

To address these interoperability issues, we argue that the GeoComputation community needs a standard operating environment that embraces a wide variety of GeoComputation data, models, methods, training and validation techniques and results. Such a standard environment will be specified in the format of Application Programming Interface (API). An API, a term from computer science community, refers to a set of routines, protocols, and tools for building software applications. Currently, there is no widely agreed upon, standard API for GeoComputation. In this paper we propose to specify a pure Java API to facilitate development of highly interoperable GeoComputation-enabled applications. The Java GeoComputation (JGC) API allows java based GeoComputation tools to be engineered to a single uniform interface that can be understood by a wide variety of client application developers and end users. Similarly, java based GeoComputation applications can be coded against a single API in an operating system, platform and vendor independent way. Ideally, the JGC specification is created through a user-driven collaborative process. First, Potential users of the JGC specification in government, private industry, and academia work together to develop a draft specification that is subject to suggestions and input from all users and the general public. Then, the draft specification is revised to accommodate all comments received. Finally, it becomes an official one after being unanimously approved by all users.

Nowadays, data mining technologies are playing an important role in helping geoscientists solve problems involving some form of geographical or environmental data. Thus, the proposed JGC specification should include open and extensible software application programming interfaces for data mining technologies. A new Java Data Mining API (JDM, JSR73) is defined by a cross-disciplinary alliance of developers, sponsored by both Sun and Oracle and now beginning to be taken up by a number of open-source computational projects (Hornick et al., 2004). In this research, we adapted the JDM API to form the data mining part of the proposed JGC specification and implemented much of the adapted JDM API to support development of data mining enabled GeoComputation tools at the GeoVISTA center.

2. Related Work

In recent years, many researchers have studied GeoComputational interoperability issues. So far, most of the previous research focuses on interoperability issues related to geographical information and services that produce or process geographical information. And that kind of research has been conducted from various perspectives.

International and U.S. standards organizations try to solve interoperability problems through the standardization process. The Open Geospatial Consortium, Inc. (OGC), a non-profit international standard organization, has been working with government, private industry and academia to study and develop open and extensible software interface specifications to enable the interchange of geographical information and services. The group of OGC approved specifications, generally called OpenGIS specifications, address a wide range of interoperability issues related to geographical information and services. Besides OGC, other standards organizations such as Federal Geographic Data Committee (FDGC), Global Geospatial Data Infrastructure (GSDI), World Wide Web Consortium (W3C) and Web Services Interoperability Organization (WS-I) etc also develop specifications that help increase interoperability of geographic information and services.

Some researchers argue that specifications developed by standards organizations like OGC only help improve technical interoperability but not the semantic interoperability that is same important to the successful sharing and exchange of geographical information and services. For example, different organizations may interpret the same geographical information or results produced by the same geographical information differently. Harvey et al. (1999) overviews semantic interoperability and through case studies shows the breadth and depth of semantic interoperability issues and approaches to addressing those issues. Schuurman (2002) presents a rule-based approach to standardizing spatial semantics to enable agencies to share and exchange geographical information and services within the same semantic context. Visser et al. (2002) introduce an intelligent broker architecture for semantic-based information retrieval and show how this approach can be used for general purposes.

The Internet and the age of distributed computing lead to an increasing interest in developing and deploying distributed GIS to facilitate the interchange and sharing of geographical information and services. As key application of distributed GIS, geoportals provide a gateway to discover and access geographic content and web services (Tait, 2005). Maguire and Longley (2005) trace the emergence of geoportals, outlining the significance of developments in enterprise GIS and national spatial data infrastructure (SDIs), with particular reference to the US experience. Beaumont et al. (2005) review and interpret the development of geoportals in the United Kingdom, assess the state-of-play in the development of geoportals, and evaluate future prospects. Askew et al. (2005) evaluate the success of the Multi Agency Geographic Information for the Countryside (MAGIC) geoportal in terms of quantitative information and qualitative but equally important aspects, and discuss its future priorities. Bernard et al. (2005) present work associated with establishing a European Spatial Data Infrastructure (ESDI) and requirements for the EU geoportal, discuss about the obstacles related to technological interoperability issues, and make proposals for approaches to speeding up the implementation of ESDI.

