Transactions in GIS, 2002, 6(4): 355–381

Review Article

A Dynamic Architecture for Distributing Geographic Information Services

Ming-Hsiang Tsou
Department of Geography
San Diego State University

Barbara P. Buttenfield
Department of Geography
University of Colorado

Abstract
Traditional GISystems are no longer appropriate for modern distributed, heterogeneous network environments due to their closed architecture, and their lack of interoperability, reusability, and flexibility. Distributed GIServices can provide broader capabilities and functions for data management, browsing, and exchange. This paper introduces a dynamic architecture for distributing GIServices. The term dynamic indicates that the architecture is constructed temporarily by connecting or migrating geodata objects and GIS components across a network. The intention of the paper is to overview components and protocols necessary for a workable implementation of dynamically linked GIServices. The paper introduces a metadata scheme for both geodata objects and software components, and proposes an implementation framework based on existing languages, computing architectures and web services. In the framework, GIS nodes form the basic processing unit. All GIServices can be accomplished through collaboration between GIS nodes. The design of the presented framework emphasizes scalability, reusability, and dynamic integration. Current distributed computing environments cannot fully support dynamic architectures for technical and other reasons. Throughout the discussion, we distinguish what currently can be implemented from what cannot. We summarize costs and benefits of adopting a dynamic GIServices paradigm in a final section.

1 Making the Case for Geographic Information Services

The development of Geographic Information Systems (GISystems) is highly influenced by the evolution of information technology. Due to the popular use of the Internet and the dramatic progress of telecommunications technology, the paradigm of GIS is

Address for correspondence: Ming-Hsiang Tsou, Department of Geography, San Diego State University, CA 92182, USA. E-mail: tsou@mail.sdsu.edu

shifting. Traditional GISystems provide capabilities to handle georeferenced data, including data input, storage, retrieval, management, manipulation, analysis, and output (Aronoff 1989). However, with closed and centralized legacy architecture, current GISystems cannot fully accommodate distributed, heterogeneous network environments due to their lack of interoperability, modularity, and flexibility. With advances in computer networking technologies, a distributed geographic information services (GIServices) paradigm becomes a reachable goal, albeit one that requires fundamental changes in architectural design.

The purpose of an *information service* is to provide information in an appropriate form for a particular task, which requires selection and abstraction (Shuey 1989). Information services include tools for data management, browsing, access, cleaning, processing, interpretation, presentation, and exchange (Buttenfield 1998, p. 161). *Geographic Information Services* open a wide range of on-line geospatial applications, including for example digital libraries (NSF 1994), digital governments (NSF 1998), digital earth (Goodchild 2000), on-line mapping (Peterson 1997, Kraak and Brown 2001), data clearinghouses (Peng and Nebert 1997), real-time spatial decision support tools (Craig 1998), and process modeling (Huang and Worboys 2001).

Provision of GIServices can synergize information sharing and may speed the diffusion of GIS technologies into communities currently identified as non-adopters. On-line GIServices encourage multidisciplinary cooperation between the GIS community and the computer science community, as well as collaborations between industry and science (e.g. Li 1996, Zhang and Lin 1996, Plewe 1997, Buttenfield 1997). For example, GIServices provide digital library resources to dispersed populations (Goodchild 1997). Prototype on-line GIServices provide a virtual classroom for distance learning (Buttenfield and Tsou 1999, Petrik 2002). Early on-line GIService application examples include the Xerox Map Viewer (Putz 1994) and GRASSLinks (Huse 1995). The Alexandria Digital Library Project (Buttenfield and Goodchild 1996, Frew et al. 1998) adopted Java to implement GIServices such as spatial queries, map browsing, and metadata indexing.

Major impediments to adoption of dynamic GIServices include technical, institutional and economic factors. An example of such factors is the lack of interoperable component technologies in current GIS design. Another factor is the hesitance to adopt interoperability standards. A third factor relates to desires to protect either individual or commercial intellectual property. It would be an oversimplification to place blame: these impediments are real, and arguments on both sides are in many cases valid and compelling. It is useful and even encouraging however to stand back for a moment from the debates; to acknowledge how close the GIS community actually stands to achieving a fully operational dynamic architecture for distributing GIServices across networked environments; and to identify what impediments may be resolved immediately given current states of knowledge and technology.

Many information services have matured in the computer science community. Such services can be adapted to provide intelligent development environments and to distribute GIServices dynamically. For example, intelligent software agents (Maes 1994, Bradshaw 1997) could provide a flexible method for searching heterogeneous GIS data and accessing programs across networks. Deployment of a problem-solving environment, or PSE (Walker et al. 2000, Wright et al. 2000) could provide an integrated software environment for rapid prototyping of scientific visualization systems and expert systems. Many essential functions of distributed GIServices will

benefit from PSEs, including knowledge repositories, sophisticated execution control systems, and visualization environments. A third very promising area of information services technology is the provision of web services. The goal of web services is to combine multiple programs distributed in different network locations and to provide integrated functions and services for their users (Caudwell et al. 2001, Newcomer 2002).

The web service concept is similar to the dynamic architecture proposed in this paper except in one important respect. Existing web service solutions focus on technology-oriented solutions. This paper argues for a task-oriented solution that emphasizes operational metadata, dynamic relationships between data objects and software components, and deployment of intelligent software agents.

This paper proposes a dynamic architecture to facilitate delivery of flexible GIServices under a distributed network environment. The new architecture should be technology and application independent. It should distribute GIServices without constraining client-side hardware or operating systems. It should be easy to modify, and it should retain computing resources locally and across networks only for the time needed to complete a requested task. Finally, the architecture should be both robust and secure, able to withstand the vagaries of network breakdown. The paper will overview elements of the architecture, communication protocols, and information exchange requirements needed to make a dynamic architecture work. As stated above, emerging information services bring us closer to achieving this goal. Technical and institutional barriers must still be overcome. The paper will identify these barriers as it describes one possible solution.

2 Extending Current GIS Architectures

The paradigm of dynamic GIServices extends two existing architectures in several ways (Figure 1). In the first (Traditional) architecture, computations occur in centralized repositories holding interfaces, programs and data. Each element is embedded and cannot be separated from the remainder of the architecture. Modeling remains platform- and application-dependent. Migrating a model or application from within a traditional architecture into a different operating system or platform is difficult. The traditional architecture is what most GIS users encounter today.

The second (Client/Server) architecture is based on generic network design. Many current implementations of 'Internet-based GIS' are built on this type of architecture. Client-side database components and program components are separated from server-side components. This architecture allows clients to access a server remotely by Remote Procedure Calls (RPC), or by techniques such as Open Database Connectivity (ODBC). Client-side components are usually platform-independent, requiring only an Internet browser to run. However, each client component can access only one (pre-) specified server at a time. Software components differ on client machines and server machines, and are not interchangeable: a client is always a client, and lacks many functions contained in a specialized server environment. Servers come with different connection frameworks that cannot be shared.

In the third (Distributed) architecture, GIServices are built upon a more advanced networking scheme. The significant difference is the adoption of distributed component technology, which can interact with heterogeneous systems without the

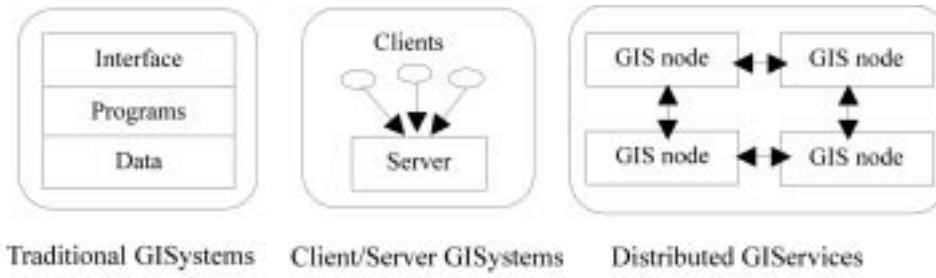

Figure 1 Three types of GIS architecture

constraints of traditional client/server relationships (Montgomery 1997). Under a distributed architecture, there is no difference between a client and a server. Every GIS node can act as a client or server based on the task. A client is simply defined as the requester of a service. A server (likewise) is simply the machine that provides the service. This architecture permits dynamic linkages between data and software. In fact, the architecture is very similar to what is called '*peer-to-peer*' computing (P2P), which permits personal computers or workstations to communicate directly with one another, without a server (Roberts-Witt 2001). The difference between P2P computing and the distributed architecture is that P2P can allow only *one-to-one* or *one-to-many* communication. A fully distributed GIService architecture has to permit *many-to-many* communications among computers; and here lies a current technical barrier to be overcome.

The driving force behind the extension of GIS architectures is the availability of new technology, especially in the context of languages and network technology. New languages such as Java and C# (C-sharp) support platform-independent applications across the Internet. The Java language, developed by Sun Microsystems, Inc., is a pure object-oriented language, designed to enable the development of secure, high performance, and highly robust applications on multiple platforms in heterogeneous, distributed networks (Gosling and McGilton 1996). One important advantage of Java is the development of Java Bean technology, which can be used for creating reusable, embeddable software code. The GeoVISTA Studio developed at Penn State University (www.geovistastudio.psu.edu) is one 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). C# is another possible language for developing distributed services. C# provides similar features to Java including inheritance, encapsulation, abstraction, and polymorphism. C# is a generic language developed under the .NET framework by Microsoft (Pleas, 2000). .NET is a next-generation-web-service framework which will be described later.

Three advanced network technologies support distributed computing environments (Orfali and Harkey 1997) and peer-to-peer computing (Roberts-Witt 2001). These technologies include the Common Object Request Broker Architecture (CORBA), the Distributed Component Object Model (DCOM), and Remote Method Invocation (RMI). Distributed component technology allows clients to access heterogeneous servers dynamically, which is an essential feature of distributing GIServices. They operationalize the maxim that 'the network is the computer'.

