SPIN! — an Enterprise Architecture for Spatial Data Mining

Michael May and Alexandr Savinov

Fraunhofer Institute for Autonomous Intelligent Systems
Schloss Birlinghoven, Sankt-Augustin, D-53754 Germany
{michael.may, alexandr.savinov}@ais.fraunhofer.de

Abstract. The rapidly expanding market for Spatial Data Mining systems and technologies is driven by pressure from the public sector, environmental agencies and industry to provide innovative solutions to a wide range of different problems. The main objective of the described spatial data mining platform is to provide an open, highly extensible, n-tier system architecture based on Java 2 Platform, Enterprise Edition (J2EE). The data mining functionality is distributed among (i) Java client application for visualization and workspace management, (ii) application server with Enterprise Java Bean (EJB) container for running data mining algorithms and workspace management, and (iii) spatial database for storing data and spatial query execution.

1 Introduction

Data mining is the partially automated search for hidden patterns in typically large and multi-dimensional databases. It draws on results in machine learning, statistics and database theory [7]. Data mining methods have been packaged in data mining platforms, which are software environments providing support for the application of one or more data-mining algorithms. So far Data Mining and Geographic Information Systems (GIS) have existed as two separate technologies, each with its own methods, traditions and approaches to visualization and data analysis. Recently, the task of integrating these two technologies has become highly actual [3,8,9] especially as various public and private sector organizations possessing huge databases with thematic and geographically referenced data began to realize the huge potential of information hidden there.

As a response to this demand a prototype has been developed [1,2] which demonstrates the potential of combining data mining and GIS. This initial prototype encouraged the development of the SPIN! [4,11,12] system the overall objective of which consists in developing a spatial data mining platform by integrating state of the art Geographic Information System (GIS) and data mining functionality in a closely coupled open and extensible system architecture.

This paper describes an open, extensible architecture for spatial data mining, which pays special attention to such features as scalability, security, multi-user access, robustness, platform independence and adherence to standards. It integrates

Geographic Information System for interactive visual data exploration and Data Mining functionality specially adapted for spatial data. The system is built on the Java 2 Enterprise Edition (J2EE) architecture and particularly uses Enterprise Java Bean (EJB) technology for implementing remote object functionality. The flexibility and scalability of the J2EE platform has made it the platform of choice for building different multitiered enterprise applications so using it as a basis for a spatial data mining platform in SPIN! project [4] is a natural extension.

EJB is a server-side component architecture, which cleanly separates the "business logic" (the analysis tools, in our case) from server issues, shielding the method developers from many technicalities involved in client-server programming. This choice allows us to meet the requirements often found in business applications, e.g. security, scalability, platform independence, in a principled manner.

The system is tightly integrated with a relational database and can serve as data access and transformation tool for spatial and non-spatial data. Analysis tools can be integrated either as stand-alone modules or, more tightly, by distributing the analysis functionality between the database and the core algorithm. Particularly, spatial database is used to execute complex spatial queries generated by analysis algorithms. The final system integrates several data mining methods adapted to the analysis of spatial data, e.g., multi-relational subgroup discovery, rule induction and spatial cluster analysis, and combines them with rich interactive functionality for visual data exploration, thus offering an integrated distributed environment for spatial data analysis.