Popular open source geospatial software products may also offer a framework for increasing the interoperability of geographical information and services. These open source projects generally have a series of open standards specifying supported data structures and formats as well as application programming interfaces and required coding formats. These standards provide guidelines for participating programmers of these projects to maintain existing modules and add new modules. As the popularity of these products grows, the number of applications conforming to their standards increase and the number of data formats supported by their standards increase as well. Thus, the wide acceptance of these open source software products facilitates the interchange and sharing of geographic information and services. GRASS is perhaps the most popular open source GIS software product that is currently used in academic and commercial settings around the world. It is a Geographic Information System (GIS) that supports geospatial data management and analysis, image processing, graphics/maps production, spatial modelling, and visualization. Geotools is another popular open source Java GIS toolkit for developing standards compliant solutions. It provides implementation of OpenGIS specifications as they are developed. Another open source software product worth mentioning is the GeoVISTA *Studio*. It is a successful example of adopting Java Bean and component-based software engineering techniques to provide a visual programming environment for geoscientific data analysis and visualization (Takatsuka and Gahegan, 2002). It allows users to quickly build applications for GeoComputation and geographic visualization.

In summary, all research surveyed above only deals with interoperability issues of geographical information and services producing or processing it. While computational technologies are also a key component of GeoComputation (Openshaw, 2000). And the successful application of GeoComputation technologies to geospatial problems depends on the successful use of computational technologies. So we argue that GeoComputational interoperability studies need to also consider interoperability issues related to computational technologies in GeoComputation. In this paper, we propose a framework to improve the interoperability of computational technologies in GeoComputation.

3. The Proposed Java GeoComputation API

The JGC specification is proposed to satisfy the immediate need for a standard GeoComputation API to increase the interoperability, reusability and flexibility of GeoComputation applications. It is expected to support common GeoComputation operations, as well as the creation, persistence, access, and maintenance of metadata supporting GeoComputation activities.

3.1 Benefits

The benefits of JGC specification to the GeoComputation community are obvious:

1. Increases the interoperability of GeoComputation tools, facilitates the sharing of GeoComputation tools both within and outside the GeoComputation community, and reduces the cost and potential risk of deploying GeoComputation tools to real world applications.
2. Contributes to the development of scientific standards in GeoComputation and therefore helps reduce some of the reluctance among the quantitative analysis community to adopt GeoComputational tools (Couclelis, 1998).
3. Provides better support for the automation of analysis and modeling functions, such as finding the optimal configuration parameters for a machine learning method, and helps create end-user friendly GeoComputation technology (Openshaw, 2000).
4. Promotes the sharing of GeoComputation technologies, services and information resources in our increasingly distributed and mobile society.

3.2 Target audience

The JGC specification is targeted at the following audiences:

- GeoComputation software vendors, companies and research organizations that intend to implement this API for their GeoComputation software products to provide end users with full access to GeoComputation technologies packaged in this API.
- GeoComputation application developers, Java programmers who wish to use a GeoComputation API for building GUIs or other applications that support solving geospatial problems with GeoComputation technologies.
- GeoComputation experts, individuals with thorough understanding of enabling technologies of GeoComputation, and extensive practical experience applying GeoComputation technologies to real world problem-solving.
- GeoComputation novices, Java-knowledgeable developers who have a basic understanding of the problems that GeoComputation can solve, who have a minimum functional level knowledge of GeoComputation tasks.

3.3 Architectural components

The proposed JGC specification has three logical components that may be implemented either as one monolithic system or in a distributed environment:

- GeoComputation application programming interface (GCAPI), the GeoComputation API gives end users access to services provided by the GeoComputation Engine (GCE). It shields users of GeoComputation applications from details about the actual implementation and supporting components of the GCE.
- GeoComputation engine (GCE), a GCE has a concrete implementation of the GeoComputation services defined in the JGC specification and provides these services to end users through clients conformant to the GeoComputation API. The GCE can be implemented as a server in which case it is called a GeoComputation Server.
- GeoComputation object repository (GCOR), the GCE uses an object repository to make various GeoComputation objects (such as geospatial data, configuration parameters, results) persistent. This repository stores GeoComputation objects so that they can be used by the GCE to support its operations. The GeoComputation object repository may exist as a flat file system or a relational database.

Figure 1 shows three possible architectures for a JGC implementation. In (a) each component is implemented as an independent unit in a three-tier architecture. These components may reside in separate computers at separate physical locations or separate processes or executables in the same computer at the same physical location. In (b) there is the classical client-server architecture where GCE and GCOR are combined together to be at the server side and the GeoComputation API is at the client side. In (c) all three components are embedded in one monolithic system.