One may question why dynamic GIServices have not been implemented yet given the existence of appropriate coding languages and distributed component technology. Currently, many on-line GIS projects (correctly) emphasize standardized, interoperable data (Bishr-Yaser 1996, Sondheim et al. 1999). What is not a focus, and should be, is the interoperability of GIServices, that is, of the code that processes the standardized, interoperable data. Most current GIServices are technology-specific. When the technology changes, and it always does, existing services are very difficult to migrate into the new framework. Very often, services have to be abandoned. Information service software is expensive to develop and more expensive to redevelop. This provides one of the most compelling arguments to design an architecture that can be upgraded readily.

But herein lies a second barrier to adoption of a dynamic GIS architecture, with technical, economic and institutional concerns to resolve. A fully open architecture with completely modular code would permit anyone to 'plug-and-play', inserting modifications and potentially revising the code completely. There are issues of software security, of ethics and responsible programming, and intellectual property issues related to re-use of code, as well as commercial impacts to opening proprietary software designs.

Notwithstanding widespread technical and institutional dilemmas, it is important to stay mindful that emerging technologies are becoming available to overcome technical impediments. The remainder of this paper introduces a dynamic architecture for distributing GIServices from a task-oriented perspective. The term dynamic indicates that the architecture is constructed temporarily by connecting data objects and software components across networks. The GIServices architecture would be organized on the fly according to the specific task. Dynamic construction is achieved in three phases, by modularizing specific services as LEGO-like components, by embedding an object-oriented metadata scheme, and by implementing agent-based communications. Each will be described in turn. The dynamic architecture will be demonstrated in a GIS scenario. Before describing the architecture, a brief history of distributed components research will help to put the proposed paradigm and architecture in context.

3 A Brief Chronology of Distributed Computing and GIS

Developments in two separate areas support the technology underlying dynamic architectures for GIServices. The first is the emergence of distributed components, which underlies software coding. The second is the demonstration that GIServices can be distributed via the Internet, through numerous on-line GIS applications.

3.1 The Development of Distributed Components

The original idea of distributed components came from Computer Science. Generic distributed components adopt features of object-oriented modeling, including encapsulation, polymorphism, inheritance, object-binding, and object relationships such as specialization, collaboration, and composition (Rumbaugh et al. 1991, Taylor 1992). Distributed components are constructed by partitioning the client and server sides of an application into self-contained units that can interoperate across networks,

integrating languages, applications, tools, and operating systems. The capabilities of distributed components include roaming agents, rich data management, abstract and generalized interfaces, intelligent self-managing entities, and intelligent middleware (Orfali et al. 1996).

The current commercial market provides three major infrastructures supporting distributed component technology: (1) the Common Object Request Broker Architecture (CORBA) developed by the Object Management Group; (2) the Distributed Component Object Model (DCOM) developed by Microsoft Corporation; and (3) the Java Platform technology developed by Sun Microsystems, Inc., and its subsidiaries, Sunsoft and Javasoft. CORBA, DCOM and Java can distribute low-level services, for example migrating data objects between machines, or global object naming. Higher-level services are also needed however, as for example the ability for an object to self-describe in standard format, the ability for a machine to broadcast (or respond to) a request for a specific service. These high level services are not yet available or at least not fully robust, and this presents another technical impediment to dynamic architectures.

Currently, web services comprise the most exciting developments in distributed component technologies. Web services technology is derived from CORBA and DCOM technology (Caudwell et al. 2001). Web services are formed by the integration of several key protocols and standards: XML (Extensible Markup Language), WSDL (Web Services Definition Language), SOAP (Simple Object Access Protocol), and UDDI (Universal Description, Discovery, and Integration). Web services are modularized applications that are self-describing and contain loosely bound functions and programs (Caudwell et al. 2001). The power of web services is to combine these elements under a single user-friendly running environment using a web-based user interface. For example, a user can combine an on-line mapping function, a point-of-interest database, and a hotel reservation system to design a three-day vacation itinerary from his/her web browser. An example of such a web service currently under development by Microsoft is called MapPoint.Net. MapPoint.Net provides mapping modules and real-time geospatial information (such as road construction, weather reports, etc.) in a form to integrate with other applications, such as car navigation, decision support systems, location-based services, etc. (Brûlé 2002). One can readily see the great potential for the integration of web services into GIS applications, for example emergency management and hazards mitigation.

The .NET acronym refers to a newer ('next generation') distributed component technology developed by Microsoft, that enables software 'building blocks' that exchange data and services between heterogeneous computing environments (Brûlé 2002). The C# language was developed as a part of the .NET toolkit. Using .NET web services, programmers can combine multiple languages in a single service. For example, a .NET application might utilize Java Swing API for the design of its graphic interface, C# for its buffering function, and Perl scripts for text-based attribute queries. With multiple language capabilities, existing high-level services can be integrated, avoiding the necessity to either recode or force the entire community to adopt a single coding language. In this regard, .NET technology provides tools to overcome both a technological as well as an institutional barrier.

One important feature of web services is improved access to distributed components on both the client side and the server side. A single machine can play a server's role or a client's role. For example, a GIS site in Colorado can access multiple

federal database servers as a client. When other GIS projects request data about Colorado, the same GIS site that was a client to federal servers can serve data to other users. In general, distributed component technologies eliminate the constraint that a machine must be a client or a server, to permit novel approaches to designing software solutions. 'These shifts are not simply due to operating in a distributed or networked environment. Rather, great diversity and innovation of information technology accompanies distributed computing which, in turn, brings new models of the world and new ways of solving problems' (Ganti and Brayman 1995, p. 33).

3.2 On-line GIS Applications

Three milestones so far have characterized the history of on-line GIS projects. The Xerox PARC Map Viewer developed in 1994 was the first mapping service prototype on the Internet (Putz 1994). The Xerox Map Viewer provided a preliminary technical solution for distributed GIServices by using a HTTP server and CGI programs. The technical framework underlying Map Viewer was adopted by many early on-line GIS applications. The second milestone, GRASSLinks was the first fully functional on-line GIService (Huse 1995). GRASSLinks illustrated a comprehensive prototype of traditional GIS functions, such as map browsing, buffering, overlay, etc. Both the Xerox Map Viewer and GRASSLinks were designed to mimic traditional GISystem functions. The third milestone was the initial version of the Alexandria Digital Library, that introduced new content for on-line GIServices, first implemented a 'digital library' metaphor for georeferenced data, and provided sophisticated library functions for geospatial information, including collections holding, catalog searching, and metadata indexing (Buttenfield and Goodchild 1996). Although these three examples are widely recognized in the GIS community, each was developed using different database frameworks, and different information technologies. Incompatible architectures and programming methods have prevented integration or sharing with other on-line prototypes. This technical limitation has institutional consequences, since every new online GIS application requires programmers to essentially start from scratch.

The development chronology of on-line GIS indicates that a technology-oriented design is problematic and hinders subsequent development and knowledge sharing. Realizing the lack of a standardized framework, the Open GIS Consortium (OGC) and the ISO/TC 211 Technical Committee were independently formed in 1994 (Buehler and McKee 1998, Rowley 1998). OGC's mandate included two components: full integration of geospatial data and geoprocessing resources into mainstream computing; and the widespread adoption of interoperable software and geodata products throughout the information infrastructure (OGC 1998). The ISO/TC211 Working Group was formed by the International Standards Organization, to emphasize a service-oriented view of geoprocessing that balanced data, task and systems (Kuhn 1997). The establishment of OGC and ISO/TC 211 illustrates GIS community determination to solve the problems associated with integrating on-line GIS applications. Based on publications from both organizations, the OGC's and ISO/TC 211's specifications to date focus on interoperable data, but do not as yet address interoperability of processing mechanisms. Without consideration of both, many implementation problems can be expected, for reasons given in the first section of the paper.

4 Designing a Dynamic Distributed GIService Architecture

Let us reiterate that the presentation of a dynamic architecture is intended as much as a proof of concept to show existing technologies are available that could achieve a dynamic architecture, as it is a platform for discussion of what technical and institutional impediments remain. The concept of a dynamic GIService architecture is derived from LEGO-like distributed software components. The LEGO metaphor refers to the well-known children's toy blocks that can be interlocked and stacked. Similar to LEGO blocks, the intention is to stack and interlock GIServices modules to form a dynamic GIS package shaped specifically for a given task. The LEGO architecture may persist only briefly, for completion of a single GIS task. Then the LEGO modules disperse, to be rearranged and restacked in a different configuration for a different GIS task. The discussion that follows presents object frameworks supporting data and services, communication strategies by which service requests are broadcast and responded to, and a metadata scheme enabling data and software components to initiate activity. In a fully dynamic architecture, GIS data and software components can be moved, combined and shared across distributed networks. The overall objective is to shift the GIS paradigm from a monolithic, inflexible approach to a modularized, plug-and-play approach.

4.1 Object Frameworks Supporting Distributed Data and Services

The establishment of GIServices is collaborated among several 'GIS nodes', that is, a group of network-based GIS workstations (Figure 2). GIS nodes incorporate two types of information. The first type, *geodata objects* are information entities that identify the geographical location and characteristics of natural or cultural features and boundaries of the Earth (Buehler and McKee 1996). Geodata objects are encapsulated in object-oriented structures, which may be vector-based or raster-based.

