Designing and Implementing the User Interface of Geographic Digital Libraries

Juliano Lopes de Oliveira
Marcos André Gonçalves
Claudia Bauzer Medeiros

Relatório Técnico IC–97-25

Dezembro de 1997

Designing and Implementing the User Interface of Geographic Digital Libraries

Juliano Lopes de Oliveira* Marcos André Gonçalves
Claudia Bauzer Medeiros†

Abstract

Geographic data are useful for a large set of applications, such as urban planning and environmental control. These data are, however, very expensive to acquire and maintain. Moreover, their use is often restricted, for lack of dissemination mechanisms. Digital libraries are a good approach for increasing data availability and therefore reducing cost, since they provide efficient storage and access to large volumes of data. Geographic applications can diminish their costs by reusing and sharing data through Geographic Digital Libraries (GDL). One major drawback to this approach is that it creates the necessity of providing facilities for a large and heterogeneous community of users to search and interact with these Geographic Libraries. We present a solution for this problem, based on a framework that allows the design and construction of customizable user interfaces for GDL applications. This framework relies on two main concepts: a Geographic User Interface Architecture and a Geographic Digital Library Model.

1 Introduction

Digital Libraries (DL) provide infrastructure for creating, structuring, storing, organizing, processing, retrieving and distributing multimedia digital information. This type of information includes texts, images, videos, audio data, and programs (games, animation, and simulation).

Hypermedia data models are an important step towards the efficient access and retrieval of these unstructured data from DL. A complete hypermedia data model must offer abstractions to describe (i) the stored information (structural model); (ii) the browse and query mechanisms (navigational model); and (iii) the user interface aspects of the applications, which will be developed using the underlying hypermedia system.

Geographic Digital Libraries (GDL) are DL which can handle *georeferenced data*. These data describe geographic entities through three components [CCH+96]: *geographic location* (the position and shape of the entity on the Earth surface); *descriptive attributes* (conventional data defining the properties and features of the entity); and *time* (date or period

*Instituto de Informática, Universidade Federal de Goiás, 74001-970 Goiânia, GO
†Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

related to the capture or to the validity of the data). Georeferenced data are inherently heterogeneous (many distinct formats and collecting procedures) and approximative (they represent a simplified model of the real world).

The standardization of GDL involves not only the organization of data, a problem inherent to DL, but also issues related to the collection and integration of these data (again, a problem of DL, but aggravated in the case of georeferenced data). Moreover, users of geographic applications have a wide range of requirements for visualization and manipulation of the data. For instance, given the heterogeneity of the users (e.g., biologists, ecologists, architects, engineers, demographers), both the vocabulary used to search for data, and the presentation format for the selected data are specific for each user profile and application.

Therefore, the user interface component must provide facilities to support requests to (1) search, browse, filter, query, and classify large sets of georeferenced data, and (2) cluster, combine, and customize these data in order to present them according to the users' profiles.

Current techniques to design and implement user interfaces are not adequate for these special requirements. Even the user interface architectures and models that are geared towards geographic applications are not able to deal with many of the problems involved. The framework presented in this paper - the Geographic User Interface Framework - aims to solve the limitations of the current approaches.

Our approach is based on combining a *Geographic Digital Library Model* and a *Geographic User Interface Architecture*, using an underlying Object-Oriented Database Management System (OODBMS) as an integration platform:

- The Geographic Digital Library Model integrates the hypermedia paradigm to a geographic data model, implemented in the OODBMS. The result is an object-oriented database which can support the requirements of hypermedia models in the context of DL (such as openness and dynamic links), for georeferenced data. This model provides two means of extracting data from a geographic DL: browsing mechanisms, inherent to hypermedia models; and metadata queries, supported by the underlying OODBMS.

- The Geographic User Interface Architecture is a framework to support the design and implementation of user interfaces for geographic applications, based on database technology. It combines concepts from three areas of Computer Science – Databases, Software Engineering and Human-Computer Interfaces – in a innovative perspective, considering interface interactions not only with the user, but also with the underlying software. The framework covers both the architecture of the interface and the mechanisms and models for its construction. The architecture can be implemented with most of the existing interface development tools, building dynamic interfaces (which can be modified at run-time). The framework relies on active mechanisms to generate customizable interfaces from reusable components.

The framework was designed to allows design and implementation of interfaces for application systems that handle georeferenced data. Parts of it have already been used to build interfaces for an urban planning system and for an environmental control system. The main advantage of the user interface developed within this environment is that it is *customizable*, i.e., it can be easily adapted to different users.

This paper shows how this framework can be adapted to support the design and the implementation of the user interface component of the geographic digital library. This interface must support distinct interaction and visualization modes according to the user profile, and user-dependent customization facilities.

The remainder of this paper is organized as follows. Section 2 surveys related research areas and identifies the main barriers for the development of user interfaces for GDL. Section 3 introduces our model for a GDL. Section 4 discusses the interface component of the library. Section 5 describes the user interface framework and shows its integration to the GDL model. Section 6 presents conclusions of this work, and discusses future work.

2 Research Issues and Related Work

The development of user interfaces for GDL involves three main research areas: Digital Libraries, Geographic Databases, and User Interfaces. We discuss, initially, the main characteristics and properties of each area, in an isolated perspective. Afterwards we analyse the three areas from an integrated point of view, identifying the requirements and the difficulties for their integration. This analysis produces a list of open problems related to the development of user interfaces for geographic digital libraries. Next, we consider open issues in interface architectures for GDL. Solutions for some of these problems are proposed in the remainder of the paper.

2.1 Digital Libraries

The term *Digital Library* is being frequently used to designate software environments which provide to their users a wide variety of heterogeneous electronic services and data [DA97].

A fundamental property of data stored in DL is their interconnection, which originates the notion of *"hypermedia documents"*. Hypermedia is the science of relationships [BI95]. Links inter and intra documents are maintained, for long periods, between (possibly remote) locations [BAN94]. Hypermedia functionality - such as navigation (including browsing and backtracking), annotation, and overview of information - are essential in the context of DL.

Database systems are another technology of fundamental importance to support DL. Many papers discuss the role of DBMS in DL, identifying features of those systems that are useful in this context [AFY94, AHN94, BAN94, GBA$^+$94], e.g., concurrency mechanisms; recovery; persistence; transaction processing; and access control. However, present DBMS do not adequately support DL requirements: for instance, they provide limited metadata queries and have difficulties in changing schemata.

2.2 Geographic Databases – GDBs

Geographic Database Management Systems (GDBMS) are the kernel of *Geographic Information Systems* (GIS), and are characterized by storing both standard and georeferenced data.