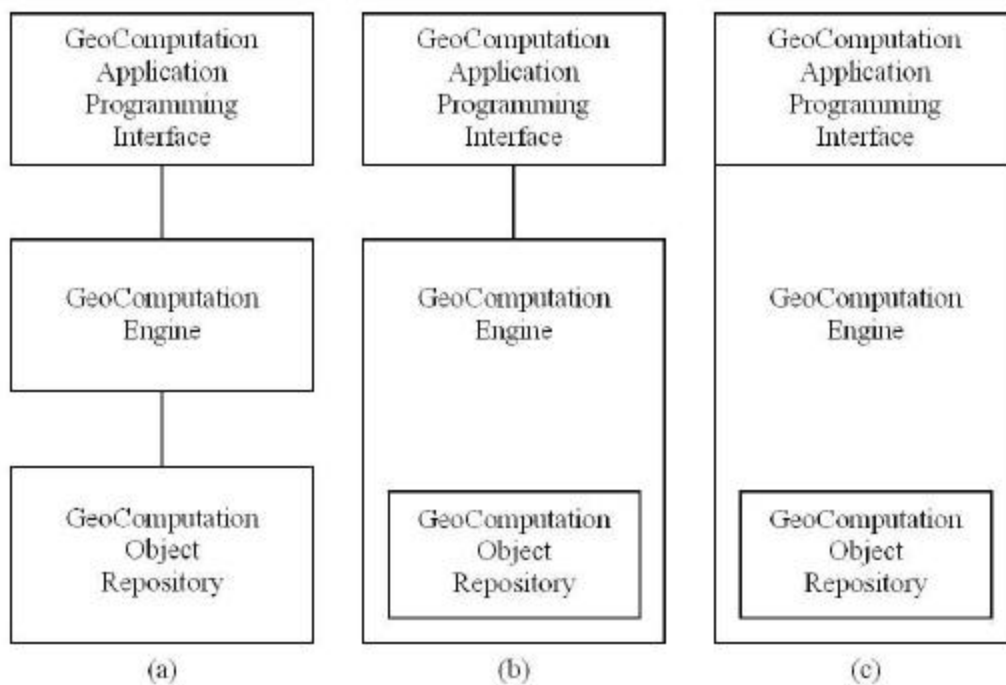


Figure 1. Architecture configuration options

In large scale distributed architecture such as the grid architecture, the three components of the JGC specification may be implemented as a set of services in the Middleware Layer, which intelligently directs user applications in the higher level Application Layer to appropriate computing, storage or other resources in the lower level Resources Layer. These services facilitate a flexible and efficient delivery of GeoComputation functionalities depending on the computational, data or other constraints. Figure 2 shows how three-tier implementations of the JGC specification can be deployed within a grid architecture. GeoComputation Engine nodes allow access to all kinds of GeoComputation services. GeoComputation object repository nodes stores objects supporting GeoComputation functionalities such as configuration and algorithm parameters, data input and results. The data input to GeoComputation task may be prepared by geographic information services or directly from spatial data stores in the grid. Index server nodes maintain a catalog of available GeoComputation services and objects in the grid. User applications conforming to the GeoComputation API look up the GeoComputation services and objects they need from index servers, create GeoComputation tasks using those services and objects, and request the tasks be processed by the high performance computing resources in the grid and results be stored back into GeoComputation object repository nodes. User applications may retrieve GeoComputation results from GeoComputation object repository nodes and present them to end users through GUIs.