The second type of GIS node, *GIS software components* are 'ready-to-run', modularized programs that are dynamically loaded into a network-based system to enable GIS functionality. For example, a 'GIS buffering component' will generate a buffer around a selected data object for the targeted GIS application. The term 'GIS components' often refers to data objects (Orfali et al. 1996). However, this paper uses

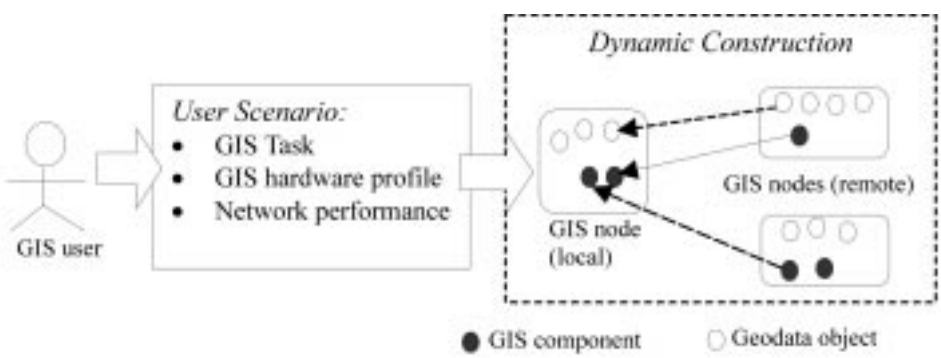

Figure 2 Dynamic construction of distributed GIServices by node collaboration

the term to refer to software in order to distinguish these elements from 'geodata objects'.

In an ideal computing environment, GIS data objects and software components would interact dynamically to generate GIServices and accomplish different GIS tasks, using protocols discussed below. The coordination of distributed components commonly requires a shared data and action schema, that is, detailed *a priori* ontological commitments on behalf of service developers and users. At present consensus does not exist in spite of continuing efforts on the part of OGC, ISO, and other standards organizations.

These two types of nodes form the LEGO building blocks. Their configuration, that is, the architecture, is dynamically constructed based on three factors. The first is the task to be performed. Different GIS tasks require different types and arrangement of GIServices. For example, the architecture responding to a task to update urban census geography will focus on GIS database functions. An architecture for hydrologic modeling will require access to GIS analysis procedures.

GIS-node hardware profiles are the second factor. Hardware specifications including CPU speed, RAM, available hard disk space will dictate what GIServices can be delivered locally, and what services may be optimally run on a thick or thin client. GIS node availability will also be important. Different strategies for distributed GIService architecture must depend on the available profiles. The way in which a particular strategy is selected is discussed below in the section describing intelligent agents.

Networking performance is the third essential factor in configuring a dynamic architecture. Different bandwidth and connection types such as Ethernet, ATM, DSL, or Cable modem services require different configuration and deployment strategies. For example, a distributed GIService running on a 56K bandwidth connection may have difficulty in uploading huge data sets to another node. A GIService networked through a 100 MB Ethernet connection would not have this problem.

Figure 3 illustrates a hypothetical scenario where a GIService architecture is constructed dynamically by configuring GIS components and geodata objects among several GIS nodes. In the scenario, a GIS user (Mike) needs to display a Colorado Road map on his GIS node. He submits a task request and the local node (A) connects to other GIS nodes (B and C) that hold GIS software components (black circles) and geodata objects (white circles). The data and software objects on node B are copied over to node A. The GIS module on C is not copied, but run across the network in distributed fashion. This could be due to hardware or to networking factors, as described above. On completion of Mike's GIS task, the data objects and components are restored to their original status. Copies of data and software migrated from node B are deleted, and the connection to node C is terminated. The architecture exists only long enough to support the requested task, following which the nodes are de-coupled to wait for the next request.

There is much in this hypothetical scenario to account for. Nodes must either have knowledge about each other, or be able to broadcast information on hardware profiles, networking, and available resources. GIS tasks must be clearly defined and modularized to the point where a node can be dynamically defined as a thick or thin client. In general, network routing or location modeling will run more effectively on a thick node, as complicated calculations and algorithms may be more efficiently handled without an intervening network. For some tasks the choice of thick or thin

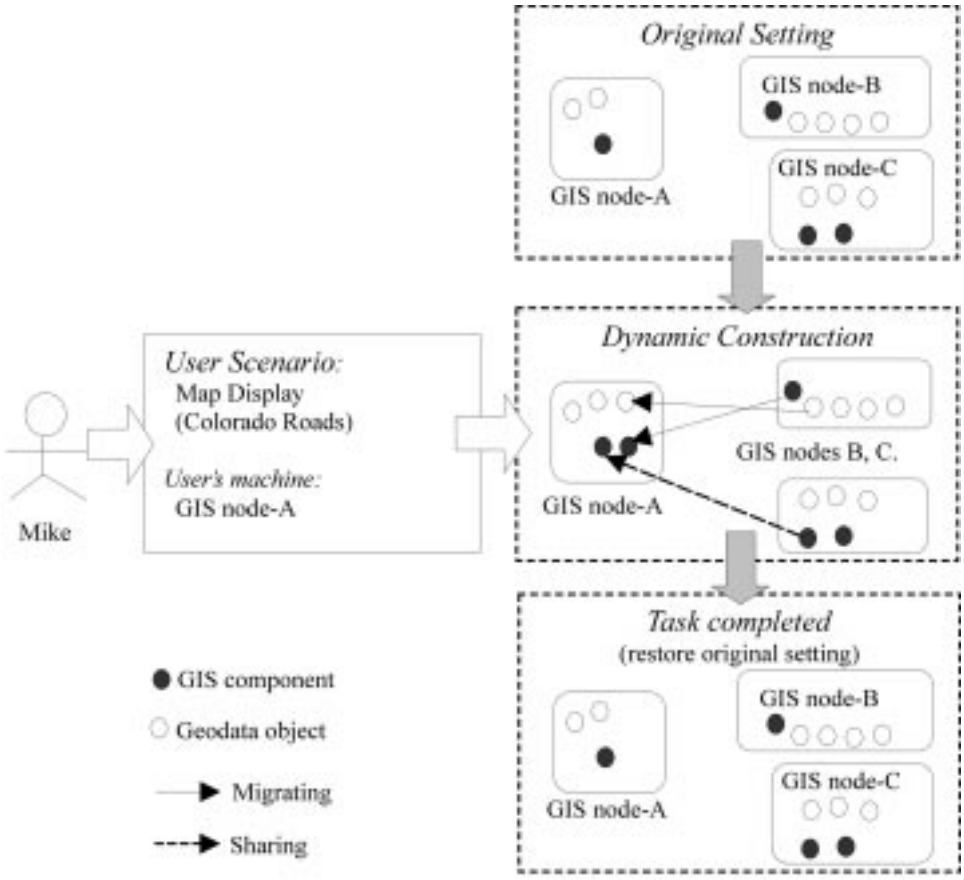

Figure 3 Building GIServices 'on-the-fly'

node may be over-ridden. For example, a local thin node may be appropriate to handle map display services, permitting the GIS user to take over intuitive decisions on graphic design and layout.

Another consideration for this scenario is the establishment of *a priori* semantic consistency. To re-locate or share distributed geodata objects and GIS components, every GIS node should agree upon several pre-defined rules. For example, if two data objects reside on two GIS nodes, which one should a task use? From an ontological standpoint, the question becomes how to determine whether two data objects are the same, or different. These ontological rules are nontrivial to define and may be a barrier far more challenging than technical issues of software engineering.

The balance of functionality between nodes is another critical issue in configuring dynamic architectures. Currently, many research papers argue about which client model is appropriate for specific GIServices (e.g. Vckovski 1998, Tsou and Buttenfield 1998a, Huang and Worboys 2001). The section below proposes a task-based approach to solve the dilemma.

4.2 Network Communication Strategies for Constructing Dynamic GIServices

Two network strategies can support dynamic configuration of distributed GIServices. The first strategy is object migration, where data or programs move from one node to another. The transferred data may or may not be deleted on task completion. In the second strategy, remote connection establishes a communication channel between two nodes, allowing data and services to be shared across the channel. Either can be applied to distribute geodata objects or GIS software components. Although the network technologies used in object migration and remote connection are quite different, the goal is the same, that is, utilizing available computing resources across networks.

To distribute geodata objects dynamically (Figure 4), the objects can be shared or copied to a requesting node. To share objects, the network strategy used is remote connection to establish a link between distributed databases. The connection is established using SQL through Application Programming Interfaces (APIs), such as JDBC, ODBC or OLE DB. The second approach (object migration) utilizes an FTP protocol to actually move the data object and save it on the requesting node's local disk. Data migration may require both automated and manual procedures for download and format conversion.

Similarly, either network strategy can distribute GIS software components dynamically (Figure 5). GIS operators may be invoked remotely, using Remote Procedure Calls. Other protocols are also possible, such as Internet Inter-ORB Protocol (IIOP) or Simple Object Access Protocol (SOAP). Technology frameworks that currently support this approach include DCOM, CORBA and Java RMI. It works as follows. A GIS application sends a request to local component services (that is, a service directory local to the node). The local service will use its *client stub* to build a connection with another node's (server) skeleton. The nodes temporarily adopt client and server roles to accomplish the connection. The server-side component services then launch the required GIS component. This scenario assumes that the service is not