GIS are systems that allow capture, manipulation, analysis and display of georeferenced data. Applications of GIS technology vary from worldwide scale (e.g., global natural resource management) to local concerns (e.g., city planning).

The GDBMS inside a GIS manages the data stored in a *geographic database*. A *Geographic Database* (GDB) is a repository of information collected empirically about real world phenomena [Goo92]. Users georeference data in a GDB by associating the data with locations expressed in some coordinate system. Coordinate system, scale and time are among the many parameters that must be considered by a GDBMS, and which account for the problem of *representation heterogeneity*.

User interaction with GIS can be characterized by two main activities: *specification*, which corresponds to modelling and designing the underlying geographic databases; and *operation*, which involves browsing and querying data in order to derive information and build/validate models of the world.

2.3 User Interfaces and Deficiencies of Present Architectures

Interactive applications are software which exchange data and controls with the user in order to perform their functions. An interactive application has two main components: the *user interface*, which controls the dialogue with the user (through input/output mechanisms); and the *semantic kernel*, which defines the functionality and the semantics of the application (in terms of transformation and manipulation of data).

The advantages of separating the user interface and the semantic kernel components of an interactive application are consensual [HS89, BC91, NMK91, Edm92, Mye95]. All user interface architectures presented in the literature are based on the hypothesis that the two components of the application can be identified and isolated.

A user interface architecture is an abstract model for the specification of the logical and functional organization of the user interface component, as well as its communication and integration with the semantic kernel component. Many user interface architectures have been presented in the literature, and they can be grouped in two main categories:

- *Modular* user interface architectures are based on the partitioning of the functionality of the user interface component into interdependent software subsystems (modules). In general, there is one module for the dialogue control, one for the presentation control, and one for communication with the semantic kernel of the application. Seeheim [Gre85] and Slinky/Arch [BFL+92] are typical modular user interface architectures.

- *Agent based* architectures use the notion of *agent* to organize the user interface component of the application. These architectures are specially indicated for the organization of dialogue control. MVC [KP88] and PAC [BC91] are two representative examples of agent based user interface architectures.

Two main functionalities of the user interface are defined in all architectures: *presentation control*, which deals with interaction objects (*widgets*); and *dialogue control*, responsible for the syntax of the interaction with the user, and for the conversion of data between interface and semantic kernel.

The user interface performs the control of presentation and dialogue, but it relies on underlying user interface software to execute parts of the involved activities. Two main user interface support softwares can be identified: windowing systems and interface toolkits. The former interact directly with the operating system to support the division of the screen into distinct regions (*windows*) and the individual management of these regions. The latter are libraries defining interaction objects (*widgets*) which can be (re)used by the programmer for building the application's user interface component.

Present interface architectures are not adequate for geographic applications because [Oli97]:

- They do not take into account the idiosyncrasies of geographic data. For instance, generic interface architectures deal with multiple views of application's objects only in the interface level. In geographic applications, real world entities may have multiple representations in the database (e.g., an image and a geometric description of a geographic entity).

- They do not consider the geographic software organization. In particular, generic interface architectures deal only with interface supporting software (such as interface toolkits), but do not consider the integration of the application with the underlying geographic software (GIS and GDBMS).

- They are abstract models of the user interface, which are, in general, not easily mapped into implementation code.

To solve these problems, many user interface architectures have been proposed to deal with the special features of geographic applications [OM96]. Many of them, however, present only limited solutions. In the one hand we have architectures which are directed to a specific GIS architecture (e.g., [Env92, AYA+92]), therefore not useful for building generic geographic applications. On the other hand, we have the open geographic interface architectures, based on the independence from the GIS. These architectures impose two types of functional restrictions:

- Restrictions on the functions of the GIS: these architectures allow the integration of the interface only with GIS that support specific services.

- Restrictions on the functions of the geographic application. The idea is to minimize and to simplify the interaction between the geographic application and the GIS, eliminating, in general, functions with side effects on data (that is, updating functions). [Rig95] and [OM95] are examples of such geographic interface architectures.

Our interface framework solves these drawbacks. As the previous architectures, we separate the user interface from the semantic kernel of the geographic application. Unlike the existing architectures, we restrict neither the functionality of the GIS, nor the functionality of the application. Moreover, the abstract model in which our architecture relies is directly mapped to an implementation model, using common user interface development tools currently available.

2.4 Integrating User Interfaces, GDBs and Digital Libraries

User Interfaces for GDL allow users to find and manipulate data stored in geographic databases within the scope of a GDL. The fundamental question to be answered in this section is: why are GDL user interfaces more complex than conventional and GDBMS user interfaces?

2.4.1 Geographic DBMS User Interfaces

GDBMS user interfaces should provide all the desired features of generic DBMS user interfaces, and furthermore support the particularities required to manipulate georeferenced data.

A first problem is automatic display generation, very simple in conventional DBMS. In object-oriented DBMS, the problem is complex, but suitable solutions have been presented. In GDBMS, however, this task is incomparably harder, involving very complex steps (for instance, transformation of arbitrary alphanumeric data into graphic format, or cartographic production).

Generalization, which is an open research area, is unavoidable in user interfaces offering different levels of visualization. There are two main types of generalization [LR93]: *abstract*, which deals with schema manipulation to provide different views of the database, and *cartographic*, which manipulates geometric objects and symbols to improve the readability of spatial data presentations. In DBMS, only abstract generalization is involved; in GDBMS, cartographic generalization contributes to the complexity of the problem.

The volume of data manipulated in GIS is usually very high and the interface has to provide large buffers to temporally store and manipulate the data retrieved from the GDB. Efficient management of buffers is thus a typical DBMS problem that the GDBMS user interface must deal with. In conventional DBMS user interfaces this problem does not arise.

Finally, the conceptual model of the underlying GDBMS cannot be used as the representation model of the interface (contrasting, for instance, with relational DBMS interfaces). Data modeling in GDBMS involves spatial concepts usually expressed as low level data structures, such as lists of coordinates, which are not adequate for human spatial cognition. Graphical representations are therefore used as an intermediate model, and the burden of translation between models is undertaken by the user interface system.

There are many other difficulties for developing GDBMS user interfaces. We classify them according to three main areas: *architecture*, *language*, and *human factors*. Figure 1 relates the three areas in a diagram [OM96]. Papers dealing with architectures for GDBMS and GIS interfaces try to optimize the communication between the database and the interface system, through a internal language L_i. The main goal is to build a representation model at the interface level that can be mapped to and from spatial data models in an efficient way.

In general, research at the architecture level does not completely describe an external language L_e. This language is used to convey the interaction of the user with the representation model of the interface, and it is sufficiently complex to be treated as a separate field of research. The objective is to provide efficient mappings between the user mental model and the interface representation model, with emphasis on the latter model.