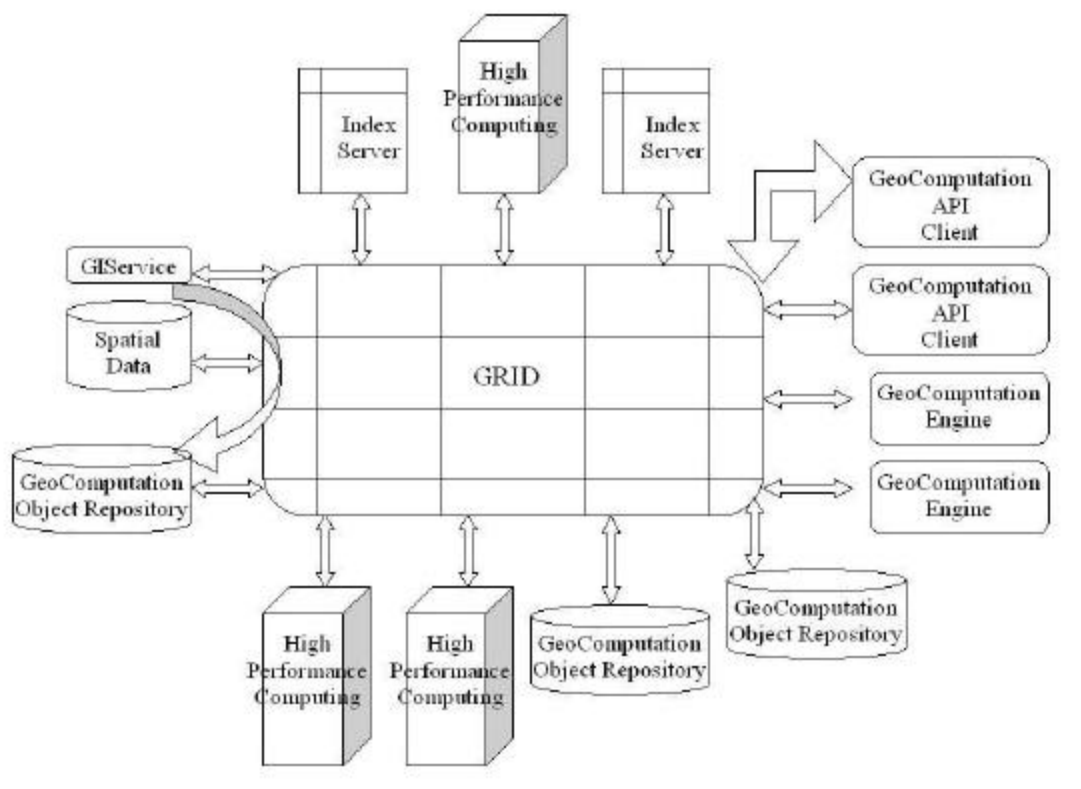


Figure 2. Deploying JGC implementations in a grid architecture

An implementer of the JGC specification may add additional utilities and management interfaces to the GCE and GCOR to enhance its JGC conformant product. However, these additional components are not part of the JGC specification but privately owned by the implementer who has

the exclusive legal right. A JGC implementer may also choose to implement a subset of the complete JGC specification to support only portions relevant to his problem domain. The JGC specification does not put any limits on the GCE and GCOR design or implementation as long as the functionality required by the GeoComputation API is supported.

3.4 Use cases

The JGC specification has many possible uses, which can be illustrated by use cases that provide a context for understanding how the JGC specification can be used. Depending if we are from a perspective of end users of JGC conforming products or a perspective of their software vendors, we can have two types of use cases: application use cases and vendor use cases.

3.4.1 Application use cases

We present these end-user use cases to help application developers explore the various situations in which the JGC specification can be used.

- **GeoComputation GUI:** A cartographer uses neural network models for predictive forest mapping. To have a better control over the use of neural network models, he contracts a software developer to produce a GUI for his JGC specification conformant neural network tool. By referring to the API definitions in the JGC specification, the developer develops a GUI that allows the cartographer to interactively adjust the control settings and traverse and visualize the neural network models built. The GUI is even able to show hidden layers and weights on connections of the neural network models built.
- **Selecting the best algorithm:** a GIS consulting company is contacted by the state department of land surveying to develop a software that automatically processes satellite images and aerial photos and builds a preliminary land use classification. The team lead of this project wants to find out the best classification algorithm among neural network, decision tree and maximum likelihood. The data analyst for the company has previously built a monolithic GeoComputation application that conforms to the JGC specification and supports those three classification algorithms. So the data analyst brings up the GUI of the application, loads the three classification algorithm components and a data source component that supplies the test data stream. Then he starts the classification operation and hot swaps the selected classification algorithm among the three algorithms. As he switches algorithms the corresponding accuracy measures for each algorithm displays simultaneously on the GUI. Then the criterion for selecting the best classification algorithm is the one that consistently gets the maximum accumulated accuracy rate.
- **Minimal top level specification:** A college student is working on his senior ecology thesis and wants to cluster the data he collected on moth species of Maine to categorize the distribution pattern of those moth species. Though he knew he could use GeoComputation tools to do it, he is not sure about the details. So he browses the Internet, reads about the JGC specification and locates a public domain GeoComputation server that provides clustering services and conforms to the JGC specification. After going through a simple instruction page, he writes a simple Java client program to connect to the server to supply his data and request a clustering operation. The server automatically applies the appropriate clustering algorithm with default settings to his data and sends back the clustering result to him.
- **Web services:** Arm Inc. offers a sophisticated association rule mining service that efficiently discovers associative relationships from categorical or ordinal data. SpaCluster Inc. provides a range of spatial clustering services that allow customers to cluster spatial data. Both Arm Inc.'s

