting the list of currently available services is neither useful nor efficient, so an event system has been integrated which informs interested parties of added or withdrawn services.

By the type of data, the broker is also a service provider. This has the advantage that a broker can offer "his/her services" to other brokers as well. This way, several brokers can be cascaded, which again is completely transparent to the client (see fig. 5). Thus, hierarchies of brokers can be established which may correspond to existing hierarchies. Within a large

company, for example, each department could have its own broker. These department brokers could offer their services wholly or partly to the subsidiary broker, so that this selection of services is available to all departments. The services offered by the various subsidiary brokers is then represented by a central company broker.

The service's interface has deliberately been kept very simple in order not to restrict its functionality right at the start. By definition, a service has only one method to start a dialogue with the user. What this dialogue looks like or of what kind it is, is not predefined. However, a service should additionally offer an interface allowing the programmable use of a service.

Distributed visualization

A further important issue besides the integration of any given resource is the possibility of an arbitrary distribution of the visualization pipeline. The exclusive visualization on the server makes a lot of things easier, but it also severely restricts the possibilities of interaction. If visualization is completely transferred to the client, this allows for a high degree of interaction, but places high demands on the client's system as well. In order to meet all these requirements, the system supports the arbitrary distribution of the visualization pipeline. To make this possible, a service is realized by using smart proxies (see fig. 6). A smart proxy is defined as a local object running on the client side that encapsulates the access to the remote object on the server [Sun98]. The smart proxy can delegate a user's request to the remote object or process it itself, i.e., locally. This allows, for example, the caching of requests. This means, that the service is split into a remote and a smart proxy part, which makes it possible to realize an arbitrary distribution of the visualization pipeline.

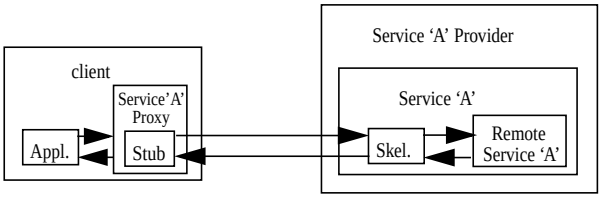

Figure 6: A service with a remote and a smart proxy

5 ARCHITECTURE

The described service model was realized by developing the system PROSECO (PROviding Services for ECOnomic data). In the following, we will introduce the architecture.

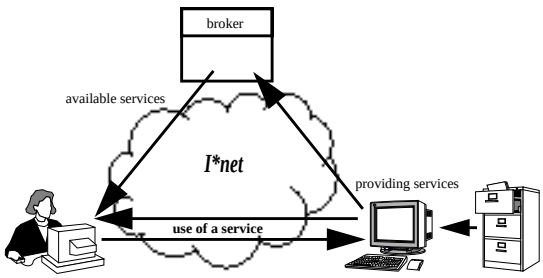

Figure 7: Architecture

PROSECO is designed as a kind of net which can be adapted to an existing infrastructure of hardware and software. Any resources, e.g. knowledge, documents, pictures or printer can be described as a service in an abstract way. These services are offered to one or more brokers. Every client can choose any service from a broker, independent of their physical location in the Internet and Intranet (I*net).

Platform independence

For a distributed system, the Internet would be the obvious choice as global infrastructure. However, it is not possible to predefine the architecture on the client, which means that the system has to be realized independently of the platform. Java [Lea97] would offer itself here, which at the beginning was often smiled at as only fooling about, but which has since developed into an established platform that has to be taken seriously. With the assistance of Java Native Interface, even applications developed in other languages (e.g. C/C++) can be integrated in Java, so that already existing systems can be reused as service with little effort.

Fail safety

In a distributed system, as opposed to a centralist one, separate components can fail. In such a case, however, the system should at least to some extent remain functional or keep its consistency. This is safeguarded by a time-based leasing. The client asks a server for a particular resource. The server allows the client the use of the resource for a certain amount of time. Should the client request the resource for more than the time agreed upon, he has to explicitly ask the server for an extension. If he fails to do so or is prevented from doing so because of network problems or computer failure, the leasing contract expires. As from this moment, the client no longer has access to the resource, i.e., it is again at the server's disposal. This is controlled by a leasing contract of which both parties have a copy. These copies run independently from each other on the client and on the server, so that failure of one does not affect the functionality of the other.

The following figure depicts the successful extension of a leasing contract (see fig. 8a) as well as the turning down of a request for renewal (see fig. 8b).