Efforts in the area of human factors share the objectives of those on external languages. The approach, however, is much more abstract. The main concern is the definition of a

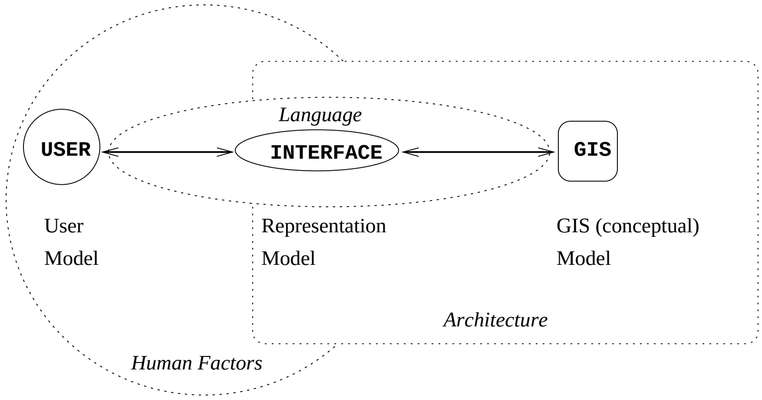

Figure 1: Research Areas in GDB User Interfaces

mental model of the user, which is mapped to the representation model, and vice-versa. The target of the research is to understand how the user thinks and, based on this knowledge, to develop the appropriate representation model in the interface.

As shall be seen, our solution is based on an architectural approach, where the representation model contains abstractions which are close to the mental model of GDL' users.

2.4.2 Geographic Digital Library User Interfaces

The user interface of a GDL must provide, at least, the following facilities: (1) query by content; (2) hypermedia navigation; and (3) georeferenced data presentation.

User interfaces for GDL have to solve all the problems mentioned in the previous section, since GDB are at the heart of GDL. In addition, GDL have particular requirements for the user interface, which are not present in GDBMS interfaces. The following discussion identifies some of these requirements, which consider a GDL implemented as a core hypermedia application of a GDBMS.

Navigational Facilities Browsing in the GDL is supported by following hypermedia links. This type of interaction must be combined with queries to the underlying GDBMS.

One problem that is common in this context is that the user gets “lost”, i.e., the user cannot keep track of the many paths followed in the net of related data. For georeferenced data, a common type of link semantics is that associated with the geographic location paradigm. Thus, a different type of link construction and maintenance should be supported.

Geographic data present the additional problem of handling multiple representations, and the fact that relationships among data may be derived from their location. This, in turn, presents new challenges to their presentation.

Customization Given the diversity of tasks and the heterogeneity of GDL users' profiles and applications, one can conclude that no single static user interface model is powerful enough in GDL. It is, thus, necessary to develop personalized user interfaces, or alternatively, interfaces that can be adapted to the preferences of users, allowing different behaviors in different contexts.

Knowledge of the Underlying Data Model The *table metaphor* is sufficient to express the concepts and operations in user interfaces for relational DBMS. The same universal metaphor does not exist in GDL, due to the semi-structured nature of geographic multimedia data, and also to the lack of organization of hypermedia environments, such as the WWW.

In cases where there exists an underlying model, the interface is responsible for offering the user access to the description of the model, to let him/her know the attributes and data types that are available, for instance. The DLITE interface [Cou96, CPW+97], developed for the Stanford DL [BCGP97a], offers to its users these kind of facilities. This interface is based on a *work centers* metaphor, in which each work center aggregates tools for a specific work (searching and publishing, for instance).

Even though there are several proposals for geographic data models, no consensus has been reached. Futhermore, end-users have two diametral concepts of geographic reality (the field view *versus* the object view) depending on the kind of application domain. Most models support just one of the views, which is reflected on the interface and hampers system usability.

Classification, Analysis and Clustering Systems User interfaces for GDL are typically general purpose tools for exploring geographic information. In this context, there is neither a limited number of possible information classification types, nor a well defined set of user information necessities and requirements [BW97]. Therefore, a dynamic set of classification and clustering mechanisms is mandatory in these user interfaces. *SenseMaker* is an example of a DL user interface using this approach [BW97, BCGP97b]. It helps users to visualize and examine collections of query results, and to navigate among such collections. The results are organized and clustered according to user defined criteria (e.g., URLs referencing the same site).

On the other hand, a user interface that embeds a predefined classification structure provides to its users a generic way for searching, and it is able to help the user on this process [All95]. A predefined classification schema is also important for the refinement of queries (restricting its computational cost) and to avoid overloading the user with spurious information. The user interface of GDL should define a trade-off solution between these two conflicting approaches for organization and access to geographic data.

2.5 A Brief Comparison of User Interfaces for GDL

All geographic digital libraries provide facilities for interacting with collections of multimedia georeferenced data, based on time and space search criteria. We summarize here three important features of current GDL user interfaces, comparing the available facilities with the proposals of this paper:

- Smooth integration of browsing and querying activities

Current GDL are based on, and limited by, the WWW technology. There are three main drawbacks to this technology, concerning the integration of browsing and querying processes. First, the stateless property of the HTTP protocol does not keep track of the performed transactions (e.g., Alexandria [ALD+95, FCF+95] and Berkeley [Wil96] GDL). Second, query services (responsible for the definition of spatio-temporal queries) are usually delegated to external *plug-in* applications (e.g., Geoscope Project [SCB95]) or to HTML forms. Only recently Java technology started to be used, but already having several limitations on the user interaction. Third, WWW is based on a filesystem paradigm and does not have any schema notion.

As we shall be seen, our GDL interface allows the activation of query services which are embedded in the underlying GDBMS. This and the browsing process allow the simultaneous visualization of data and metadata, helping the users to understand and to redefine the query scope. The GDBMS maintains information about executed queries, which can define the user navigation space, supports interleaving queries, browsing and filtration actions, the latter supported by a database view mechanism.

- Full exploration of hypermedia functionality

  From a database point of view, our GDL is a hypermedia geographic application on top of a OODBMS. It takes advantage of a data model which combines hypermedia and geographic concepts, offering to the users different navigation paths and paradigms. Futhermore, the presentation of data through interfaces customized for specific contexts helps the user in not getting lost in the hypermedia navigation space.

  This approach contrasts with current GDL interfaces, which use hyperlinks only to show images or maps associated with query results through HTML tables (e.g., Alexandria GDL).

- Trade-off solution between Static and Dynamic Classification Systems

  Most GDL provide some kind of thematic categorization or hierarchical classification of metadata to help users on searching georeferenced data (e.g., the Metadata Browser of the Geoscope Project and the user interface of the Berkeley GDL).