and SpaCluster Inc.'s services are available as web services so that their customers can integrate those services seamlessly into their own GeoComputation systems using the internet. An urban geography researcher wants to mine spatial association rules in the US 2000 census data. To do that she needs to convert the numerical geographical coordinates for each census tract to categorical values generated by spatial clustering. And she then has to perform association mining on the spatial category column and other categorical or ordinal census data columns. Since both Arm Inc.'s association mining service and SpaCluster Inc.'s spatial clustering services conform to the JGC specification, she is able to have her contract developer build a system that seamlessly incorporates both companies' web services and mines spatial association rules for her.

3.4.2 Vendor use cases

These use cases show how GeoComputation software vendors can take advantage of the JGC specification in commercial implementations.

- Broad support of JGC: A GeoComputation software vendor implements a wide range of algorithms that address all JGC GeoComputation functions in its product. To make its product easy to use for unsophisticated GeoComputation application users, the vendor provides an automatic configuration feature that can intelligently select appropriate algorithms and relevant algorithm-specific control parameters according to the user's desired GeoComputation operation.
- Narrow support of JGC: A GeoComputation software vendor Cellular Automata, Inc. (CAI) supports a series of cellular automata algorithms, either developed by itself or third parties. These algorithms can help people solve a variety of geographical problems. The vendor chooses to conform to the JGC specification to increase the popularity of his products in the GeoComputation software market.

4. Implementation

To demonstrate the utility of the proposed JGC specification, we adapted and implemented much of the new Java Data Mining API (JDM, JSR73) as the data mining part of the future GeoComputation API. We use this lightweight implementation of the proposed JGC framework as an example to show how the proposed JGC framework allows us to seamlessly incorporate a range of data mining functions into existing GeoComputation tools to enhance their problem-solving capabilities.

The JDM API supports every aspect of data mining activities ranging from functional areas like classification, clustering and association, to learning algorithms like decision trees, neural networks, K-Means and Apriori, and to common operations like model build, test, and apply. Despite the large number of available features in the JDM API, we only implemented a portion of it to support the supervised classification and unsupervised clustering of geospatial data, via a suite of classification and clustering methods that includes k-means, k nearest neighbour, maximum likelihood, linear regression, linear discriminant analysis and quadratic discriminant analysis. Since the JDM API only covers a limited set of data mining algorithms, we had to expand it by defining new software interfaces representing algorithm-specific settings and model details for those unsupported mining algorithms including k nearest neighbour, maximum likelihood, linear regression, linear discriminant analysis and quadratic discriminant analysis. Cartographic classification algorithms

like quantiles, equal intervals and standard deviation are useful tools for cartographers to make univariate and bivariate data maps. However, the JDM API does not cover them either. So we adapted the JDM API to support one and two dimensional cartographic classification operations and those cartographic classification algorithms. Because the JDM API doesn't sufficiently support customized or interactive classification process during which user may want to intervene directly with the classification criterion or move back and forth between the classification results and pre-classification state to find the optimal settings, we also expanded the JDM API with new software interfaces that help maintain and exchange state information of the classification process.

We highlight the benefits of using our implementation of the JDM API by showing the ease with which our classification and clustering tools can be accessed, configured, swapped, and connected to other supporting tools such as map visualization. Figure 3 shows how client applications interact with the Data Mining API.

