PERFORMANCE ISSUES IN DESIGN AND IMPLEMENTATION OF GISterm

Zsolt VESZELKA, Joachim WIESEL
University of Karlsruhe, Germany
Institute of Photogrammetry and Remote Sensing
veszelka@ipf.uni-karlsruhe.de
wiesel@ipf.uni-karlsruhe.de

KEY WORDS: Interoperability, Spatial databases, Networks, Performance, Query

ABSTRACT

Geoinformation systems (GIS) and their underlying spatial databases are often distributed on several isolated computer systems. The main reasons for this distribution can be identified as differences among the requirements of specialists recording or managing the data as well as the spatial distribution of the data management process (e.g. among several administrative districts or authorities). Joining isolated systems enables a higher integration of geographic data (geodata) and in this way, makes building of new information possible. However, this requires interoperability between these systems. Interoperability and networking (intranet, Internet) are considered nowdays often as the two most important properties of a modern GIS. These aspects are intensively researched and they represent the objects of standardizing activities. Performance plays only a small if any role in these efforts. Taking account of this important aspect is mostly up to the implementation. A GIS user wants to work in a distributed environment using the same functionality as he/she used to do with an isolated desktop GIS. In spite of new problems arising out of such distributed system architectures, response times must be kept within limits.
GISterm, a virtual GIS, is designed to be capable of online accessing different geodata sources within a network environment and to give a unified view of data from these geodata sources to a user. Its architecture not only allows for great flexibility in building applications based on it and installing this in different environments, but also has a strong impact on system performance. The paper presented here describes some of the design aspects and experiences made during the implementation of this distributed system.

1 INTRODUCING GISterm

GISterm developed at the Institute of Photogrammetry and Remote Sensing of the University of Karlsruhe, was born out of a research and development project of the Ministry of Environment and Transport in the state of Baden-Württemberg, Germany. The environmental information system of the ministry was composed of many GIS components and other databases containing data with spatial content. Integration of these components and handling of the various data formats was a huge challenge. Hiding the heterogeneous environment and providing a uniform interface to all that spatial data sources became the main task of the newly developed framework named GIStermFramework (simply: GISterm). The framework aims to integrate different geodata sources into a network-based (intranet, Internet) distributed information system. Furthermore, it provides a generic class and component library allowing for reusability in various other projects (Hofmann et al., 1999).

2 SYSTEM ARCHITECTURE

2.1 Architecture components

The concept of GISterm is based on a three-tier client-server architecture (Figure 1). The middleware component (GIStermServer) acts as an application server playing a server role against one or more clients (GISterm). At the same time, this acts as a client for the underlying geodata servers. This application server implements an abstraction layer where geodata is transformed into a uniform representation. Geodata servers are accessed through adapters which can be viewed as drivers to the underlying GIS. The adapters enable a transparent access to different data sources while hiding system specific interfaces and behaviour (Rolker et al., 1997). Currently, adapters are available to MapInfo SpatialWare and Smallworld GIS. A special form of adapters does not interface any real GIS, but implements very simple GIS functionality on the basis of some data storage mechanisms (set of files or database). Such adapters are the ESRI-shapefile server and the JDBCRIPSAdapter (the latter manages geoobjects with vector and raster geometries in a relational database). Client components can be dynamically distributed over the Internet or an intranet onto any computer in the network. A client consists of a graphical user interface (GUI) and system layers realizing some data management functionality.