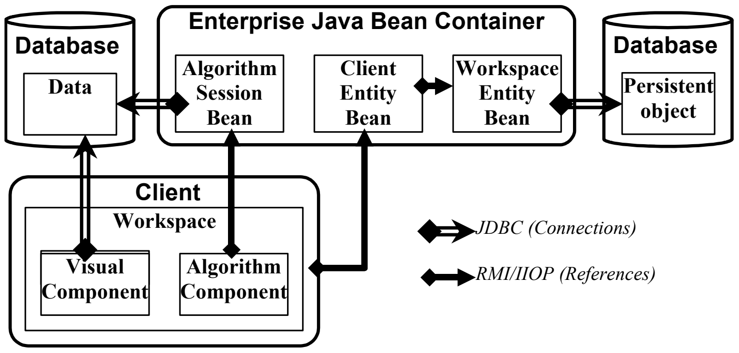


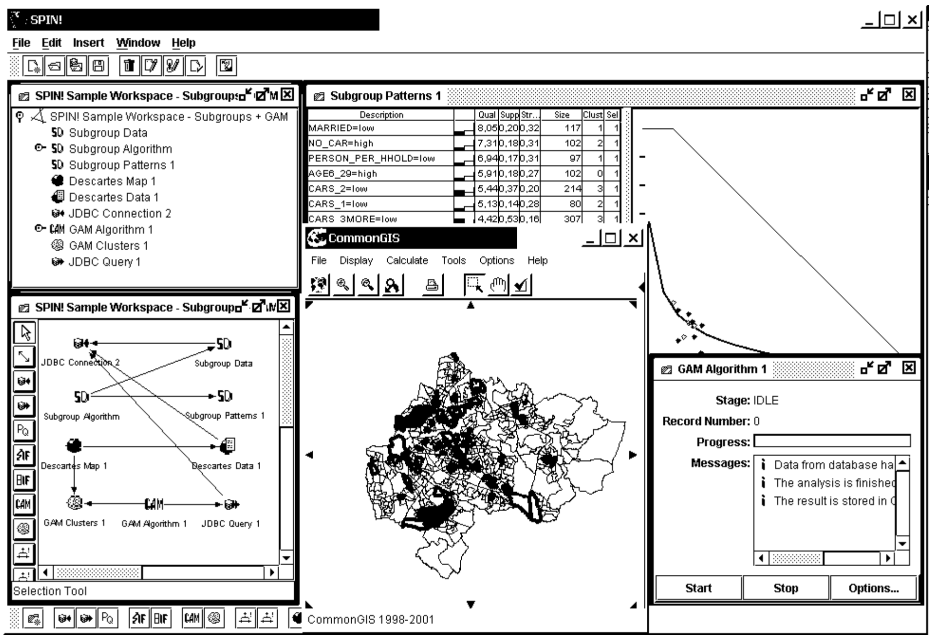
Fig. 1. SPIN! platform architecture. Main components are a Java-based client, an Enterprise Java Beans Container and one or more databases serving spatial and non-spatial data..

2 N-tier EJB-based Architecture

The general SPIN! architecture is shown in Fig. 1. It is a *n*-tier Client/Server-architecture based on *Enterprise Java Beans* for the server side components. A major advantage of using Enterprise Java Beans is that such tasks as controlling and maintaining user access rights, handling multi-user access, pooling of database connections, caching, handling persistency and transaction management are delegated to the *EJB container*. The architecture has the following major subsystems: *client*, *application server* with one or more EJB containers, one or more *database servers* and optionally *compute servers*.

The SPIN! *client* is a standalone Java application. It always creates one server side representative in the form of session bean the methods of which are accessed through the corresponding remote reference via Java RMI or CORBA IIOP protocol. The client session bean executes various server side tasks on behalf of the client. In particular, it may load/save workspace objects from/in its persistent state.

The client is based on component connectivity framework, which is implemented in Java as connectivity library (CoCon). The idea is that the workspace consists of components each of which is considered a storage for a set of parameters and pieces of functionality (e.g., algorithms). The system functionality is determined by a set of available components.



Description		Qual	Supp	Str...	Size	Clust	Sel
MARRIED=low		8,05	0,20	0,32	117	1	1
NO_CAR=high		7,31	0,18	0,31	102	2	1
PERSON_PER_HHOLD=low		6,94	0,17	0,31	97	1	1
AGE6_29=high		5,91	0,18	0,27	102	0	1
CARS_2=low		5,44	0,37	0,20	214	3	1
CARS_1=low		5,13	0,14	0,28	80	2	1
CARS 3MORE=low		4,42	0,53	0,16	307	3	1

Stage: IDLE
Record Number: 0
Progress:
Messages:
Data from database ha
The analysis is finished
The result is stored in C

Fig. 2. SPIN! client. The workspace consists of interconnected components such as database connections, database queries, data mining algorithms, analysis results and spatial object visualizers.

The workspace components and connections among them can be edited in two views: tree view and graph view. In the tree view components from the system repository can be added into the workspace (Fig. 2, left top). User connections among workspace components can be established in the Connection Editor dialog. A more user friendly way of editing workspaces is through a Clementine-style workspace graph view, which shows both components and their user connections (Fig. 2, left bottom). In this view components can be added by selecting them from the system tool bar and connecting them by drawing arrows between graph nodes. It is also very important that components can be arranged within views into visually expressive diagrams.

The *application server* is an Enterprise Java Bean container. It manages the client workspace, analysis tasks, data access and persistency. There may be more than one simultaneously running container on one or more servers so that, e.g., different algorithms and other tasks can be executed on different computers under different restrictions. The SPIN! system uses an EJB container for making workspaces persistent in the database and for remote computations. For the first task the client creates a special session bean, which is responsible on the server side for workspace persistence and access. Particularly, if the client needs to load or save a workspace it delegates this task to this session bean. The client creates one remote object for each analysis task to be run so that data is transferred directly from the database to the algorithm. After the analysis is finished its result is transferred to the client for visualization.

User data are stored in primary *data storage*, which is a relational database system (it may be the same machine as the application server). There may be one or more optional *secondary databases*. In addition, data can be loaded from other sources – databases, ASCII files in the file system or Excel files. It is important that for remote computations in application server data is transferred directly into the remote algorithm bypassing the client. It is only a set of components (subgraph of the workspace) that is transferred between application server and client.