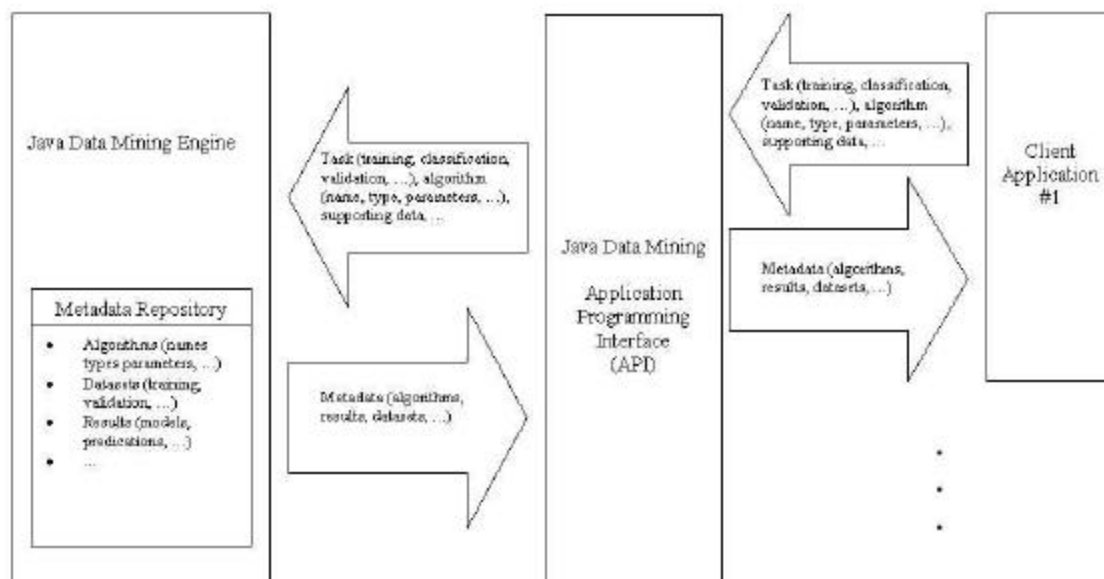
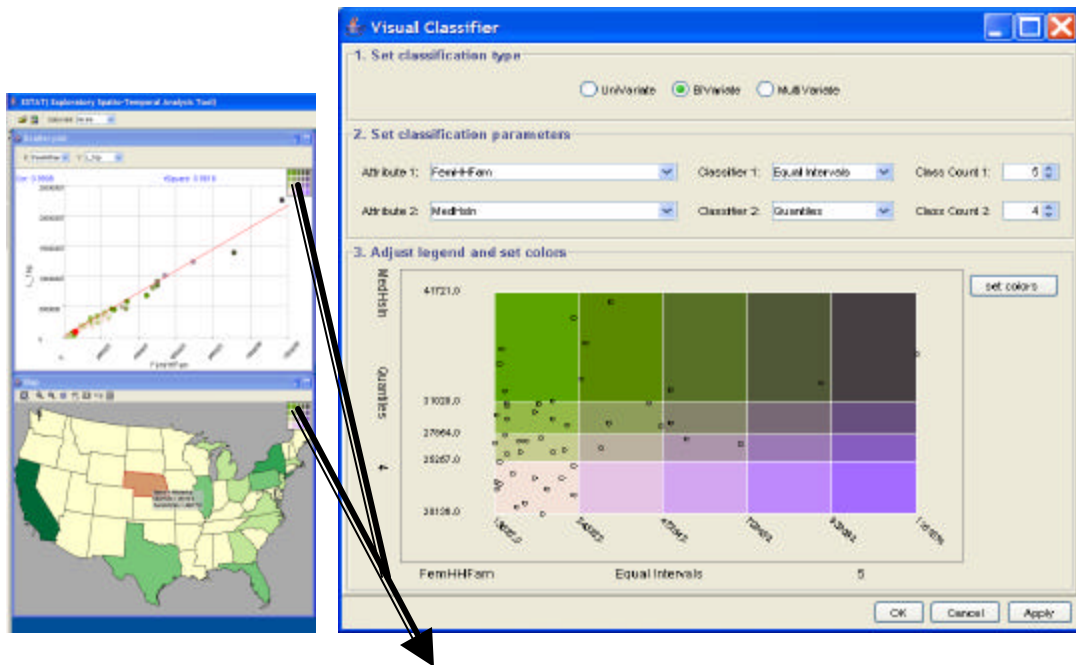


Figure 3. Interaction between user applications and the JDM API

We experimented with integrating data mining functions through the JDM API with a real world GeoComputation application developed at the GeoVISTA center, the Exploratory Spatio-Temporal Analysis Tool (ESTAT, a collection of GeoVISTA *Studio* components). Figure 4 shows the user can access one or two dimensional cartographic classification and multi dimensional clustering services via the JDM API from applications such as scatter plot and map visualization through a GUI.