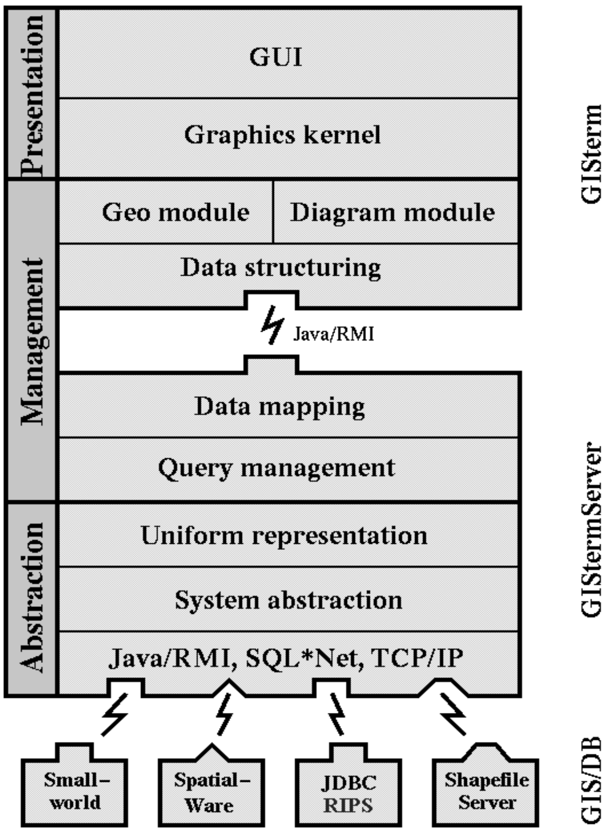

Figure 1. System architecture of the framework

2.2 Applet or application

The entire system is implemented in Java, an object-oriented programming language and platform designed to be used in networks of heterogeneous computers. Java programs can be designed for stand-alone use (Java application) or for operating in the Web (Java applet). In the latter case the byte code of the applet is loaded from a web server into a Java-enabled browser. The applet is then executed inside the Java virtual machine of that browser.
GISterm can be configured to run either as an application or as an applet downloaded from a web server in an intranet environment or over the Internet (Hofmann et al., 1998). The main difference between the two configurations is that client and server components run in the same process space in case of an application, the applet configuration on the other hand, distributes client and server in a network so they must communicate over this network.

2.3 Communication among architecture components

The network interface between client and application server component is realized with the Java middleware technology, RMI (remote method invocation). RMI enables a communication on a high abstraction level through remote objects instantiated in different virtual machines (on different computers in a network). Adapters access data sources either locally or through a network protocol supported by the particular data server (TCP/IP, SQL*Net, JDBC, Java/RMI etc.). If necessary, adapters can be separated from the application server component and communicate through RMI with it. The client component is able to run even without the other architecture components. In this configuration, it draws its data from the web server via HTTP or directly from the local file system.

3 GEODATA REPRESENTATION

Geoinformation systems deal with data about spatial objects. Data originating from various geodata sources connected by GISterm is transformed into (Java) objects called features. Features own spatial and alphanumeric attributes. The representation of spatial attributes (geometries) relies on the OpenGIS Simple Features Specification (Open GIS Consortium, 1999). As real world objects can be abstracted differently depending on the purpose of the abstraction, features can have several geometries each corresponding to a different representation on a given abstraction level (e.g. different geometries for different map scales). A feature can own geometries of both vector and raster type, they are handled in the same way as far as possible. Features are typed objects, i.e. they have a feature type describing their geometries and attribute data. Feature type descriptions are managed in a feature schema, which knows about relationships among types and provides the query processing subsystem with necessary metadata.

4 PERFORMANCE ASPECTS AT DESIGN

4.1 Platform independence

GISterm has been designed to work in a Java environment, i.e. inside a Java virtual machine in order to achieve one of our main goals – platform independence. This enables GISterm to run on a variety of computer platforms but on the other side this platform independence greatly affects performance. As the Java virtual machine must continuously interpret pieces of the platform independent byte code of the program, execution is relatively slow compared to native code. Fortunately, Java just-in-time compilers (JIT) significantly reduce, the new HotSpot™ technology from Sun Microsystems nearly eliminates this performance difference between interpreted and machine code, so it will become a non-issue in the future.