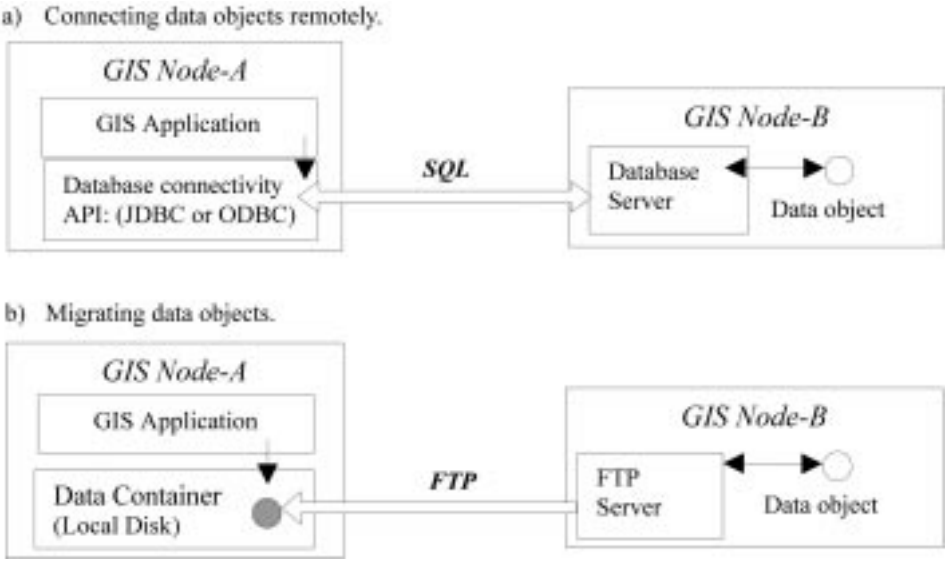

Figure 4 Two types of network connection for geodata objects

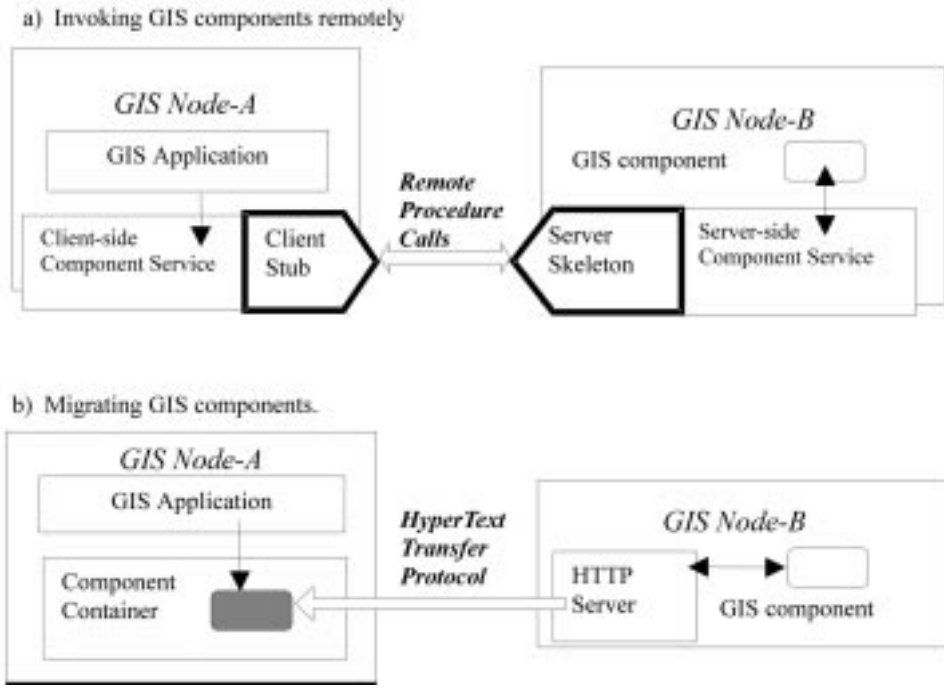

Figure 5 Two types of network connection for GIS software components

locally available, and that the service directory can point to another node where the service actually resides. There will be more discussion about these issues in a moment.

The second network strategy actually moves the software modules (the services) from one node to another. The migration process uses an HTTP protocol to transfer the required GIS procedure dynamically into the targeted GIS application. The downloaded GIS component is stored inside a *container*, which binds with the local GIS application. This approach is currently supported by several technology frameworks, such as Java applets in a Virtual Machine, or Active X containers. It has not been widely integrated into GIS environments as yet.

In general, the requirements for dynamically migrating or connecting GIS components and geodata objects require several things. A decision making process must be in place for identifying nodes with appropriate data and software. A self-describing node framework must be in place for a node to broadcast its directory of services. Each node requires one or more local data containers, and component services that can be distributed. Current technologies such as DCOM, CORBA, and Java can distribute low-level services, for example object migration, global naming, life-cycle management, and object implementation. However, a dynamic GIService architecture will also need high-level services such as a self-describing node infrastructure and an agent-based mechanism to support node broadcasts and other decision processes.

To summarize, network strategies are already in place that can support distribution of data and software by sharing or migrating information, but GIS software designs have not embraced them. Flexible distribution of GIS components can provide customizable services for different users and the architecture can be modified or updated according to specific tasks. For example, users may add a new category of

services, such as 3D presentation. The customization cannot occur, however, without the ability of data objects, component services and nodes to self-describe and to broadcast on the network what services are available. The self-description problem can be solved using metadata, as described below.

4.3 A Metadata Scheme for Distributing GIServices

In general terms, metadata describes the content, quality, condition, and other characteristics of data. Many recent GIS projects have conducted metadata research (e.g. FGDC 1995, Smith 1996, ISO/TC 211/WG 3 1998). Existing metadata schemes standardize format and adopt traditional relational database concepts, where each metadata item is represented as an individual record. However, these approaches do not scale, and may interfere with interoperability (Baldonado et al. 1997). Ironically, standardization of metadata formats may undermine the distribution of GIServices. For example, a single metadata profile has proven inadequate to simultaneously describe both a TIN data model and a raster data model, as evidenced by the numerous profiles and extensions published by ISO/TC 211 (Kuhn 1997, OGC 1998). Likewise, a single metadata profile will likely be inadequate to describe both interpolation and buffering services. A single standard for metadata likely will be both cumbersome and inefficient.

An additional problem is that traditional GIS relational database design detaches metadata from associated data, and jeopardizes metadata availability when geodata objects are moved or modified (Tsou and Buttenfield 1998b). Geodata objects with encapsulated metadata permit flexibility, where metadata are tailored to the type of object they describe. Figure 6 compares a traditional detached scheme and an encapsulated metadata scheme.

When a user moves or copies geodata or services, embedded metadata are automatically exchanged. Additionally, encapsulation protects metadata from external environments. Only authorized programs can access the metadata information (as for example with electronic access keycodes). When a new object is generated, it can inherit parent metadata information, and add new metadata information for itself. For example, if a sub-region is clipped from a satellite image, the new image could inherit the information about image resources, sensor types, and resolution from the original image, adding its newly-clipped spatial boundary coordinates. The object-oriented scheme described below meets these requirements, and demonstrates how metadata can be implemented for both geodata objects and software components.

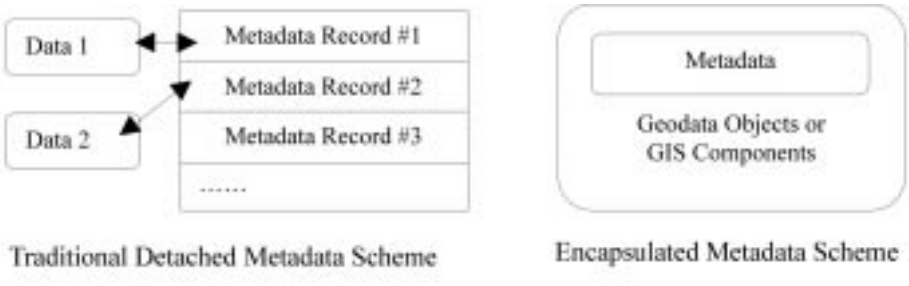

Figure 6 Two metadata schemes (detached and encapsulated)

4.3.1 Metadata for Geodata Objects

Figure 7 illustrates two encapsulated metadata elements for geodata objects. Metadata elements are wrapped inside data objects. These elements are essential for object self-description, remote database connectivity, and object migration. Encapsulation protects the critical contents of metadata from outside intervention.

The first type *operation metadata* permits GIS software components to operate on geodata objects. Operation metadata bridges the connection between geodata objects and software components. For example, to interact with a specific software component such as data display, a geodata object's operation metadata should include information on appropriate display scales, coordinate projection, map units, spatial footprint, and data type (discrete, categorical or continuous). From a technical standpoint, it is currently possible to build an object wrapping these types of metadata. The more difficult technical challenge is to create software modules for displaying data that will request this information prior to display. Other examples of operation metadata include information on topologic thresholds for buffering services, or information on positional accuracy for overlay services.

The second type of metadata supports data connectivity, permits access to geodata by remote systems, and facilitates object migration. *Data connectivity metadata* identifies acceptable connection protocols (e.g. JDBC, ODBC) and preferred database engines (e.g. ORACLE, Informix, Access). Data connectivity metadata must include rules for data migration. For example, a 'copy' rule would duplicate the data object on the target node. A 'move' rule would duplicate the data object on the target node and delete the original object on the local node. A 'lifecycle' rule would dictate how long a migrated geodata object can reside at a target node.

The combination of XML and object-oriented databases can provide a possible implementation framework for metadata as described above. For example, a [San Diego Roads] data object might use XML elements to specify its metadata elements as follows:

Operation metadata:

- Coordinate system: State Plane Coordinate California Zone VI
- Map Units: feet
- Data type: categorical attributes, line features
- Topology: Built and Cleaned
- Scale threshold for display: 1/1,000,000–1/300,000
- Positional Accuracy: 0.534 RMS

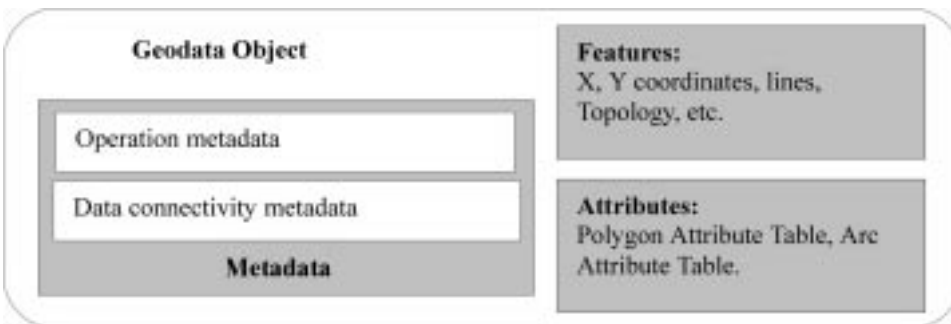

Figure 7 The content of encapsulated metadata for geodata objects

Data Connectivity metadata:

- API-type: OLEDB
- Protocol: Socket based TCP/IP
- Default database: ORACLE 8i
- Data Migration rule: copy
- Lifecycle on Target Node: 1 week

With support of operation and connectivity metadata, distributed geodata objects will become more accessible, self-describing and self-managing. Distributed GIS nodes would be able to handle geodata objects more automatically and more efficiently. To reiterate, this type of metadata scheme is dependent upon creation of software components that request information from data objects prior to processing. At present, GIS software components request this information from the user, or based upon existing system defaults.