Clicking on the color legend brings up the classification interface

Figure 4. ESTAT users can access classification and clustering services via the JDM API

Figure 5 shows the GUI of another client application that allows access to a set of supervised classification services through the JDM API. Developers of both ESTAT and the client application have minimal knowledge of the technical and implementation details of the underlying JDM API. By referring to the JDM API, they are able to build GUIs for users to access the data mining functions provided by the JDM API. These GUIs are also able to let users to adjust and visualize algorithm-specific settings and model details. The users can hot swap among the available data mining services to freely experiment with them. Customized and interactive classification process is also supported to allow user to find the optimal algorithm settings. In each of those two applications, a variety of data mining methods are seamlessly integrated together to be a comprehensive system. Though all those methods are from the same vendor GeoVISTA center, methods from third parties can also be seamlessly integrated as long as they conform to the JDM API. Similarly, methods from the GeoVISTA center can be smoothly incorporated into other implementations of the JDM API. While we implemented the JDM API as a monolithic system, it can be easily converted to a distributed architecture. For example, it may be implemented as a client-server architecture, where the data mining engine is implemented as a dedicated data mining server that provides data mining services to client applications through the JDM API; or it may also be implemented as a three-tier architecture, where the data mining engine is implanted as a distributed data mining web service on a high performance computing server while the data mining object repository resides in a database server and the JDM API conformant client application runs from a separate work station.

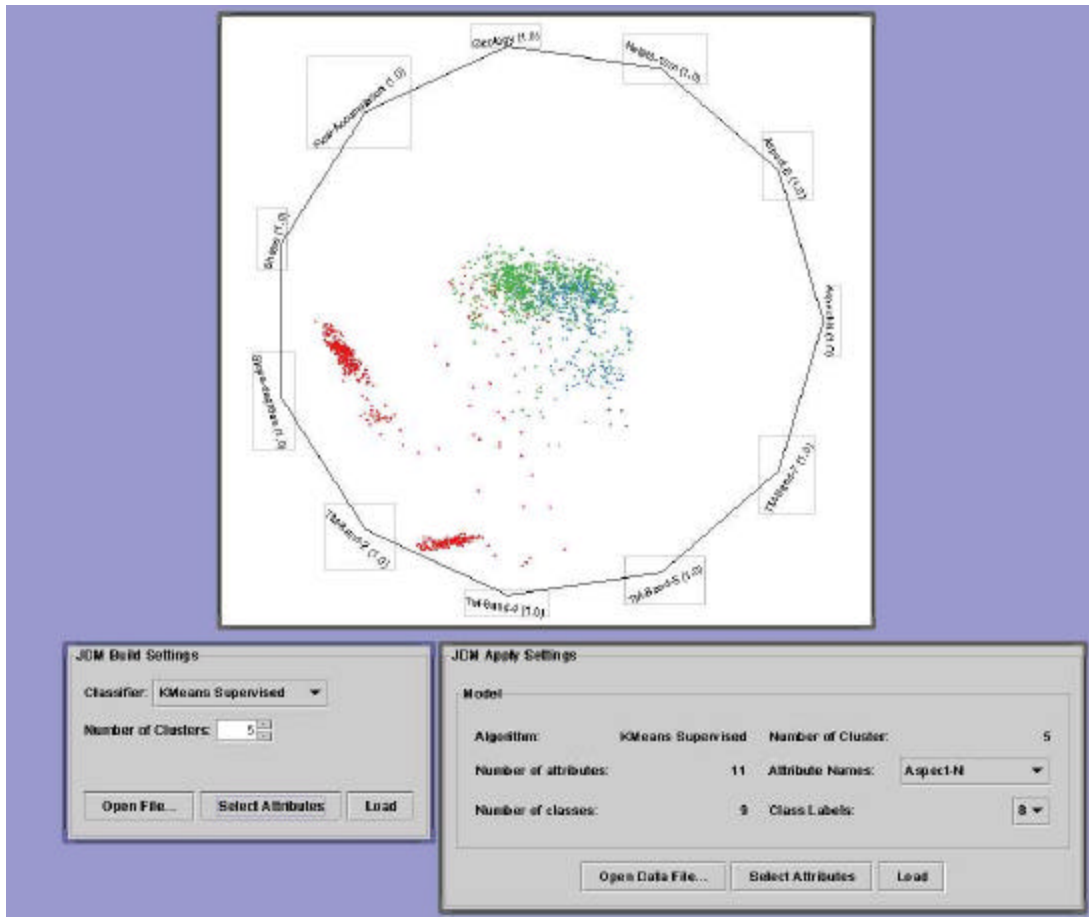


Figure 5. GUI of a client application that provides access to supervised classification services through the JDM API