  Our GDL architecture also provides these facilities. The distinctive feature in our interface is that the hierarchical navigation contexts may be dynamically defined via users' queries or via navigation contexts. For instance, if the user navigates from a collection of images organized by theme to a collection of versions of one of those images, the contents of the second collection are dynamically determined.

3 Geographic Digital Library Architecture

This section describes the architecture proposed for a GDL. This architecture is based on combining a GDB with concepts of hypermedia management. In order to do this, it uses the following building blocks:

- a hypermedia geographic data model, which combines and extends features from different hypermedia models (OOHDM [SRB96] and Extended Dexter Model [GT96]) and a geographic object-oriented data model (GMOD [Pir97]).
- a methodology for building GDL, which we treat as geographic hypermedia applications. The methodology extends the OOHDM proposal [Ros96], a metodology for creating hypermedia applications based on object-oriented abstractions. OOHDM is well suited for DL, since it defines three steps for the creation of an application: conceptual modelling, navigational modelling, and user interface modelling.
- an OODBMS to provide storage management.

The GDL is specified using the methodology, following the three steps mentioned. Conceptual Modeling, using the geographic dimension of the data model; Navigational Modeling, which uses the hypermedia dimension of the data model, modifies the conceptual model in order to provide links and user work contexts; and Interface Modeling, which specifies the user interface, by means of special diagrams, which allow customization. The specification is then mapped to the OODBMS. This section describes the main features of this architecture, concentrating on aspects relevants to user interaction and interface. The interested reader should consult [Gon97] for details.

3.1 The Hypermedia Geographic Data Model

The hypermedia geographic data model of the GDL combines hypermedia concepts to a extension of the GMOD object-oriented data model.

On the hypermedia side, it is assumed that users work within *hypermedia contexts*. A context is a specific organization imposed on the underlying data, in terms of context nodes, context links, and display look-and-feel. Contexts can be dynamically created, using database views, as we will describe later.

Nodes and links are semantic units for navigation within or across contexts. Navigation may be either dependent or independent of context. The former navigation is based on fixed and predefined anchors, while the latter is based on dynamic selections or on queries.

From an interface point of view, the most important issue is that the user handles *context nodes*, which are triples $< node, context - info$, ADV>, where *node* is a hypermedia node; *context-info* is data which describes how the node should be considered in the context (e.g., navigation direction); and ADV defines the presentation of node in the context.

The georeferenced dimension of the GDL data model is provided by GMOD, which is an object-oriented model that:

- separates the conceptual and the representation specifications of georeferenced data. In traditional design, the question of deciding how to represent the properties of an element defined at the conceptual level is too simple to deserve separate discussion. By contrast, the representation of the spatial properties of geographic elements involves questions that deserve careful attention – e.g., scale, precision, cartographic projection.

- contemplates three modeling concepts: object-oriented classes; relationships, which allow connecting these classes in several ways; and constraints, which are imposed on classes, relationships, and their instances. Classes and relationships may be temporal or atemporal, according to whether or not their instances are allowed to vary with time. Besides relationships found in other models, GMOD also considers causal relationships, which allow associating classes in a cause-effect link, thereby enabling modeling the dynamics of geographic phenomena.

- provides field and object views of the world, which allow modeling, respectively, continuos phenomena (e.g., temperature, relief) and discrete identifiable objects (e.g., buildings, rivers, forests).

The GDL model is based on three class hierarchies: Conventional-Classes (for modeling non-georeferenced entities); Geo-Classes (which are specialized into Geo-Field and Geo-Object, to accomodate field and object views); and Metadata-Classes, which are fundamental in a GDL. Metadata are organized according to a class hierarchy that combines and extends features from Dublin Core network data standard [WGMD95] to the FGDC georeferenced standard [Com].

We use the concept of parametrized views of the underlying database [AB91], in order to allow the dynamic specification of contexts, according to the users' mental models. This helps on solving the problems of lack of orientation and cognitive overload. Other distinctive features of our GDL model are: (i) inclusion of multimedia queries and methods to generate dynamic links and navigation paths inside a context; (ii) allowing users to alternate between the standard hypermedia navigation mode and the database query mode; and (iii) the combination of dynamic links and parametrized database views to add flexibility to the management of the natural growth of the GDL [Gon97].

3.2 Navigational Design of GDL

The navigational design is responsible for specifying navigation paths among context nodes and contexts. It describes the dynamics of the GDL and is therefore of direct interest to user interaction issues. Besides predefined paths, users are offered facility of the *Selection/Action* navigation paradigm.

In this paradigm, the user selects parts of data in one node (e.g., one section of a text) and performs an action on the selected data. A typical action is navigating to other data that are associated to the selected data. We use the concept of *transient anchors* [GT96]: the selection-based navigation uses the contents of the node, rather than predefined anchors, allowing the users to explore generic dynamic links. Modelling this style of navigation involves extra activities, such as defining sets of data that are reactive, specifying generic dynamic links to transient anchors, and defining some auxiliary menus on the interface.

Navigational design consists basically of defining the classes users will access in the GDL. Figure 2 shows the database schema of the basic hypermedia classes of the GDL. The Database class, for instance, allows specifying external databases as objects that can be reached from the library via navigation paths. A *temp_appl* anchor represents the *temporal*