4.2 Network protocol

Network interfaces in the GISterm architecture has been implemented with the Java middleware technology RMI. This high level protocol deals with remote objects distributed in a network. Communication among remote objects takes place through remote method invocations. Remote methods take object parameters and return objects which are serialized before sending and deserialized upon receiving. The costs (in time) of this serialization process stands against the (for us very important) flexibility of working on a higher abstraction level. Communicating through sockets would definitely increase performance, but it would also require the design and implementation of some own protocols which can make development and maintenance more difficult.

4.3 Component clustering

The three-tier client-server architecture of GISterm allows for a great flexibility in system configuration. The client side of GISterm can even be used alone if no online access is needed to geodatabases. In this case, the client is able to access local data (if executed as an application) or remote data over an intranet or Internet connection via a web-server. In case of a need for accessing online databases, the server component (GIStermServer) will be started as well. The network interface between server and client components is implemented with Java/RMI which can become a bottleneck with slow network connections. Therefore, eliminating network interfaces where possible can boost the whole system performance. If configured as an application, GISterm and GIStermServer execute in the same process space (component clustering) communicating locally, i.e. eliminating the relatively slow network interface. Furthermore, if server-side data sources are also locally available, network interfaces between them and the server component can be removed as well.

4.4 Feature caching

Geoobjects (features) are retrieved from their data source only once in a transaction session and are cached in the server component. Another queries resulting features already in the cache will not retrieve these features again; they return merely a reference to them. The performance benefit of caching objects is obvious, and avoiding duplicated representations of features in memory ensures consistency. Furthermore, this consistency also allows for altering data in future versions of the framework.

4.5 Querying geoobjects

As already mentioned, vector and raster geometries and attribute data are aggregated by features representing geoobjects in GISterm. Upon querying geoobjects from various data sources, one or more geometries are retrieved and instantiated for each feature depending on query parameters. Attribute data can be retrieved with each feature if necessary, but if further operation will not make use of this data as a whole, overall performance can be increased by querying attributes only on demand.
Spatial queries are formulated deliberately simple in order for every underlying data source can process them. The adapters transform the internal system independent query representation into system specific ones. These range queries are based on a bounding box (query range), which describes the area of interest. The rectangular form of the bounding box directly relates to the rectangular view the user of the system faces while working with the graphical user interface. Queries retrieve only features geometries of which intersect the specified query range.
Processing of queries happens in asynchronous processes. After the user has initiated a query, processing runs in a separate thread and the user can interact with the system not needing to wait for the results. He/she can even create new queries, those ones will also be processed in their own threads (QueryHandler) independently of one another. If the data source is capable of parallel processing of queries, this mechanism can lead to increased throughput. The other advantage of this design is that user activity is not blocked until results are returned. This consideration is also of ergonomic nature.
If processing a query takes a long time, it can make sense to return results as soon as possible, even if data is not yet completely available. Data can be returned first in low resolution (assumed this data is quickly available) and then in full resolution if all data will have been retrieved. However, another approach has been taken with GISterm in order to improve ergonomics with long lasting queries. The result of such queries is retrieved in smaller steps one after the other. This solution reduces throughput, but response times become more tolerable for the user.
It takes a relative long way for a query to go through all architecture levels to the appropriate data source. This process is illustrated by Figure 2.