4.3.2 Metadata for GIS Software Components

GIS software components must self-describe to interact dynamically with geodata objects, and with other software components. Component interfaces must be described to manage software connectivity with remote nodes. Figure 8 shows three metadata elements that are encapsulated to accomplish these dynamic exchanges.

Operation requirements metadata specifies input, output and data model(s) that a software object can interact with. For example, a buffer service will operate on a line with topology by generating a single envelope. Without topology, the buffer will generate a circle around each individual coordinate pair. A vector buffer service cannot operate on a raster data object. The buffer service must check for compatibility, by accessing the geodata object's operation metadata, and by publishing its own requirements metadata. If metadata are not available, the buffer service will fail.

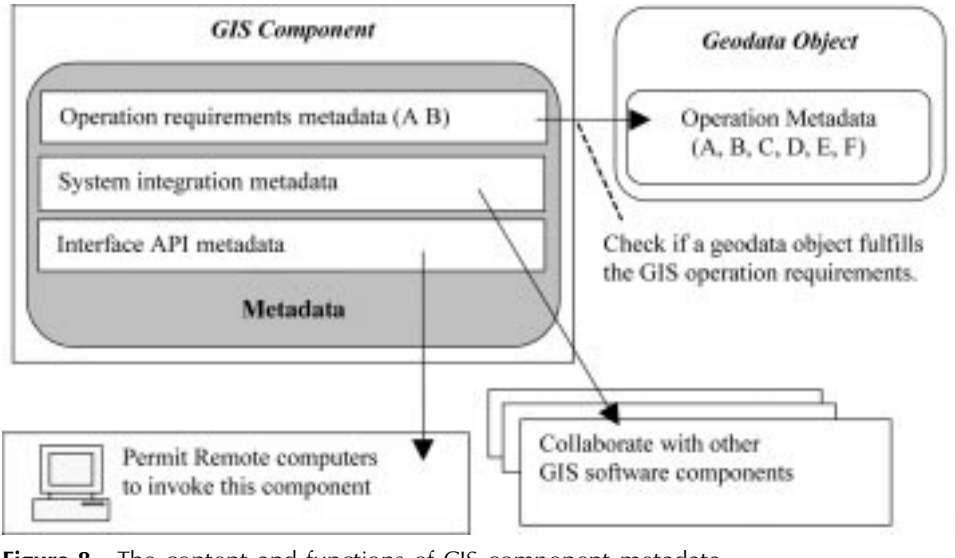

Figure 8 The content and functions of GIS component metadata

A second metadata element is *system integration metadata*, which describes available functions, methods, and behaviors for collaborating with other GIS software components. Taking the hydrological modeling example once again, flow accumulation is often computed. The task cannot be accomplished without first computing flow direction. Hypothetically speaking, a dynamic architecture for distributed hydrologic modeling would require that the flow accumulation service collaborate with the flow direction service. Again, exchange of metadata will determine compatibility of data model (TIN or raster). Metadata must be exchanged to query the data volume (number of TIN nodes, or raster size), and so forth. System integration metadata must also include component migration methods. Currently, distributed component frameworks such as CORBA and DCOM already include system integration metadata in their design. As yet, GIS software suites do not incorporate system integration checks, likely because the code relies on consistency at creation time, not at run time. Changing to the latter form of coding requires much more attention by software engineers, thus this presents not only a technical but a cultural challenge.

Just as software must interact dynamically with data objects and with other software components, dynamic exchange of metadata is required for a software object to operate on other GIS nodes. The third metadata element *interface API metadata* describes the Application Programming Interfaces (APIs). As described above, APIs support remote connectivity in distributed environments. Here, metadata on APIs is exchanged to check that the code encapsulated in a distributed software component will run on a remote node, whether a component can be migrated, or launched remotely.

One possible implementation framework for GIS component metadata is to adopt the Web Service Definition Language (WSDL) as mentioned before. WSDL can be used to define all three types of software metadata. For example, a [Map display] service's component metadata might be described as follows:

Operation requirements metadata:
- Topology: optional
- Map Units: required
- Coordinate system: required
- Positional Accuracy: optional

System Integration metadata:
- Available functions: Zoom-in, Zoom-out, Pan, Redraw
- Disk requirement: 5MB
- Display requirement: Color
- Component migration method: Java RMI

Interface API metadata:
- API type: Java 2D API
- API Protocol: SOAP
- API Runtime environment: Java Virtual Machine.

With specification of GIS operations, system requirements, interface connection and database connectivity, distributed GIS components can become reusable, modularized, self-describing, and self-managing. In spite of the existing technical impediments, we view the use of embedded metadata as key to interoperability and modularity for a distributed and dynamic GIServices architecture.

4.4 An Agent-based Communication Mechanism

Within a traditional GISystems environment, a centralized system relies on previously established data models and command syntax. A distributed environment, on the other hand, is heterogeneous. Objects may be based on various data models, varied program syntax, and a range of component frameworks. Heterogeneity is bound to intensify with the increasingly complicated nature of GIS tasks. An agent-based communication mechanism will help to automate searching, locating, and binding data objects and software components across networks. An agent is an autonomous computer program with specific functions that respond to specific events. Agent communication tends to reduce user work and information overload (Maes 1994). Essentially, agents form the mechanism by which GIS nodes interact. Agent activities essential to distributing GIServices dynamically include filtering information, interpreting information, and making decisions (Knapik and Johnson 1998, Tsou and Buttenfield 1998a).

A filter agent helps at a minimum to locate requested information and to filter out elements that are unnecessary to complete a specified task. A filter agent can play a more active role. If a specific task cannot be fully accomplished, the agent may suggest modifications or provide an alternate task completion strategy.

An interpreter agent conveys information from one node to another. In distributed network environments, heterogeneous data models and systems cannot communicate directly. An agent can bridge heterogeneous information 'islands', translating different data types and models. To translate these correctly, the agent has to acquire knowledge and methods about translation procedures. The methods are encapsulated in the metadata.

A decision agent is more complicated than the other two types, and makes choices autonomously based on encapsulated knowledge and rules. The agent collects information about specific events in a network environment, such as migration and connectivity, and then decides which of several actions to invoke. In the case of migration versus connectivity, the decision would be based upon factors such as the network throughput, GIS node processing speed, data volume, and nature of the GIS service being requested. A decision agent may analyze information about a specific event (such as a user request or a disk-full situation), and make a decision in collaboration with a user or with other agents. For example, a decision agent might detect whether a specific server is shutdown and connect a GIS node to another available server. In some cases, the agent decision is based on rules defined by users or other agents. Decision rules therefore require appropriate interfaces between users and agents. Collaborations among agents require formalized agent interaction mechanisms, and a well-defined hierarchy to resolve communication conflicts.

Two implementation approaches can be adopted to implement agent-based communication. The first approach defines three different agent types (filter, interpreter, and decision maker) that play a permanent role in the network environment. The alternative approach is more difficult to implement, and grants each agent multiple roles in their runtime cycle. For example, a single agent can act as a filter in one situation and an interpreter in another. Dynamic agent roles improve flexibility but are more difficult to code and manage. Dynamic agents are implemented as objects with polymorphism that allows an object to display multiple behaviors. With polymorphism, a single agent can interact differently with other agents, and produce different but appropriate outcomes.

How to design an appropriate communication protocol presents another challenge for the design of software agents. Agent communication protocol differs from traditional network communication protocol, such as Transport Control Protocol (TCP) and Internet Protocol (HTTP) (Peterson and Davie 1996). Agent communication protocol supports high-level, application-oriented communications that include the exchange of knowledge bases, user-defined rules, control of agent behaviors, and interactions between agents and systems (FIPA 1998). One agent communication protocol is developed by adopting the Knowledge Query and Manipulation Language (KQML). KQML is a language and protocol developed by the ARPA Knowledge Sharing Effort, developing techniques and methodology for building large-scale knowledge bases that are sharable and reusable (Bradshaw 1997, Weiss 1999).

Currently, several agent systems have been proposed or are under development in the computer industry and academic departments, such as Telescript by General Magic, Tacoma at Cornell University, Agent Tcl at Dartmouth College, Aglets by IBM, Voyager by ObjectSpace, Concordia by Mitsubishi Electric, and Ajanta at the University of Minnesota (Weiss 1999). In the cartography community, the AGENT (Automated Generalization New Technology) project developed by Lamy and Weibel (Lamy et al. 1999) demonstrates the great potential of agent based methodologies in providing solutions in autonomous map generalization.

With the help of an encapsulated metadata scheme, network communication strategies, and LEGO-like data and software components, a dynamic architecture for GIServices can be built on-the-fly when users request GIS tasks. Software agents collaborate as follows among several GIS nodes, which form the basic processing units in a distributed and dynamic GIService environment.

5 Implementing Distributed GIServices

To provide truly dynamic GIServices, the implementation has to abandon traditional concepts associated with a client/server architecture, and adopt a GIS node architecture, as described at the beginning of this paper. To collaborate, GIS nodes require network access capability, and four containers (Figure 9). All GIServices and tasks can be accomplished through collaboration between GIS nodes using these elements.

The *hardware profiles* container stores all hardware and operating system information about the GIS node. This information is automatically retrieved from the operating system and updated whenever the hardware and peripherals are modified. The hardware profile container also includes metadata about network performance and node-to-node network connectivity (in real time). Machine agents (described below) use hardware profiles as one criterion for deploying services.