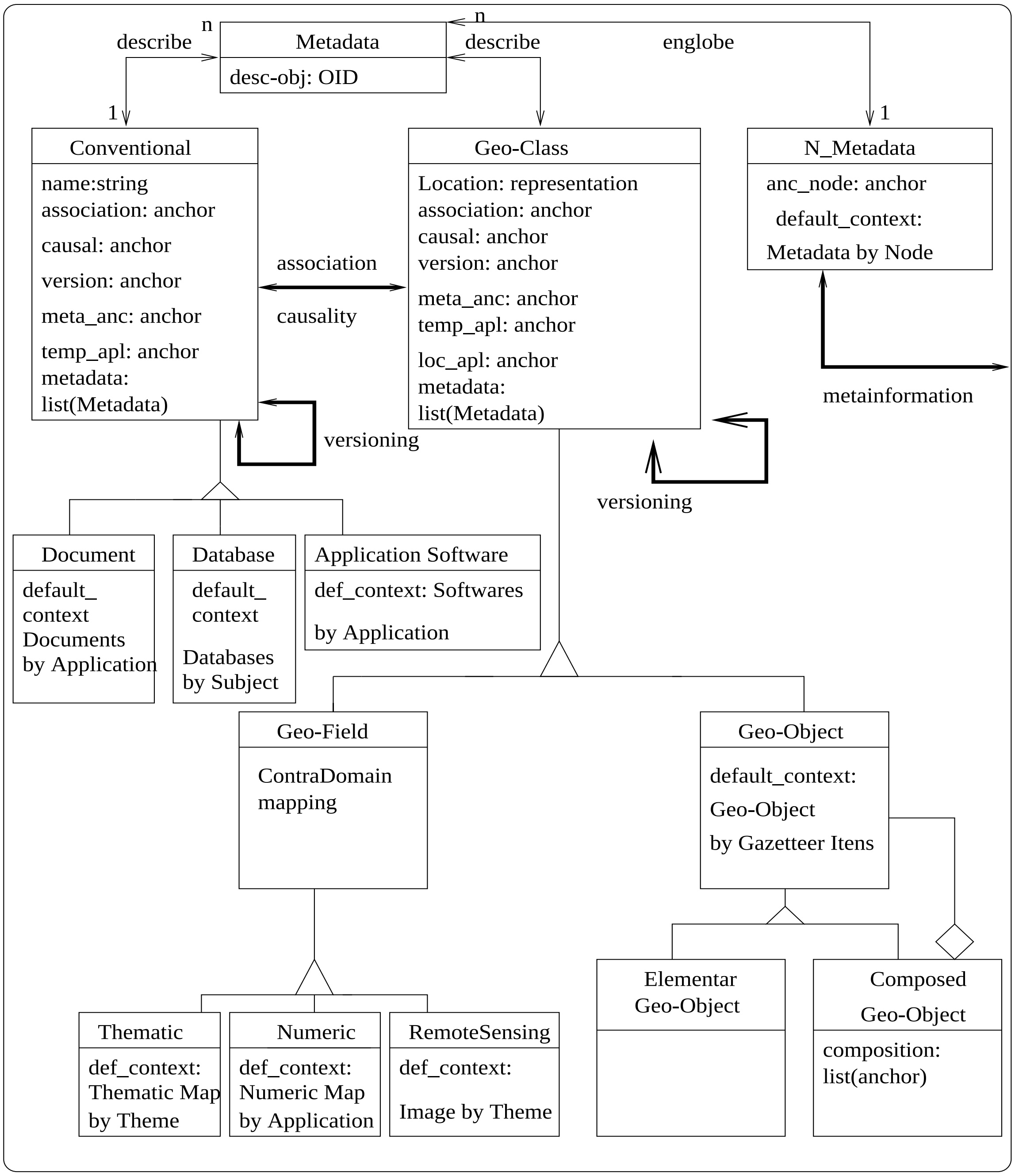

Figure 2: Navigational Schema of the GDL

relationships, and a *loc_appl* anchor allows connecting data which are related by a location (spatial) relationship.

The *N_Metadata* class encapsulates all metadata that refer to a particular node. Therefore, queries used by dynamic links can use the object's attributes and its metadata, and both are stored on the same database. This simplifies the management of queries, since there is a uniform treatment for system level queries (for navigation) and for user defined queries (for selecting data). Moreover, the navigational design of the library is thus closely related to the available metadata.

3.2.1 Contexts

Contexts are roadmaps to guide the user in browsing and querying the GDL. They are organized and interconnected in the GDL according to the assumption that browsing via hyperlinks and database querying are the two main interaction paradigms. These interaction styles are integrated and may be interleaved according to the users' needs. The basic set of contexts is shown in figure 3. The rightmost part of the figure contains the context, whereas the rest of the figure shows how the user can access them. Arrows indicate permissible navigation directions and change of context. Contexts are represented by boxes with dotted lines; contexts are grouped within dashed-lines boxes; and classes are boxes built with solid lines.

The leftmost block describes the GDL main access menu containing entries for indices (which may be nested in arbitrary depth) and applications; the right block describes the navigation contexts and their interconnections; the middle block describes available navigation paths. The term *index* is used here in the sense explored in hypermedia applications, i.e., it is a type of menu that allows direct access to a context. For instance, the selection of the "Time" option from the main menu triggers the "Time Form" application, while the selection of a given anchor gives access to a specific context within three main contexts:

1. Geo-Class: contains navigation contexts related to Geo-Classes, grouped in the *Geo-Class by Location and Time* context. The semantics of this context is that objects within it satisfy a given time and space interval constraint.

2. Conventional-Class: in a similar way, contexts involving Conventional-Classes are constrained by a time interval specified by the *Conventional-Class by Time* context.

3. Metadata: this context can be reached from the other two contexts. It allows accessing the N_Metadata object for a given node of the GDL, through the *Metadata by Node* context.

Consider, for instance, context Image by Theme. Users who work in this context will have access to navigation across Image data linked according to theme (e.g., vegetation or soil). Arrows indicate when it is possible to go from one context to another. For instance, it is possible to navigate from an *Images by Version* context to an *Images by Theme* context and vice-versa, since every image has a theme and a version. However, it is not possible to navigate from *Images by Theme* to *DTM by Application* because there are no images in the latter context.