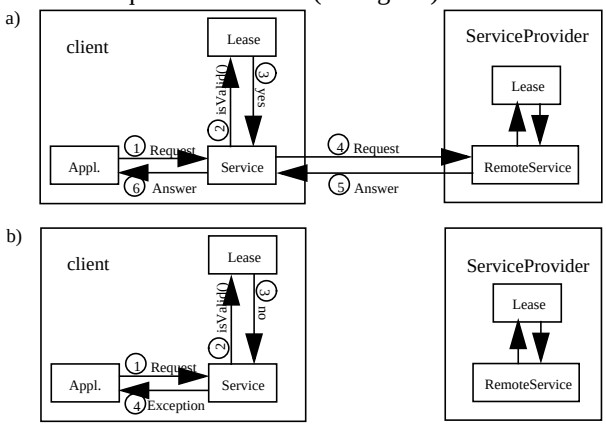

Figure 8: Process of a leasing request

Additionally, an event system has been integrated inform-

ing about any changes in a leasing contract, so that the extension of a contract can be automated, for example. This has been realized by using a class that takes over the control of changes in leasing contracts (see fig. 9). Within this class, a thread runs that functions as a timer [Oaks97]. In order for this to work efficiently, the thread is sleeping for the period up to the next change in the leasing contract. This means, it only uses CPU time for the creation and distribution of the event. Since the timer has to control any number of contracts simultaneously, these are sorted in order of urgency, i.e., the time up to the next change.

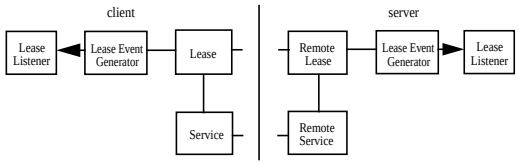

Figure 9: Leasing event system

Security

In a distributed system, security is an important factor. Sensitive data has to be protected against improper access. This system has been supplied with a possibility for general access restriction based on the concept of accounts that is well-known from many operating systems. The user logs in with an ID and a password -transparent to him/her- and an account is assigned to the user. This account contains his/her access rights which can be checked by the service providers. Thus, access can be restricted in as much detail as desired.

Most certainly there are situations in which a user's preferences need to be changeable during running time, e.g. in order to add new rights. Should a service request these rights only a single time, say at the start of a dialogue with the user, then these changes might not be carried through. So, this information needs to be kept centrally and requested continually. Permanent access, however, would put (unnecessarily) heavy pressure on the network. A solution to this problem, again, would be presented by a split into a remote object and a smart proxy (see fig. 10).

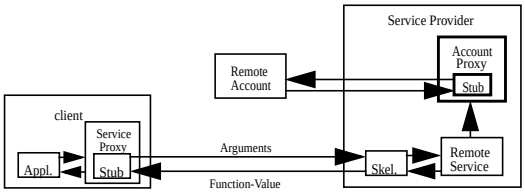

Figure 10: Account

The remote account would be kept on a central server, while the services would be provided with the smart proxy enabling them to gain access. The proxies can cache requests, so that the remote object does not have to be accessed every time. To propagate the changes, the contents of the cache will have to be declared invalid regularly. This presents a good compromise and is completely transparent.

Terminals

After the login, the client does not receive his account but a *terminal* that has internal access to the account. The terminal is a level that encapsulates direct access to the broker. There are several reasons for this middle level. On the one hand, it facilitates the introduction of a leasing contract regulating the use of the broker's resources. On the other hand it serves to hide the account from the user, or to make it transparent. Since the client has no direct access to the account, he cannot manipulate it. The terminal has basically the same interface as the broker, but without the account as parameter. The terminal propagates the user's requests to the broker and adds the account to the parameters. As the user can allocate the broker's resources, measures have to be taken for these resources to be freed again in case of failure. If the terminal is split into a remote and a smart proxy part and furnished with a leasing contract, the remote part can release the resources still reserved upon expiry of the contract (see fig. 11). Thus, the system remains in a consistent state.

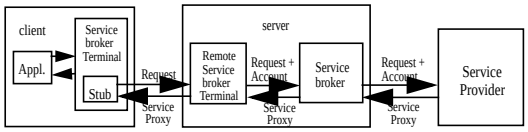

Figure 11: Terminal

The accounts are a means to save user preferences and, thus, to realize sessions tailor-made for the user's requirements and needs. Furthermore, the billing of the services can be integrated with them.

Load balancing

To avoid a central broker as exclusive resource and, thus, as a bottleneck, a service provider can put his services at the disposal of several brokers. Using a number of brokers offering the same services means, that the failure of one of them does not lead to total failure of the system. But it also entails that the broker himself cannot keep a record of the load balancing to the individual service providers, as he does not know to what extent other brokers use them. For this reason, service providers remember the use of their capacity and inform all relevant brokers when their charge changes by an adjustable limit. Since a broker is also a service provider, he offers the same mechanisms for "his" services. If there are several instances of the same service, he forms the mean of their load and chooses the one least used.