GIS component containers store software modules. Several commercial products such as ActiveX and the Java Virtual Machine currently implement software containers. In general, GIS component containers integrate software modules either locally (using 'plug-and-play' mechanisms) or remotely (using remote invocation). Agents use these containers to download or distribute software modules. GIS component containers provide universal, virtual environments for GIS components to launch on different types of GIS platforms.

Geodata object containers store geodata and associate them with different database engines. Geodata are stored in and retrieved from containers based on their

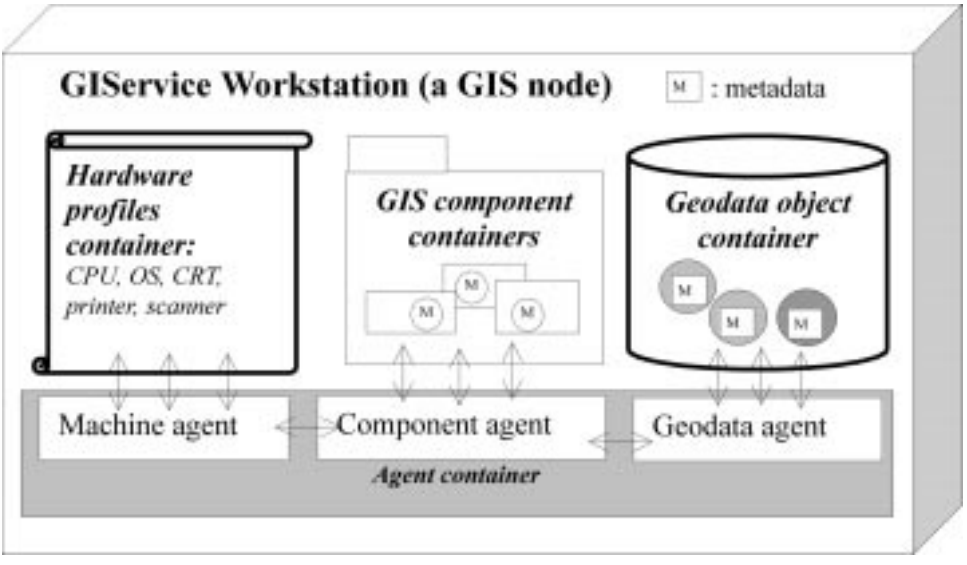

Figure 9 A GIS node within a distributed GIServices framework

encapsulated metadata. Geodata object containers also serve as packaging ('wrappers') to migrate geodata from one node to another.

Agent containers hold different types of agents including mobile agents transferred from another GIS node. Agent containers facilitate communication between agents and between other containers in a GIS node. As shown in Figure 9, machine agents, component agents and geodata agents are stored in the container and communicate with each other.

With these four containers, each GIS node becomes an independent GIS-processing unit, able to perform a complete GIS task, to respond to requests from other nodes, and to initiate GIService requests. GIS nodes broaden the capability of an isolated system into a group-based, collaborative network. Figure 10 illustrates a conceptual design of three possible collaborations among GIS nodes, across a local area network, across an Intranet, and Internet-wide.

Interaction with GIS nodes in the local area network brings benefit from proprietary data sharing and integration inside a secure local environment, such as an office building or a department. Collaboration can extend to the Intranet to share data and software modules within a company or a university. Internet-wide collaboration can distribute GIServices nationally and globally. For example, a GIS modeling task can be distributed from a GIS node in the Geography Department at the University of Colorado to request data and software from other universities, federal agencies and commercial organizations. At present, users must initiate collaborations one at a time, in preparation for initiating a GIS task. In a dynamic architecture, the act of requesting a GIService will initiate collaboration. With distributed collaboration, the learning curve for novice users may be damped down; advanced GIS users can tackle more complicated research and analysis. With a scalable dynamic architecture and independent processing units, all GIS users can accomplish tasks more efficiently and effectively.

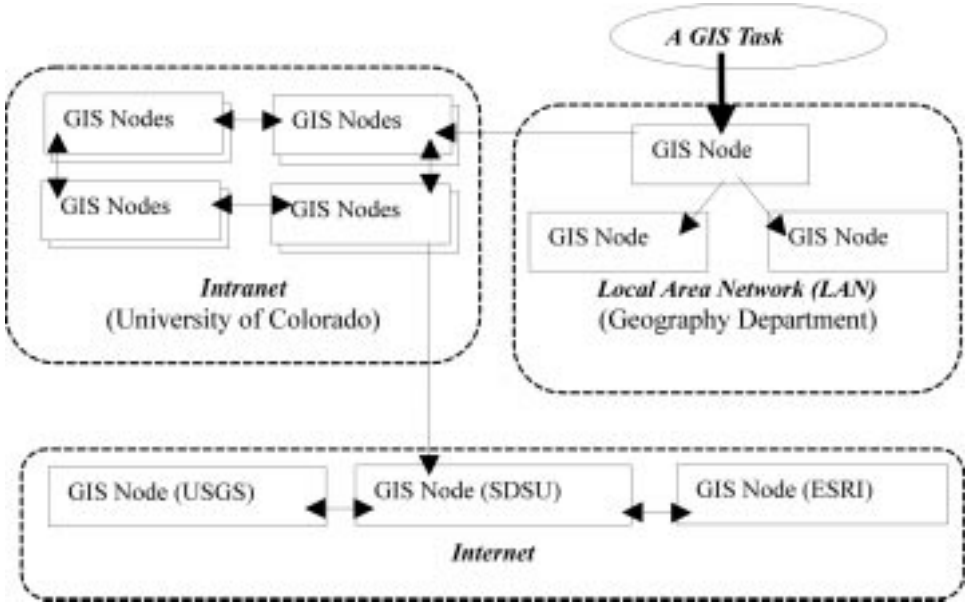

Figure 10 Three possible collaborations among distributed GIS nodes

To illustrate the potential of GIS nodes and software agents, the following user scenario illustrates how collaborations between GIS nodes and software agents could benefit GIS applications. This scenario emphasizes distributed GIS procedures, automatic data conversion, and the efficient use of computing resources.

Scenario Description:
A GIS spatial analyst, Dick, wants to locate a new Wal-Mart store in Boulder. He needs to obtain related map information and perform a GIS overlay analysis for this task. The following criteria must guide the Wal-Mart site selection:

1. The land use must be commercial urban.
2. The site must lie above the 500 year flood plain.
3. The site must be located within 200 meters of a major road.
4. The neighborhood within 1 mile of candidate sites should have appropriate demographic characteristics: annual salary $>$\$50,000, median age $<$40 and population density $>$1,000 people per square mile.
5. The optimal site fulfills criteria 1–4 and has the lowest land value.

Based on this scenario, Dick will need the following data for his GIS analysis: Land use, Flood zones, Roads, block level Census data, and Land values and parcel records.

The site selection criteria must be converted into specific tasks in order to direct the behaviors of software agents and the collection of geodata. The following list is the task request submitted by Dick based on this GIS scenario.

Task Request (submitted by Dick):

1. Create geodata land-use-select: Select all land use = 'commercial urban'.

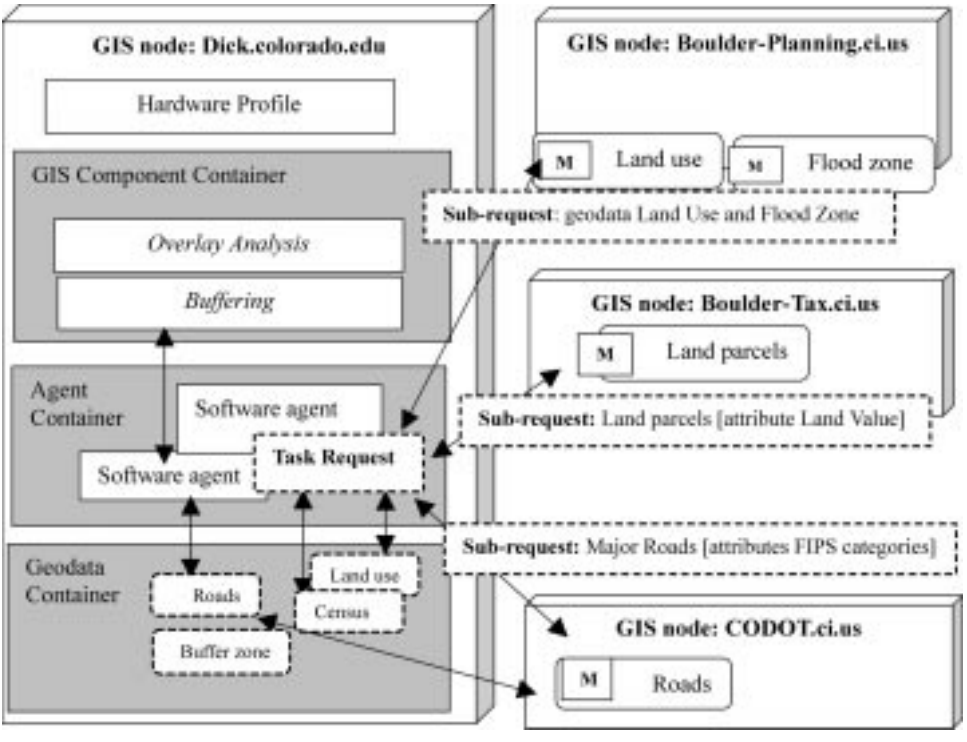

Figure 11 A Wal-Mart site selection scenario

2. Create geodata flood-polys500: Select flood plain polygons = '500 year flood plain'. (A raster alternative is possible).
3. Create geodata road-buffer200: Buffer Roads to 200 meters.
4. Create geodata Landscape: Overlay land-use-select, flood-polys, and road-buffer200.
5. Create geodata Candidate-sites: Clip Census block polygons by Landscape (retain whole blocks).
6. Create geodata Refine Sites: Query Candidate-Sites = Annual salary > $50,000 AND median age < 40 AND
7. Population density > 1,000 people per square mile
8. Create geodata Final-sites: Spatial Query Refine Sites by Land Parcel Value.
9. Query Final Sites = Parcel Value [MINIMIZE].