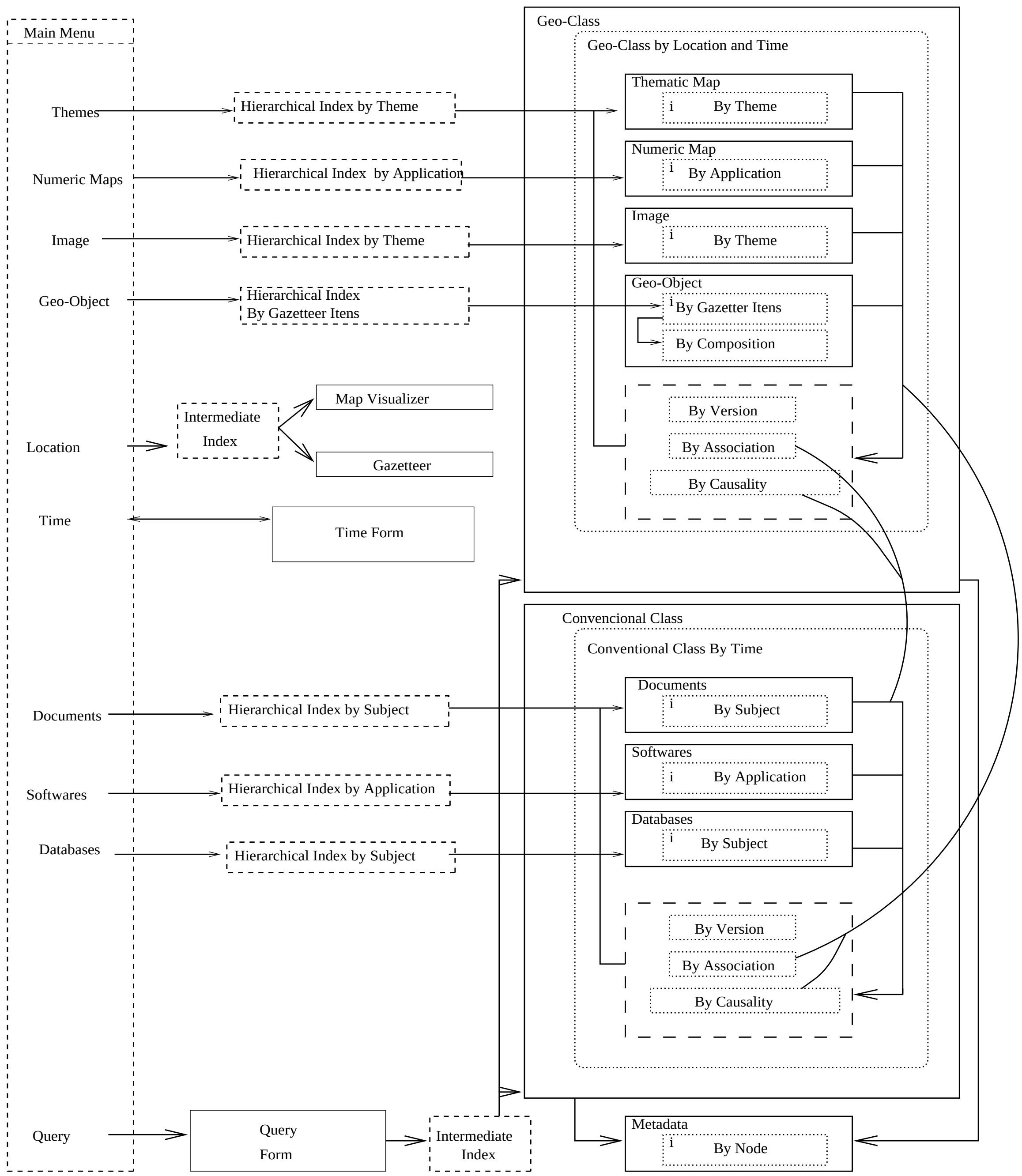

Figure 3: Navigation Contexts in the GDL

A context definition has to specify (i) the name and type of the context; (ii) its nodes, which may be defined explicitly (giving the name of each node) or indirectly (using a query); (iii) the associated Contextual Information; (iv) the entry points (nodes of the context that can be reached externally); and (v) the path (ordering of elements for navigation). Moreover, a context may define attributes, such as indices to be shown when the context is activated.

Our architecture meets the functionality requirements of a GDL, but is based on a complex model. Applications built within the GDL architecture should hide its complexity from the end user of the library. We next show how to specify the interface components of context nodes and the interface of the GDL.

4 The User Interface for the GDL

The user interface of the GDL is based on the navigational model described in section 3. The idea is to construct the user interface of the whole library as a set of hypermedia applications which define the user interfaces of each navigation class in the GDL model (see figure 2). The definition of a user interface for a navigational class involves:

1. The appearance (*look*) of the presentation of the instances of the class. It is important to support different presentations for instances, according to specific contexts. It is also important to note that these presentations may contain auxiliary interaction objects, to represent application functions, besides the widgets that represent the instance's contents.

2. The behavior (*feel*) of the user interface. It is necessary to describe the reaction of the interface to two types of external events: users' events, such as mouse and keybord actions, and applications' events, for example, to update the contents of the instance.

4.1 GDL User Interface Design

For GDL users, the hypermedia interaction element is the context node. We recall it is defined as a tuple $< node, context - info$, ADV$>$. The logical design of the GDL user interface is based on the Abstract Data View (ADV) concept, introduced in [RSLC95]. An ADV is an object, in the object-oriented sense, that describes the user interface aspects of a hypermedia object. The ADV component of a context node contains the subset of the node's properties which are visible to the user, and the users' events definitions for these properties.

An ADV object may be considered as a proxy object for user interface purposes. Therefore, the structure of the ADV reflects the structure of the represented object. A node's ADV contains nested ADVs that represent the visible attributes of the node. The anchors inside the node are associated to active nested ADVs (e.g., navigation buttons). Standard ADV, such as text, image, and buttons, may be stored on user interface libraries and reused on new logical designs.

A *configuration diagram* is a graphical representation of an ADV, defining (i) the interface objects and their behavior (in terms of users' events), (ii) the services provided by the

ADV (e.g., display), and (iii) the interaction between the ADV and the object it represents. Figure 4 shows an example of an ADV for context nodes of type Geo-Class. Navigation Classes Geo-Class and Metadata Class (on the right) are linked to the Geo-Class ADV. User interactions invoke methods that get data and metadata from the classes extension (e.g., *get attributes*) and have data displayed by the interface.

The main ADVs of the GDL can be classified in: (1) *Geo-Class* ADV (Thematic, Numeric, Image, and Geo-Object); (2) *Conventional-Class* ADV (Document, Application, and Database); (3) *Metadata* ADV (N_Metadata); and (4) *Support* ADV (Navigation Menu, Query Interface, and Context Interface).

4.1.1 Support ADV

The Context Interface ADV contains ADVs that represent anchors for intra- and inter-context navigation. The Navigation Menu ADV allows users to follow links based on selections on the node's contents. The navigation may be in the forward, backward, or in any direction, according to the value of the direction parameter (*forward, backward* and *any*, respectively).

When a context node is accessed, the content of its Context-Info component are presented. For instance, anchor attributes *next* and *previous* are presented for intra-context navigation. The Query Interface ADV includes buttons for anchors that allow accessing different GDL services, supporting distinct query interaction models. For instance, The anchor to map visualizer will invoke an application which allows the user to query the GDL using a map interaction and visualization paradigm.

4.1.2 Geo-Class ADV

The Geo-Class ADV is a generic ADV for describing objects that are instances of Geo-Class, and its subclasses. Figure 4 presents a configuration diagram for this ADV, containing several nested ADVs (e.g., Image, Description, Navigational Menu, Context Interface).

The Description and Metadata ADVs in the diagram refer to metadata of the Geo-Class. The text presented in the Description ADV is retrieved from these metadata. The Metadata ADV defined within the Context Interface represents an anchor and is active. It allows users to navigate to a predefined navigation context, namely *Metadata by Node*, where the complete metadata set of a node is available. Moreover, the strings ADV (e.g., name or location) represent attributes and metadata of the Geo-Class and are also reactive, allowing the navigation based on selection.