5. Conclusion

This paper proposes to specify an open and extensible Java GeoComputation API to support the development of highly interoperable, reusable and flexible GeoComputation systems. The proposed API framework offers a standard operating environment that aids the interchange of GeoComputation data, models, methods, training and validation techniques and results among the GeoComputation community. Within this framework, all GeoComputation applications will be coded by developers against a standard API and accessed by end users through the same standard API. Technical and implementation details of GeoComputation systems are shielded from unsophisticated users. GeoComputation tools from different vendors can be seamlessly assembled into one comprehensive system. Users of GeoComputation applications can experiment freely with a toolbox of methods from a variety of providers. This contributes significantly towards a better GeoComputational interoperability, especially when most similar studies only focus on increasing the interoperability of geospatial data and services producing it instead of the essence of GeoComputation revolution “computational methods”.

We envision that not only will the web be the primary channel for interchange and sharing of

geographic information and geographic information services but much of the GeoComputation operations will shift towards a distributed heterogeneous environment. Increasing interoperability of GeoComputation systems as proposed in our framework becomes very natural.

6. Acknowledgements

The research reported here has been supported in part by grants from the National Cancer Institute (grant # CA95949), and Center for Disease Control (grant # TS-1125).

7. References

- Askew, D., Evans, S., Matthews, R., and Swanton P., 2005, MAGIC: a geoportal for the English countryside. *Computer, Environment, and Urban Systems*, 29, 71-85.
- Beaumont, P., Longley, P. A., and Maguire D. J., 2005, Geographic information portals – a UK perspective. *Computer, Environment, and Urban Systems*, 29, 49-69.
- Bernard, L., Kanellopoulos, I., Annoni, A., and Smits P., 2005, The European geoportal – one step towards the establishment of a European Spatial Data Infrastructure. *Computer, Environment, and Urban Systems*, 29, 15-31.
- Couclelis, H., 1998, GeoComputation in context, In P.Longley, S. Brooks, R. McDonnell and B. Macmillan (Eds), *GeoComputation: A Primer* (Chichester: Wiley), pp. 17-30.
- Gahegan, M., 2000, What is GeoComputation? A history and outline. <http://www.geocomputation.org/what.html>.
- Harvey, F., Riedemann, C., Kuhn, W., Pundt, H., and Bishr, Y., 1999, Semantic interoperability: A central issue for sharing geographic information. *Annals of Regional Science*, 33, 213-232.
- Hornick, M. et al. 2004, JSR73: Java Data Mining API. <http://www.jcp.org/en/jsr/detail?id=73>.
- Maguire, D. J., and Longley, P. A., 2005, The emergence of geoportals and their role in spatial data infrastructures. *Computer, Environment, and Urban Systems*, 29, 3-14.
- Openshaw, S., 2000, GeoComputation, In S. Openshaw and R. J. Abrahart (Eds), *GeoComputation* (London: Taylor & Francis), pp. 1-31.
- Openshaw, S., 2000, GeoComputation research agendas and futures, In S. Openshaw and R. J. Abrahart (Eds), *GeoComputation* (London: Taylor & Francis), pp. 379-400.
- Schuurman, N., 2002, Flexible standardization: Making interoperability accessible to agencies with limited resources. *Cartography and Geographic Information Science*, 29, 343-353.
- Takatsuka, M., and Gahegan, M., 2002, GeoVISTA *Studio*: A codeless visual programming environment for geoscientific data and visualization. *Computers and Geosciences*, 28, 1131-1144.
- Tait, M. G., 2005, Implementing geoportals: applications of distributed GIS. *Computer, Environment, and Urban Systems*, 29, 33-47.
- Visser, U., Stuckenschmidt, H, Schuster, G., and Vogele, T., 2002, Ontologies for geographic information processing. *Computers and Geosciences*, 28, 103-117.