Software agents will help Dick to collect geodata. Dick sends his Task Request (Figure 11) to the software agent container residing on his GIS node (his workstation). A software agent encapsulates sub-requests to data servers specified in the GIS node service directory. Some sub-requests will involve additional agents, but for simplicity we will deal with only a single agent for this scenario. One sub-request (to query census demography within a 1-mile buffer of each proposed site) requires Census data that is resident on Dick's node. It is logical that an agent will check the local node prior to broadcasting a request; thus criteria (or sub-requests) may achieve responses in a different order than Dick specifies. Other sub-requests require the agent to request geodata across the network, for example to migrate Road geodata from the Colorado

Department of Transportation's data clearinghouse. The agent monitors an event loop and updates status as geodata for each sub-request achieves a response.

The Task Request involves a series of GIS operations, including overlay and buffering. The procedure is the same as for geodata collection: the agent encapsulates sub-requests to do the processing for each criterion, checking the local node before broadcasting network requests to migrate or copy software components. In this scenario, both procedures are available on Dick's node. As for the geodata sub-requests, the agent monitors an event loop and updates status as each sub-request achieves a response. As each geodata set becomes available, the buffering component (or overlay component) and geodata will negotiate by exchanging encapsulated metadata to determine compatibility with the dataset and the hardware requirements. It is possible that the agent encapsulate the compatibility negotiations when broadcasting sub-requests, to avoid downloading inappropriate data. Pre-processing analysis forms another type of negotiation. For example, each geodata set will need to be clipped; and this forms one type of overlay analysis.

By applying the software agents, Dick will not need to worry about the local system resources, actual data implementation or other mechanics underlying the task. He will focus on translating the model into a workable Task Request, and on establishing suitability thresholds (as for example median age). This is of course a very simple scenario, and much is left unaccounted for, as for example availability of appropriate data, a local GIS node with full processing resources, a stable network, the sequencing of GIS buffers and overlays to generate the final suitability model, the syntax of the task request, etc. Many of these relate to barriers and impediments discussed throughout this paper. But the scenario demonstrates how the three pieces of a dynamic architecture could interlock to provide LEGO-style GIS processing in a dynamic and distributed environment.

6 Discussion and Conclusions

The adoption of distributed component technologies, agent-based communication, and encapsulated metadata, leads to a dynamic GIService architecture able to provide flexible, dynamic, and comprehensive GIServices. Implementation will force the envelope on technical and institutional constraints, such as vendor-dependency, complex software specification and implementation of flexible standards, to insure compatibility across different component frameworks and heterogeneous databases. To deploy a dynamic GIService architecture, the GIS community must work together to confront these limitations. The following discussion illustrates the major impediments to adopting dynamic GIService architecture, from various stakeholder perspectives.

The system developer's perspective alludes to many continuing technical impediments. First, selecting the right component technology for distributed GIServices is extremely difficult. The selected technology must provide a robust, secure, and efficient communication mechanism via the Internet/Intranet. Security and stability will remain major considerations for distributed GIServices, because many geospatial datasets and services are valuable and critical. Network vulnerability can cause serious problems for GIServices, due to viruses, hackers, and network traffic jams. Inevitably, distributed GIServices will face security problems because a GIS node cannot effectively

monitor distributed objects shared across the Internet. An associated issue involves the challenge to integrate distributed solutions with legacy systems. Many valuable GIS datasets and programs reside in legacy systems that provide essential services for the scientific and lay public. Current distributed component technologies provide certain approaches to integrate legacy systems, such as 'object-wrapping' and middleware solutions. However, these approaches may reduce the performance of legacy systems, or simply prove incompatible.

Second, customizing existing technologies for distributed GIServices forms a major implementation challenge. Since distributed component technologies and web services are not designed specifically for GIServices, many requirements of GIServices are not taken into account. For example, existing technologies do not account for the complexity of geodata models and functions, or for the huge volume of geospatial databases and digital image archives. More important perhaps are the special characteristics of geospatial information, including scale-dependent geometry and content, spatial dependency and spatial heterogeneity, and relationships between measurement dimensions (Yuan et al. 2001). Adding adequate GIS functions and APIs to establish a high-level functionality to accommodate these special characteristics are essential for the successful distribution of GIServices, and increase system development time and complexity incrementally.

From the perspective of vendors, integrating different component technologies is at present an expensive and presently unnecessary hindrance. Many people think that a 'superior' technology will guarantee successful adoption and popularity in the future (this is the 'Adopt it and they will come' argument). However, in many cases the computing industry does not adhere to this belief. Examples include the failure of the NeXt operating system, of OpenDoc, and of IBM's OS2 operating system. In practice, the 'best' technology does not ensure automatic acceptance. Vendor support, marketing strategies, and usability feedback can ensure or derail the future development of any technology. Thus, the choice of an appropriate distributed component technology requires consideration of technical features, implementation details, and actual experience in practice.

Inevitably, GIServices will have to tolerate heterogeneous frameworks because no distributed component technology will be optimal for all kinds of tasks. A robust technology should integrate with, and migrate into different frameworks. Currently, most distributed component technologies have proposed solutions for integrating to future technologies. However, little evidence demonstrates that these proposals will work. Software vendors are still partially or completely hesitant to integrate their technology with other vendors' because of marketing considerations. The GIS community should encourage vendors to adopt a true integration of distributed component technologies; it will not happen automatically. It is dangerous for the GIS community to just wait and see what happens in the commercial sector.

From the perspective of users, it is important to keep in mind that the goal of all the effort in revising system design is to maximize capabilities. Traditional GISystems do not provide users with flexible and dynamic services. Future GIServices should innovate functions instead of simply mimicking current capabilities. Putting traditional GISystems on-line is not equal to distributing GIServices. The GIS community will invent new tasks, such as digital libraries, distance learning, cyberspace navigating, network-based decision support systems, virtual tourism, etc. that cannot be supported by current GIS functionality. Innovative capabilities will

energize the development of distributed GIServices and provide users with better, more comprehensive tools.

To summarize, the GIS community needs to consider the advantages and disadvantages of distributed GIServices, and to acknowledge the very compelling arguments of all stakeholders prior to adoption. The following paragraphs itemize the major advantages and disadvantages of dynamic, distributed GIServices in comparison with traditional GISystems.

Several advantages are easily identified. Flexible integration of heterogeneous data models and software components would mean working with no bigger GIS software suite than is appropriate for a task. The reduced requirements for local computing resources would additionally speed implementation of mobile GIS computing as well as facilitate adoption across both sides of the Digital Divide. Modular, customizable services based on reusable programming code can reduce development costs for new services, and move access to GIServices to a more equitable status. Intelligent management of geospatial information through encapsulated metadata should by itself promote easier updates, help to monitor uncertainty levels and thus improve usability. Self-broadcasting metadata can reduce risks in cases where decisions based on GIServices must be made quickly.

Several disadvantages must be respected, regardless of one's enthusiasm about distributed technologies. All allude to the extra costs associated with distributed communication (Schroeder 1993). These include node failures, service failures, possibility of unreliable communication or disconnected network links, over which a user or developer may have little control. Complicated software architectures will be more difficult to design, to integrate, and to maintain with changing technology, and this has implications for system administrators as well. The issues associated with security apply to systems, services, and data equally.

In the opening keynote address of the Fourteenth Symposium on Reliable Distributed Systems, Malek (1995, p. xii) alluded to the uncertainties associated with moving to open architectures. His statements apply especially as our community considers adopting the new, distributed GIS paradigm. He said:

- Past: Mainframes = Sanity and Order;
- Present: Open Systems = Insanity & Chaos;
- Future: Closed & Open Systems = Peaceful Coexistence

In the past, traditional GIS implied 'sanity and order' because they reside on closed systems and architectures. As the development of GIServices begins to shift into a distributed paradigm, things become chaotic for all the reasons discussed above. With the support of industrial standards and the progress of network technologies, GIS users hope to utilize distributed GIServices from both their standalone PCs or Workstation and network-based distributed GIS nodes, in an easy and friendly way. Developers and vendors face a daunting task of having to re-think how GIServices are delivered, and through what channels. Coordination and collaboration present a cultural barrier to distributed GIServices that may be far more obstructive than the technical engineering aspects.

The truth of the matter is that a fully open GIServices architecture may not survive in a free market economy without considering economic incentives, and furthermore will not be adopted until the GIS community identifies collaborative as opposed to competitive solutions to institutional and economic barriers, regardless of what

information technologies emerge. This paints a potentially dark canvas, but it need not remain so. Ultimately, distributed GIServices should help all stakeholders in the GIS community to communicate, interact, and learn from each other. It is hoped that these barriers and impediments will be explored and answered by the efforts of spatial scientists, information scientists, and computer scientists in the future.

Acknowledgements

This paper forms a portion of the Web-based GIServices project, sponsored by the NASA ARC project. Funding by the NASA ARC Program is greatly appreciated. Matching funds from the San Diego State University Foundation and from the University of Colorado are also acknowledged. We thank the anonymous reviewers whose comments helped to sharpen the focus of our discussion.