The external events defined for the Geo-Class ADV are *Display, MouseOn, MouseClicked, MouseDoubleClicked,* and *SelectionOnText*. The behavior of the ADV, discussed in section 4.1.4, is based on these predefined events.

4.1.3 Other ADVs

The generic ADV for the subclasses of Conventional-Class is similar to that for the Geo-Class. The Conventional-Class ADV includes specific attributes for each subclass: *publisher*, and *document type* for Document subclass ADV; *file type*, and *platform* for subclass Software ADV; and *schema information* for Database subclass.

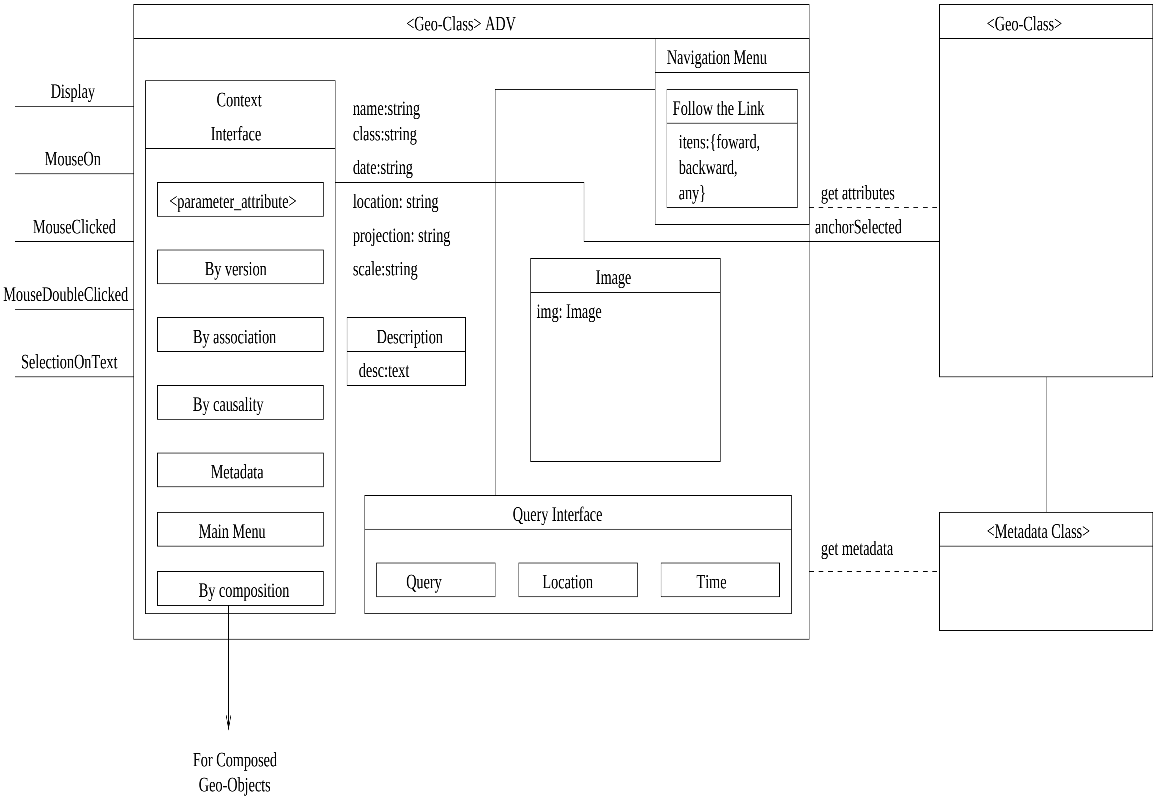

Figure 4: Generic Configuration Diagram for Geo-Class Subclasses

In GDL interfaces, metadata are a very important source of information, and this interface design must consider this issue. The configuration diagram for the N_Metadata ADV is shown in figure 5. It can be interpreted in the same way as the previous one.

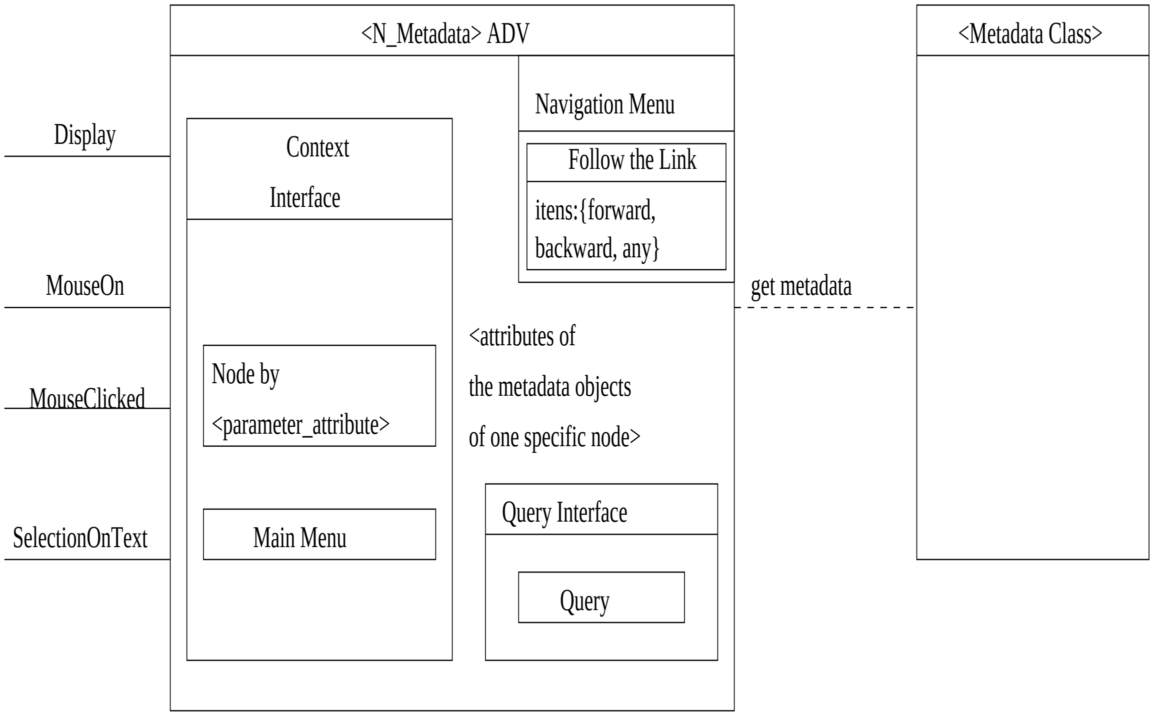

Figure 5: Configuration Diagram for N_Metadata ADV

4.1.4 ADVCharts

Statecharts are the *de facto* standard notation for user interface behavior [Har87]. ADVCharts are a specialization of StateCharts, allowing the application's designer to express the associations of external (users) events and ADVs, and to define the behavior of the interface in response to each event. Figure 6 describes a generic ADVChart for Geo-Class. Interface states are denoted by rounded rectangles. Lines starting on nested ADVs indicate events and related actions. Arrows define state transitions.