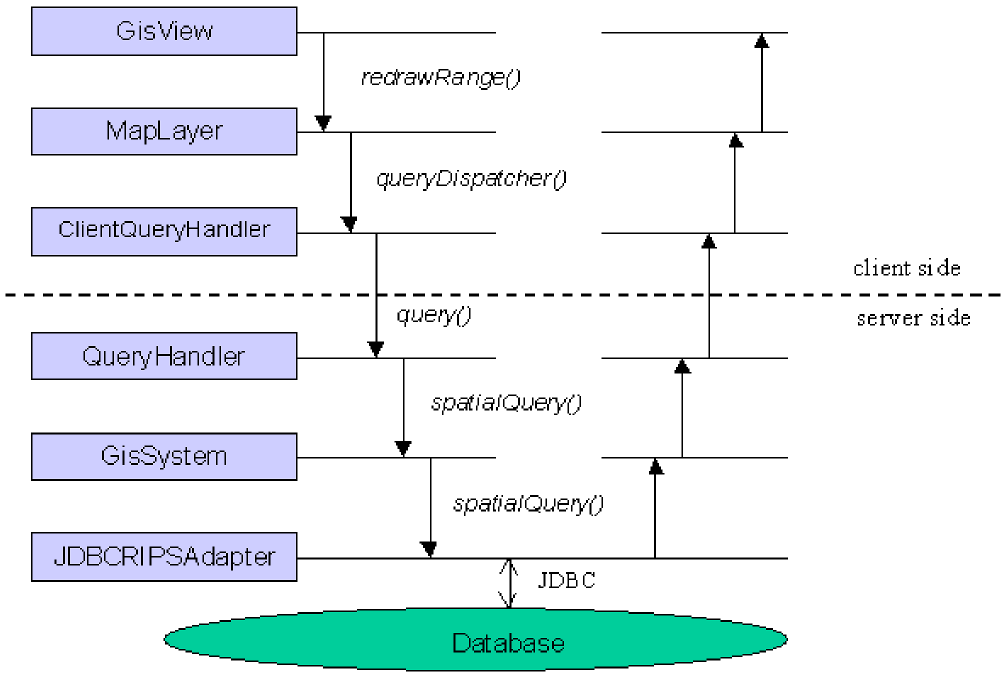

Figure 2. Query processing

4.6 Vector geometries

A feature can own several vector geometries possibly one for each generalization level defined for the feature type. A spatial query can return all or a subset of these geometries. They are always complete geometries, i.e. they are not clipped against the query bounding box. The advantage of this approach is that geometries need not have to be clipped in a time consuming operation and the retrieved original geometry can be used for computing geometric properties. Moreover, the geometry does not need to be retrieved again if the user changes to another view range.

4.6.1 Active layers

Geodata layers (MapLayer) constitute the basic structuring element for map presentation in GISterm. They are containers for semantically related geoobjects. Each layer can either be defined by a single query (static layer with fixed feature set) or they can change their feature sets dynamically through several queries.
The latter layer type is named active layer and is designed for managing geoobjects in a visual manner. The purpose of this management is that only those features which are inside (or overlap) the actually visible map range are managed. If the visible map range changes, it may result in subranges in which no features has been retrieved before. Active layers automatically generate queries for all these ranges retrieving the missing features. The automatic queries do not block user activity because processing is done asynchronously. Result sets of queries are checked against feature sets of the layer initiating the query. Objects which are no more in the visible map range will be removed from the layer, new objects on the other hand, will be given to the layer and displayed. In this way, allocated memory can be effectively minimized on the client side.

4.6.2 Feature mapping

Features are actually stored in the server component. Before they can be visualized on the client side, they are mapped onto corresponding proxy objects each holding a reference to its underlying feature. This mapping creates visual representations of the geometries as well. If the same feature must be displayed in several layers, only the references (proxy objects) should be instantiated on the client, the original feature with its geometries is kept only once on the server. Another advantage of the mapping is that the fineness of geometries can be reduced depending on the graphical capabilities of the client (Hofmann, 1999).

4.7 Raster geometries

Working with geographic data requires special methods as huge quantities of data need to be managed efficiently. This problem becomes even bigger in case of raster geometries where gigabytes of data must be handled interactively without serious performance problems.
The need for managing much raster data with GISterm arose in a project of the state Baden-Württemberg where GISterm must also handle several topographic maps. The huge data quantity (60GB uncompressed) made us design a new subsystem embedded into an adapter with special care. Raster maps had to be stored in a way that accessing them can be realized with the same speed as accessing vector data. Because of the distributed architecture, raster data must be transfered over network in order to be shown on the client side without causing heavy performance problems there.

4.7.1 Raster data in a database

We designed a simple relational database-based raster management subsystem which has several advantages against a file-based solution:

- Vector and raster data can be managed together with attribute data in an integrated system.
- Accessing raster data can be realized through standard network interfaces such as JDBC.
- Geometry and attribute data can be archived in a single process.