6 RESULTS

At the beginning, we explained that economic data is to be seen as a representative example of the prob-

lems concerning user-friendly access to information. Considering an economist's workflow, the demands to the WWWW model can be shown. Therefore, we will now take a short look at an economist's workflow (see fig. 12).

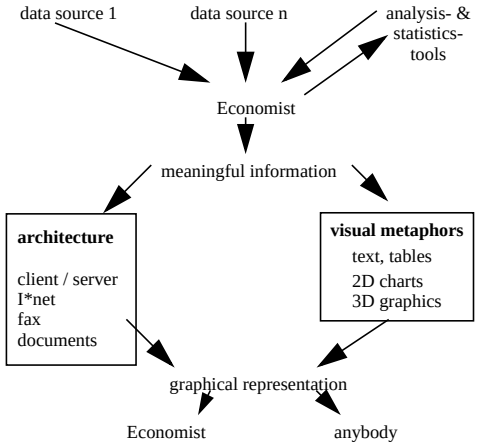

Figure 12: Workflow of an economist

Typically an economist gathers data from different sources according to his/her analysis aim. During the exploration of the data, he/she uses analysis and statistics tools to receive a set of data that is 'meaningful' information from his/her point of view. The next step will be to propagate the gained results to other people. Now, two questions are appearing: to whom and how to present the results. On the one hand, an architecture is necessary to distribute the results to clients via I*net, to send it to a fax or printer, etc. On the other hand, the graphical coding of the information has to been chosen, i.e., a visual metaphor.

The system we have developed can be seen as a framework in which all these requirements can be easily fulfilled. We will demonstrate the integration of services into this architecture by using the system ShareVis [Lux98]. ShareVis is a system for the interactive visualization and analysis of share prices and stock indices. Its purpose is to visualize a large number of share prices or data derived from the stock market, like share indices, in a highly interactive manner. ShareVis enables the viewer to estimate the situation of stock exchange activities in a global context. Additionally, the tool supports the user in detecting connections and interdependencies among shares. Using innovative 3D techniques, the system helps users to analyze the data more efficiently. With the integrated virtual trackball, the navigation in the 3D world is possible and the three-dimensional data is perceptible. The user can optimally customize his/her view on the data. He/she can choose interactively the visual metaphor used to represent the information. ShareVis provides data imported from different on-line databases, so that share prices as well as additional information, such as business reports or news, can be linked to the system.

The following example shows the use of ShareVis as a service. At first a client registers with a broker, to receive events, so that he is informed of services added or withdrawn. As a next step, a service is registered with the broker who informs the client of this event. The client now displays the available service (see fig. 13).

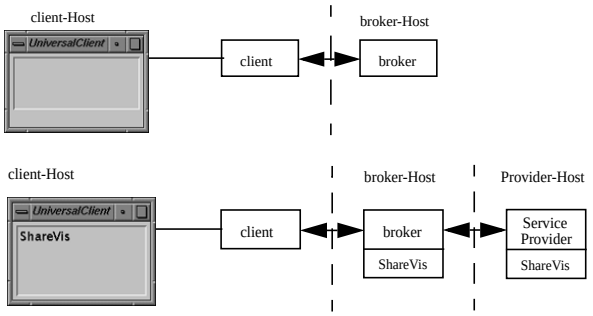

Figure 13: Available service

With a click, the client requests a service from the broker and the service's user dialogue is started. In this case, the dialogue consists of three windows that make the service's functionality available. The user can choose the date, i.e., the trading day, and the type of visualization. To demonstrate the distribution of the visualization pipeline, this service offers the possibility to render on the server or on the client. First, we will discuss rendering on the server. The client part of the service, i.e., the smart proxy, forwards the parameters adjusted by the user to the server, requests a rendered image, and displays this image locally (see fig. 14).

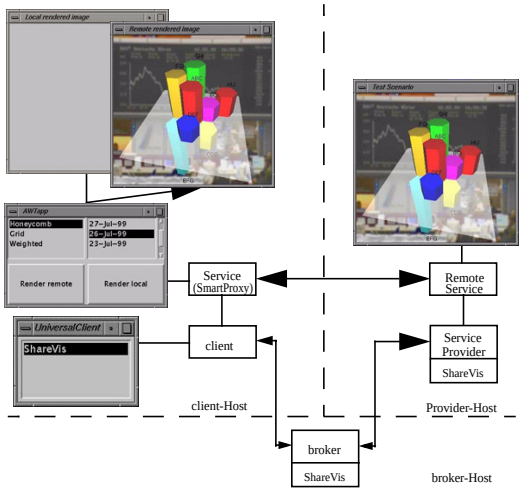

Figure 14: Remote rendering

This being a static image, the user has no possibility of interaction. It has to be noted that communication now only happens between smart proxy and server, i.e., the broker is no longer needed.