3 Remote Algorithm Management

The developed architecture supposes that all algorithms are executed on compute servers. For each running algorithm a separate session bean is created which implements high-level methods for controlling its behavior, particularly, starting/stopping the execution, getting/setting parameters, setting the data to process, and getting the result. The session bean then is responsible for the methods implementation. There are several ways how it can be done.

- A clean and very convenient but in some cases inefficient approach is using Java for implementing the complete algorithm directly within the corresponding EJB, loading all data via JDBC into the workspace.
- A second approach divides the labor between the EJB container and the relational database. We have implemented a multi-relational spatial subgroup-mining algorithm [6] that does most of the analysis work (especially the spatial analysis) directly in the database. The EJB part retrieves summary statistics, manages hypotheses and controls the search.
- A third approach consists in implementing computationally intensive methods in native code wrapped into shared library by means of Java Native Interface (JNI). A rule induction algorithm based on finding largest empty intervals in data [13,14] has been implemented in this way, namely, as a dynamically linked library the functions of which are called from the algorithm EJB.

- A fourth option is that the algorithm session bean directly calls an external executable module. This approach has been used to run SPADA algorithm [10].
- And finally other remote objects (e.g. CORBA) can be used to execute the task.

The algorithm parameters are formed in the client and transferred to the algorithm EJB as a workspace component before the execution. In particular, data to be processed by the algorithm has to be specified. It is important that only a data *description* is specified and not the complete data set is transferred. In other words, the algorithm EJB gets information where and how to take data and what kind of restrictions to use. Thus when the algorithm starts, the data is directly retrieved by the algorithm EJB rather than passes through the client.

For example, assume that we need to find interesting subgroups in spatially referenced data [6]. The data is characterised by both thematic attributes, e.g., population, and spatial attributes, e.g., proximity to highway or percentage of forests in the area. The data to be analysed is specified in the corresponding component where we can choose tables, columns, join and restriction conditions including spatial operators supported by the underlying database system. The algorithm component is connected to the data component and the subgroup pattern component. The algorithm component creates a remote algorithm object in the EJB container as a session bean and transfers to it all necessary components such as the data description. The remote object (EJB) starts computations while its local counterpart periodically checks its state until the process is finished. During computations the remote object retrieves data, analyses it and stores the result in the result component. Note that each client may start several local and remote analysis algorithms simultaneously and for each of them a separate thread is created. Once interesting subgroups have been discovered and stored in a component they can be visualised in a special view, which provides a list of all subgroups with all parameters as well as a two-dimensional chart where each subgroup is represented by one point according to its coverage and strength. Additionally, the data analysed by subgroup discovery data mining algorithm can be viewed in a geographic information system and analysed by visual analysis methods.

4 Workspace Management

One task, which is very important in distributed environment is workspace management. During one session user loads into the client and works with one workspace from some central storage. As the work is finished the workspace is stored back into its initial or new location. There are several alternatives how persistent workspace can be implemented: (i) the whole workspace is serialized and stored in one object like local/remote file or database record, and (ii) the workspace components and connections are stored separately in different database records. The first approach is much simpler but it is difficult to share workspaces. The second approach allows us to treat workspace components as individual objects even within persistent storage, i.e., the whole workspace graph structure is represented in the storage.

We implemented both approaches and in both cases the workspace is represented as special graph object, i.e., a set of its nodes (workspace components) and a set of its edges (workspace connections). The graphs can be created from existing run-time workspace objects by specifying constraints on its nodes and connections. For example, for loading and storing workspaces view connections are ignored. Then the selected subgraph is passed to the persistence manager. If it needs to be stored as one object then the whole graph is serialized. Otherwise individual node and edge objects are serialized. We used XML for serialization, i.e., any object state is represented as an XML text.