4.7.2 Data compression

Uncompressed (raw) raster data often means enormous data quantities and demands of huge storage capacities. Compressing data can greatly help in reducing this problem. (On the negative side compressed raster data must be decompressed before presentation at the latest.)
There are two main categories of compression methods: lossless and lossy compression. Lossless methods (as used in TIFF, GIF, PNG) are suitable for compressing topographic maps because they produce no artifacts at line structures. Lossy methods (e.g. JPEG, Wavelet) can be used for compressing areal or satellite images heavily where quality loss is not so apparent as in topographic maps.
The adapter managing raster data in a database (JDBCRIPSAdapter) keeps raster maps in PNG (Portable Network Graphics) format. This format can achieve compression ratios of 7:1 to 9:1 for our topographic maps. Supporting other formats than PNG was also a design aspect but it has not been implemented yet.

4.7.3 Raster pyramids

An entire raster map is rarely presented through the GUI, only a part of it is usually shown to the user. Moreover, there is no need for detailed maps if the actual presentation scale is small. Such considerations lead to the idea of managing raster maps cut into smaller pieces and stored in different resolutions.
Raster pyramids efficiently realize this sort of management of small tiles (Figure 3). They define several planes for the storage in multiple resolutions. Each plane serves the sheetless management of raster maps in a given resolution and contains many tiles which have been produced from the original big map sheets through splitting them up.

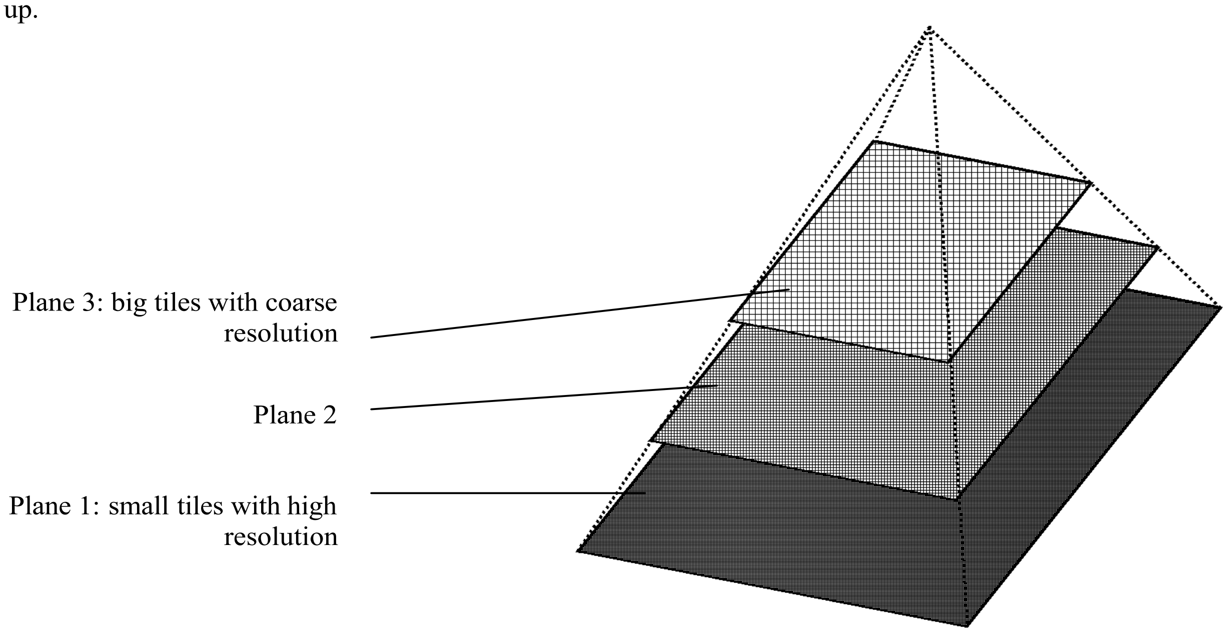
Plane 3: big tiles with coarse resolution
Plane 2
Plane 1: small tiles with high resolution

Figure 3. Raster pyramid

The aforementioned adapter implements a simplified raster pyramid. The simple pyramids and their metadata are completely stored in a relational database (Veszelka et al., 2000).
Tiling big raster maps can be a time consuming process, therefore it is done in a batch job. These tiling processes are usually run only once while filling the database with raster maps. Then raster data is available for queries as static geometry data.