The next example deals with rendering on the client (see fig. 15). To begin with, the rendering system has to be started on the client. The resources needed for this will only be requested when they are actually needed. This has the advantage of offering the client several possibilities which he/she can use depending on his/her hard- and software. Once the rendering system has been started successfully, the smart proxy again sends the parameter to the server, but now requests a description of the scene that is rendered locally rather than a static image. The user is now in a position to manipulate camera position and orientation and, thus, gain more insight into the data by per-

ceiving the three-dimensional space.

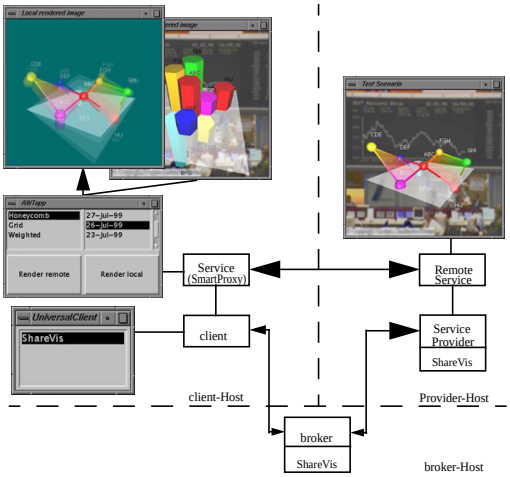

Figure 15: Local rendering

7 CONCLUSION

This work defines an open, scalable framework to visualize economic data in a distributed environment. The concept itself meets all demands on simple technical access and offers concrete solutions to the varied problems. To achieve the required platform independence, the example implementation covering financial data sets is based on Java. Java's RMI interface was chosen to distribute the objects forming the visualization pipeline. By defining a general model of a service, it is possible to distribute application or data sources in an abstract way. This level of abstraction facilitates their uniform integration into new systems. The services are provided by a broker rather than directly. This middle level is used to implement load balancing strategies or to move the actual location of the service in a transparent way. Security within the proposed framework is handled at an access restriction level. To minimize the time if services are unavailable because the client using them is failed, a time-based leasing scheduler was integrated. This scheduler assures that resources are freed after a fixed amount of time. Overall, the design developed in this work offers an open and flexible framework for the distributed visualization of economic data. Available resources can be integrated at any time to enhance the performance of the whole system. We have implemented an example for a service where an existing application has been integrated into the system. Using this example, we have been able to demonstrate the fulfillment of all requirements, in particular the distribution of the visualization pipeline.

Future work will be to integrate some methods for the security. This concerns the coding of the password and account files on the one hand, and the general coding of communication with SSL on the other hand. Java, however, offers cryptography APIs that makes the coding of files relatively easy, since it provides the necessary mechanisms on a high level of abstraction.

Furthermore, this system should be used to integrate all possible applications into the system as a service. This does not only mean new applications, since existing systems can also be adapted to the service model with little effort. Using appropriate drivers, even hardware can be made available as a service, so that a complex distributed infrastructure can be built.

ACKNOWLEDGMENTS

Special thanks go to A. Wilde and M. Unbescheiden for stimulating discussion and presentation suggestions. Additionally, we would like to thank B. Mueller and S. Wurster for proof-reading this paper.

REFERENCES

[BoHa99] Bock, M.; Haase, H.; Herber-Pflüger, A.; Koppert, H.J.: Weather on Demand, Individual Interactive Weather Visualization in the World Wide Web, *ECAM'99*, Sweden, 1999.

[CaShEi99] Card, S.K.; Shneiderman, B.: Eick, S.: Readings in Information Visualization - Using Vision to Think, Morgan Kaufmann Publishers, Inc., 1999.

[Lea97] Lea, D.: Concurrent Programming in Java, O'Reilly & Associates, Inc., 1997.

[LePi99] Lefer, W.; Pierson, J.-M.: A Thin Client Architecture for Data Visualization on the World Wide Web, *Proceedings of the International Conference on Visual Computing (ICVC'99)*, India, 1999.

[Lux97] Lux, M.: Visualization of financial information, *Proceedings of the Workshop on New Paradigms on Information Visualization and Manipulation (NPIVM'97)*, 1997.

[Lux98] Lux, M.: Level of data - a concept for knowledge discovery in information spaces, *Proceedings of the International Conference on Information Visualisation*, London, 1998.

[Lux00] Lux, M.: A framework for user-friendly access to economic information, *submitted to VisSym'00.*

[Oaks97] ,Oaks, S.; Wong, H.: Java Threads, O'Reilly & Associates, Inc., 1997.

Sun98] Sun Microsystems: Java Remote Method Invocation Specification, Revision 1.50, JDK1.2, 1998.

[WoBroWr96] Wood, J.; Brodlie, K.; Wright, H.: Visualization over the WWW and its application to environmental data, *Proceedings of Visualization '96*, 1996.