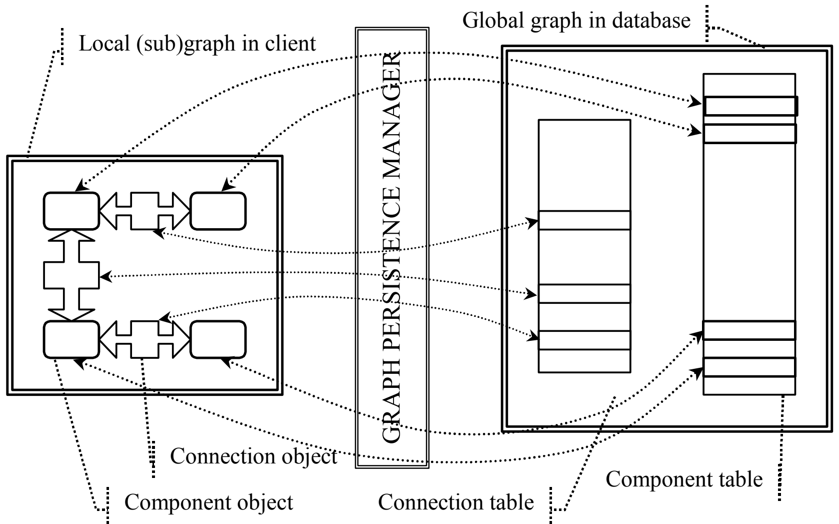


Fig. 3. Workspace is a graph where nodes are components and edges are connections between them. All workspaces are stored in a database and retrieving a workspace means finding its component and connection objects. The persistent workspace management functionality is implemented as a session bean, which manipulates two types of entity beans: workspace components and workspace connections.

The functionality of remote workspace management is implemented by a special session bean. This EJB has functions for loading and storing workspaces. If the workspace is stored as a set of its constituents then the session bean uses entity beans, which correspond to the workspace components. The state of such workspaces is stored in two tables: one for nodes and one for edges. There exist two classes of entity beans, which are used to manipulate these two tables. The workspace management architecture for this case is shown in Fig. 3.

5 Conclusion

We have described the general architecture of the SPIN! spatial data mining platform. It integrates GIS and data mining algorithms that have been adapted to spatial data. The choice of J2EE technology allows us to meet requirements such as security, scalability, platform independence, in a principled manner. The system is tightly integrated with a RDBMS and can serve as data access and transformation tool for spatial and non-spatial data. The client has been implement in Java using Swing for its visual interface. Jboss 3.0 [5] has been used as an application server. Oracle 9i database has been used as a spatial data and workspace storage. In future it would be very interesting to add the following features to this architecture: persistent algorithms running with no client, web interface to data mining algorithms via conventional browser, data mining functionality as web services via XML-based SOAP protocol, shared workspaces where components can belong to more than one workspace.

Acknowledgement: Work on this paper has been partially funded by the European Commission under IST-1999-10536-SPIN!

References

1 Andrienko, N., G. Andrienko, A. Savinov, and D. Wettschereck, "Descartes and Kepler for Spatial Data Mining", ERCIM News, No. 40, January 2000, 44–45.

2. Andrienko, N., Andrienko, G., Savinov, A., Voss, H. and Wettschereck, D., "Exploratory Analysis of Spatial Data Using Interactive Maps and Data Mining", *Cartography and Geographic Information Science* 28(3), July 2001, 151-165.

3 Ester, M., Frommelt, A., Kriegel, H.P, Sander, J., "Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support", in *Data Minining and Knowledge Discovery, an International Journal*, 1999

4 European IST SPIN!-project web site, http://www.ccg.leeds.ac.uk/spin/

5 JBoss Application Server, www.jboss.org.

6 W. Klösgen, May, M. Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database, PKDD 2002, Helsinki, Finland, August 2002, 275-286.

7 Klösgen, W., Zytkow, J. (eds.), Handbook of Data Mining and Knowledge Discovery. Oxford University Press, 2002.

8 Koperski, K., Adhikary, J., Han, J., 1996. Spatial Data Mining, Progress and Challenges, Vancouver, Canada, Technical Report

9 Koperski, K., Han, J. "GeoMiner: A System Prototype for Spatial Mining", Proceedings ACM-SIGMOD, Arizona, 1997

10 Lisi, F.A., Malerba, D., SPADA: A Spatial Association Discovery System. In A. Zanasi, C.A. Brebbia, N.F.F. Ebecken and P. Melli (Eds.), *Data Mining III,* Series: Management Information Systems, Vol. 6, 157-166, WIT Press, 2002.

11 May, M.: Spatial Knowledge Discovery: The SPIN! System. Fullerton, K. (ed.) Proceedings of the 6th EC-GIS Workshop, Lyon, 28-30th June, European Commission, JRC, Ispra.

12 May, M., Savinov, A. An integrated platform for spatial data mining and interactive visual analysis, Data Mining 2002, Third International Conference on Data Mining Methods and Databases for Engineering, Finance and Other Fields, 25-27 September 2002, Bologna, Italy, 51-60.

13 Savinov, A.: Mining Interesting Possibilistic Set-Valued Rules. In: Da Ruan and Etienne E. Kerre (eds.), Fuzzy If-Then Rules in Computational Intelligence: Theory and Applications, Kluwer, 2000, 107-133.

14. Savinov, A.: Mining Spatial Rules by Finding Empty Intervals in Data. Proc. of the 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'03), 3-5 September 2003, University of Oxford, United Kingdom (accepted).