4.7.4 Querying raster geometries

Spatial queries on raster geometries result in only those tiles which intersect the query range and lie on a given plane in the raster pyramid. The plane needed is automatically determined from the pyramid metadata and the actual presentation scale in the client component. The whole query mechanism for raster geometries is very similar to that for vector geometries (Figure 2).

4.7.5 Raster data management

As raster geometries are handled similar to vector geometries, the active layer management also works with features having raster geometries. However, a client-side caching mechanism has been added to the active layer management for raster geometries. Transfer of raster data from data source to client over slow networks can take much time because of the usually big data quantities. The purpose of the implemented "geometric" caching mechanism is to achieve an acceptable performance during typical user activities through minimizing network traffic.
The slightly different behaviour compared to the basic active layer management mechanism is that in case of a change in the visible map range not all tiles are removed from the layer, which fall outside of the new range. Those ones which lie next to the new range will still be cached. This strategy fits well to one of the most often performed user interactions – the panning.

4.8 Operations

Working with a GIS involves the execution of many operations. Operations can be categorized into two main groups: lightweight and heavyweight operations (Hofmann, 1999). Lightweight operations are highly interactive and should give the user immediate feedback. Graphical operations on the GUI of the system belong to this group; examples are zooming, panning the map view, selecting objects, changing visual attributes, displaying basic information about objects etc. Heavyweight operations include the execution of generic or application specific data intensive algorithms. Querying geodata from a database, intensive numeric computations are such operations.
User acceptance of an application is greatly influenced by the response time of the system. In a distributed environment where network capacity and resources of component nodes are limiting factors, system design must carefully weigh where operations should execute and how much data should be moved between nodes. The approach taken in design of GISterm places highly interactive lightweight operation onto the client, data intensive operations on the other hand are executed on the server side to prevent unnecessary data transportation.
GISterm tries to provide a transparent view of the different geodata sources hiding system specific characteristics from the user. Since not only real GI systems but also servers are connected which do not provide GIS functionality, a small set of geooperations (e.g. spatial predicates, buffer) has been implemented in the server component. These operations can uniformly handle all geoobjects in the server but for objects originating from a real GIS they represent only an alternative solution with reduced performance. Recent development of GISterm addresses this problem and will use operations provided by real GI systems which can perform these operations much more efficiently.

5 IMPLEMENTATION AND TUNING

Although carefully designed in an iterative process, straight implementation of the components and algorithms in GISterm did not immediately result in high performance. Further optimizations had to be taken in order to achieve the performance similar to that of a desktop GIS. The following paragraphs list only a few of the tuning points, a detailed discussion would require more knowledge of system internals.

- The use of a just-in-time compiler (or HotSpot™) is essential, the performance of interactive graphical operations as well as common operations on the server (object instantiation, serialization etc.) becomes unacceptable without it.
- The flexible architecture allows for several network connections among architecture components. As communication through network connections is significantly slower than communication among local processes or inside one process, network interfaces should be eliminated where possible. If GISterm should run as an application, client and server component can be executed in one process. The most dramatic performance improvement can be achieved by this component clustering.
- Geographic information systems are designed to manage a huge number of geoobjects. GISterm can connect many such systems rapidly increasing the number of objects to deal with. Therefore, minimizing the number of objects in memory becomes one of the most important optimization goals. A large number of abstraction levels in a system eases design, implementation and maintenance but increases the number of intermediate objects needed to instantiate (Berg, 1998). This number can be greatly reduced by the reduction of the abstraction levels.
- There are several properties of geoobjects that are continually used in GISterm (e.g. bounding boxes of geometries). If these properties are not available directly from the data source, they must be computed. Properties often used are cached internally in order not to have to compute them every time when they are needed. Furthermore, the initial computation is not made until the property is actually accessed.