References

Aronoff S 1989 *Geographic Information Systems: A Management Perspective*. Ottawa, WDL Publications

Baldonado M, Chang C K, Gravano L, and Paepcke A 1997 The Stanford Digital Library Metadata Architecture. *International Journal on Digital Libraries* 1: 108–21

Bishr-Yaser M S 1996 A Mechanism for object identification and transfer in a heterogeneous distributed GIS. In *Proceedings of the Seventh International Symposium on Spatial Data Handling*, August 12–16, Delft, The Netherlands: A.1–A.13

Bradshaw J M (ed) 1997 *Software Agent*. Menlo Park, CA, AAAI Press

Brûlé M R 2002 Deciphering .NET for GIS. *Geospatial Solutions* 12(6): 32–7

Buehler K and McKee L (eds) 1996 *The Open GIStm Guide: Introduction to Interoperable Geoprocessing*, Wayland, MA, Open GIS Consortium Inc

Buehler K and McKee L (eds) 1998 *The Open GIStm Guide: Introduction to Interoperable Geoprocessing and the OpenGIS Specification* (Third Edition). Wayland, MA, Open GIS Consortium Inc

Buttenfield B P 1997 The future of the spatial data infrastructure: Delivering geospatial data. *GeoInfo Systems* 7: 18–21

Buttenfield B P 1998 Looking forward: Geographic information services and libraries in the future. *Cartography and Geographic Information Systems* 25: 161–71

Buttenfield B P and Goodchild M F 1996 The Alexandria Digital Library Project: Distributed library services for spatially referenced data. In *Proceedings of GIS/LIS '96*, 19–21 November, Denver, Colorado: 76–84

Buttenfield B P and Tsou M H 1999 Distributing an internet-based GIS to remote college classrooms. In *Proceedings of the Nineteenth Annual ESRI International Users Conference*, San Diego, California: CD-ROM

Cauldwell P, Chawla R, Chopra V, Damschen G, Dix C, Hong T, Norton F, Ogbuji U, Olander G, Rchman M A, Saunders K, and Zaev Z 2001 *Professional XML Web Serviecs*. Birmingham, Wrox Press

Craig W J 1998 The internet aids community participation in the planning process. *Computers, Environment and Urban Systems* 22: 393–404

Federal Geographic Data Committee (FGDC) 1995 *Content Standards for Digital Geospatial Metadata Workbook* (Version 1.0). Reston, VA, Federal Geographic Data Committee

Foundation for Intelligent Physical Agents (FIPA) 1998 The FIPA 97 Specification: Part 2, Agent Communication Language (Version 2.0). WWW document, http://www.fipa.org/spec/fipa97.html

Frew J, Freitas N, Hill L, Lovette K, Nideffer R, and Zheng Q 1998 The Alexandria Digital

Library System Architecture. WWW document, http://www.sbg.ac.at/geo/eogeo/authors/frew/frew.htm

Ganti N and Brayman W 1995 *The Transition of Legacy Systems to a Distributed Architecture*. New York, John Wiley and Sons

Goodchild M F 1997 Towards a geography of geographic information in a digital world. *Computers, Environment and Urban Systems* 21: 377–91

Goodchild M F 2000 Communicating geographic information in a digital age. *Annals of the Association of American Geographers* 90: 344–55

Gosling J and McGilton H 1996 The Java Language Environment. WWW document, http://www.Java.sun.com/docs/white/langenv/

Huang B and Worboys M F 2001 Dynamic modeling and visualization on the Internet. *Transactions in GIS* 5: 131–9

Huse S M 1995 GRASSLinks: A New Model for Spatial Information Access in Environmental Planning. Unpublished PhD Dissertation, University of California at Berkeley

ISO/TC 211/WG 1 1998 *Geographic Information: Part 2, Overview*. Geneva, International Organization for Standardization Report No ISO/TC 211-N541, ISO/CD 15046-2

ISO/TC 211/WG 3 1998 *Geographic Information: Part 15, Metadata*. Geneva, International Organization for Standardization Report No ISO/TC 211-N538, ISO/CD 15046-15

Knapik M and Johnson J 1998 *Developing Intelligent Agents for Distributed Systems: Exploring Architecture, Technologies and Applications*. New York, McGraw-Hill

Kraak M-J and Brown A 2001 *Web Cartography*. London, Taylor and Francis

Kuhn W 1997 *Toward Implemented Geoprocessing Standards: Converging Standardization Tracks for ISO/TC 211 and OGC*. Geneva, International Organization for Standardization Report No ISO/TC 211-N418

Lamy S, Ruas A, Demazeu Y, Jackson M, Mackaness W, and Weibel R 1999 The Application of Agents in Automated Map Generalisation. WWW document, http://agent.ign.fr/public/ica/index.html

Li B 1996 Issues in designing distributed geographic information systems. In *Proceedings of GIS/LIS '96*, 19–21 November, Denver, Colorado: 1275–84

Li B and Zhang L 1997 A model of component-oriented GIS. In *Proceedings of GIS/LIS '97*, 28–30 October, Cincinnati, Ohio: 523–8

Maes P 1994 Agents that reduce work and information overload. *Communications of the ACM* 37: 31–40

Malek M 1995 Omniscience, consensus, autonomy: Three tempting roads to responsiveness. In *Proceedings of the Fourteenth Symposium on Reliable Distributed Systems*, 13–15 September, Bad Neuenahr, Germany. Los Alamitos, CA, IEEE Computer Society Press: xii–xiv

Montgomery J 1997 Distributing components. *BYTE* 22(4): 93–8

National Science Foundation 1994 NSF Announces Awards for Digital Libraries Research. Washington, DC, National Science Foundation Press Release No 94–52

National Science Foundation 1998 Digital Government Program Announcement. Washington, DC, National Science Foundation Press Release No 98–121

Newcomer E 2002 *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Reading, MA, Addison Wesley

Open GIS Consortium Inc (OGC) 1998 *The OpenGIS Abstract Specification* (Version 3). WWW document, http://www.opengis.org/techno/specs.htm

Orfali R and Harkey D 1997 *Client/Server Programming with Java and CORBA*. New York, John Wiley and Sons

Orfali R, Harkey D, and Edwards J 1996 *The Essential Distributed Objects Survival Guide*. New York, John Wiley and Sons

Ostensen O 1995 Mapping the Future of Geomatics. *ISO Bulletin* (December 1995): 13–15

Petrik K 2002 Internet-Based Geographic Information Services in College Classrooms. Unpublished Masters Thesis, Department of Geography, University of Colorado

Peng Z-R and Nebert D D 1997 An Internet-based GIS data access system. *Journal of Urban and Regional Information Systems* 9: 20–30

Peterson L L and Davie B S 1996 *Computer Networks: A Systems Approach*. San Francisco, CA, Morgan Kaufmann

Peterson M 1997 Cartography and the Internet: Introduction and research agenda. *Cartographic Perspectives* 26: 3–12

Pleas M 2000 Microsoft .NET. *PC Magazine* (December 5): IP01–IP08

Plewe B 1997 *GIS Online: Information Retrieval, Mapping, and the Internet*. Santa Fe, NM, OnWord Press

Putz S 1994 Interactive Information Services Using World Wide Web Hypertext. WWW document, http://www94.web.cern.ch/WWW94/PrelimProcs.html

Roberts-Witt S 2001 Peer pressure. *PC Magazine Internet Business* (June 26): 8–16

Rowley J 1998 Draft business case for the harmonization between ISO/TC 211 and the Open GIS Consortium, Inc. Geneva, International Organization for Standardization Report No ISO/TC 211-N472

Rumbaugh J, Blaha M, Premerlani W, Eddy F, and Lorensen W 1991 *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ, Prentice-Hall

Schroeder M D 1993 A state-of-the-art distributed system: Computing with BOB. In Mullender S (ed) *Distributed Systems*. Wokingham, Addison-Wesley: 1–16

Shuey R 1989 Data engineering and information systems. In Gupta A (ed) *Integration of Information Systems: Bridging Heterogeneous Databases*. New York, IEEE: 11-23

Smith T R 1996 The Meta-Information Environment of Digital Libraries. *D-Lib Magazine* 2(7/8): http://www.dlib.org/dlib/july96/new/07smith.html

Sondheim M, Gardels K, and Buehler K 1999 GIS interoperability. In Longley P A, Goodchild M F, Maguire D J, and Rhind D W (eds) *Geographical Information Systems: Principles, Techniques, Applications and Management* (Second Edition). New York, John Wiley and Sons: 347–58

Takatsuka M and Gahegan M 2002 GeoVISTA *Studio*: A codeless visual programming environment for geoscientific data analysis and visualization. *Computers and Geosciences* 28: in press

Taylor D A 1992 *Object-Oriented Information Systems: Planning and Implementation*. New York, John Wiley and Sons

Tsou M H and Buttenfield B P 1998a An agent-based, global user interface for distributed geographic information services. In Poiker T K and Chrisman N (eds) *Proceedings of the Eighth International Symposium on Spatial Data Handling, Vancouver, Canada*. Burnaby, BC, Simon Fraser University: 603–12

Tsou M H and Buttenfield B P 1998b Client/server components and metadata objects for distributed geographic information services. In *Proceedings of the GIS/LIS '98*, 10–12 November, Fort Worth, Texas: 590–9

Vckovski A 1998 *Interoperable and Distributed Processing in GIS*. London, Taylor and Francis

Walker D, Rana O F, Li M, Shields M S, and Huang Y 2000 The software architecture of a distributed problem-solving environment. *Concurrency Practice and Experience* 12: 1455–80

Weiss G (ed) 1999 *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA, MIT Press

Wright H, Brodlie K, Wood J and Proctor J 2000 Problem solving environments: Extending the role of visualization systems. In Bode A, Ludwig T, Karl W, and Wismueller R (eds) *Euro-Par 2000 Parallel Processing*. Berlin, Springer-Verlag Lecture Notes in Computer Science No 1900: 1323–31

Yuan M, Buttenfield B P, Gahegan M, and Miller H 2001 Geospatial Data Mining and Knowledge Discovery. Unpublished White Paper, University Consortium of Geographic Information Science

Zhang L and Lin H 1996 A client/server approach to 3D modeling support system for coast change study. In *Proceedings of GIS/LIS '96*, 19–21 November, Denver, Colorado: 1265–74