6 CONCLUSION

This paper introduced GISterm, a framework which can be used in heterogeneous network environments in different configurations. It allows for accessing distributed geodata sources in a transparent manner, i.e. the user of the system sees a unified data model and the same functionality is available for him/her with every piece of geodata independently from the specifics of the source of the data.
The design and implementation of the framework has been an iterative process based on experience and user feedback. In designing the framework, concepts of the OpenGIS standardization effort have played an important role. Some of the concepts, performance aspects and problems which arose at design or implementation time were discussed in the above chapters. The valuable experiences gathered while implementing the distributed

(virtual) geoinformation system, GISterm, can be applied to other Internet/intranet-based GI systems as well. Therefore, the authors hope the paper presented here will provide similar projects in the future with useful information.

REFERENCES

Berg, C. J., 1998: Advanced Java Development for Enterprise Applications. Prentice Hall PTR, 1998, pp. 357-367.

Hofmann, C., 1999: A Multi Tier Framework for Accessing Distributed, Heterogeneous Spatial Data in a Federation Based EIS. In Proceedings of the 7th International Symposium on Advances in Geographic Information Systems (ACM-GIS'99), Kansas City, MO, USA, November 1999.

Hofmann, C., Veszelka, Zs., Wiesel J., 1998: Weiterentwicklung von GISterm zu einer allgemeinen Komponente für den Zugriff und die Visualisierung von räumlichen Daten. In: Mayer-Föll, R.; Jaeschke, A. (editors): Projekt GLOBUS, Multimediales Recherchieren und Verarbeiten von globalen Umweltsachdaten im Umweltinformationssystem Baden-Württemberg, Phase V 1998. Wissenschaftliche Berichte, FZKA 6250, Forschungszentrum Karlsruhe Technik und Umwelt, Karlsruhe.

Hofmann, C., Veszelka, Zs., Wiesel J., 1999: GIStermFramework - Das flexible, komponentenbasierte Geodatenzugriffssystem. In: Mayer-Föll, R.; Jaeschke, A. (editors): Projekt GLOBUS - Von Komponenten zu vernetzten Systemen für die Nutzung globaler Umweltsachdaten im Umweltinformationssystem Baden-Württemberg und anderen fachübergreifenden Anwendungen, Phase VI 1999. Wissenschaftliche Berichte, FZKA 6410, Forschungszentrum Karlsruhe Technik und Umwelt, Karlsruhe.

Open GIS Consortium, Inc., 1999: OpenGIS Simple Features Specification for SQL, Revision 1.1. http://www.opengis.org/techno/specs/99-049.pdf

Rolker C., Nikolai R., Kramer R., Schmidt R., Hofmann, C., Veszelka, Zs., Wiesel J., 1997: WWW-Szenario. In: Mayer-Föll, R.; Jaeschke, A. (editors): Projekt GLOBUS, Umsetzung der neuen Systemarchitektur und Entwicklung weiterer Produktionssysteme für globale Umweltsachdaten im Umweltinformationssystem Baden-Württemberg, Phase IV 1997. Wissenschaftliche Berichte, FZKA 6000, Forschungszentrum Karlsruhe Technik und Umwelt, Karlsruhe.

Veszelka, Zs., Hofmann C., Stein W., 2000: Visualisierung und Verwaltung von sehr großen Rasterkarten aus Datenbanksystemen. Hypermedia im Umweltschutz, 3. Workshop, Ulm 2000. Metropolis Verlag, Marburg 2000 (Umwelt-Informatik aktuell; Bd. 24), pp. 212-216.