The following example illustrates the behavior of a ADV. Consider the ADVchart for the Image class, which is a subclass of Geo-Class, and assume it uses the same ADVChart of figure 6. Suppose this chart defines two internal states: normal and zoomed. Consider, futhermore, that a given Image object is being displayed within an Image context. When a *MouseDoubleClicked* event occurs with the focus on the Image object, the ADVChart defines a transition from normal to zoomed, if the current state is not already zoomed. Otherwise, it defines a transition from zoomed to normal. These transitions have side effects in the interface appearence; for instance, the size of the ADV display is changed.

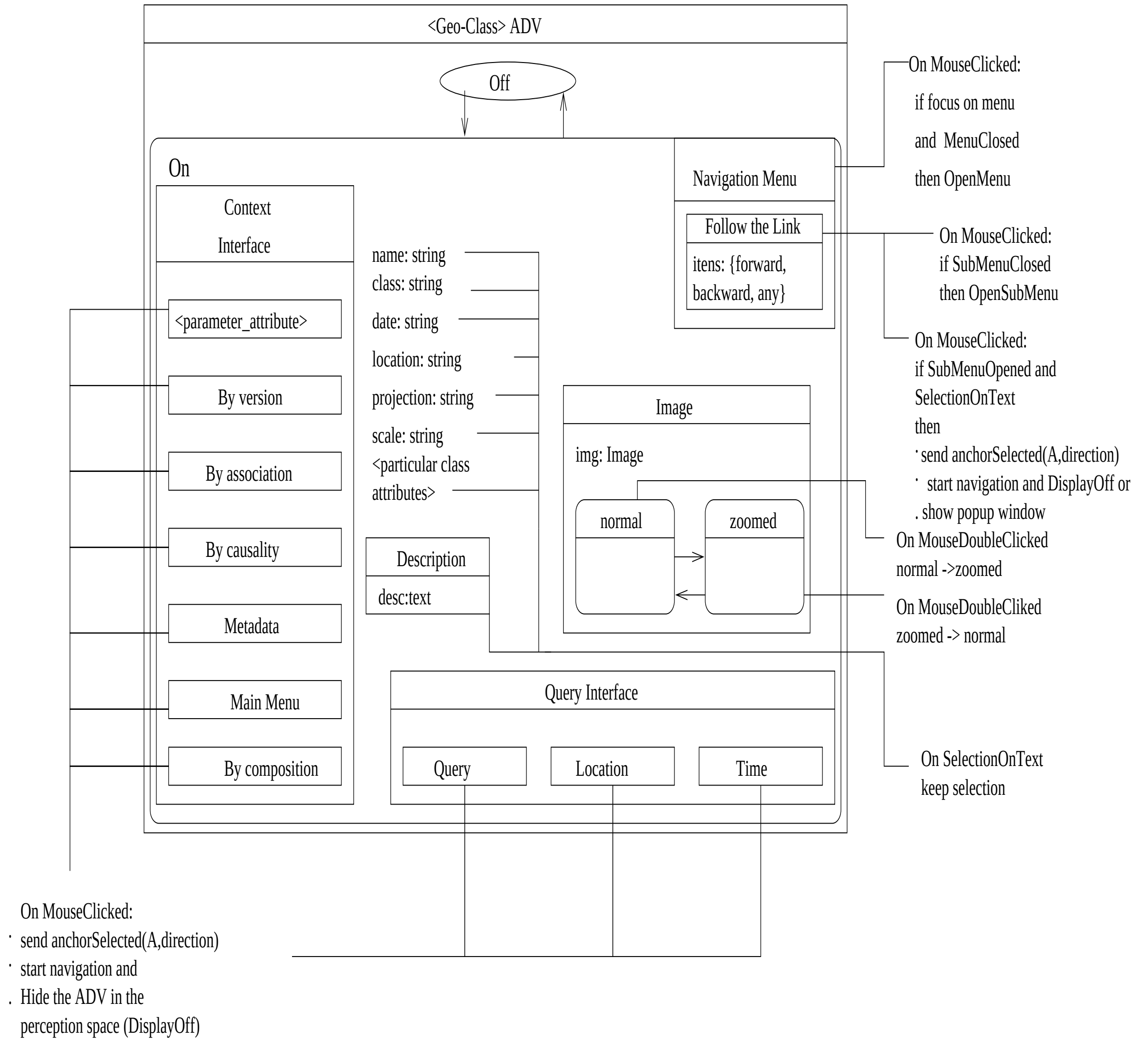

Figure 6: Generic ADVChart for Geo-Class

4.2 User Interaction with the GDL

In GDL, the querying and browsing facilities are equally important. Our query model defines a symbiotic integration of hypermedia navigation facilities - based on browsing hierarchical structures and on hypermedia links - with database querying facilities - based on contents of data and metadata. These integrated facilities define the generic interface mechanism, which can be customized to different users and applications.

A working session in GDL may have three starting points: (a) the Hierarchical Indices, to access navigation contexts; (b) the Query Form, to submit a specific query to the library; and (c) the services, which are special GDL applications (Map Visualizer, Gazetteer, and Time Form), which restrict the spatio-temporal space for navigation (see figure 3).

Users who are not familiarized with the schema of the GDB underlying the GDL will probably start with the first option, which involves browsing functions. The other two options involve querying the database metadata catalog, and will be discussed latter.

4.2.1 Browsing the Library

The GDL main menu offers an overview of the available data and functions. The buttons in this menu provide to the user a predefined set of hierarchical indices to reach contexts. Some of these contexts may be dynamically built, using users' parameters for selecting time and space ranges.

The hierarchical indices that give access to contexts organize geographic data in two ways: intrinsic categories and thematic categories. Indices are classified by *Theme*, *Application*, *Gazetteer Item*, and *Subject*. For instance, the Theme index may provide access to common geographic themes, such as vegetation, transportation, and hydrography, organized according to Thematic Map and Image contexts. We recall that a given GDB object (Node) may appear within different contexts (context-info), and be customized in several ways (ADV).

Besides the conventional top-down navigation via hierarchical indices, GDL offers the following navigation facilities: (i) intra-context navigation, for browsing inside a specific context; (ii) inter-context navigation, for changing the contexts while browsing in the GDL; and (iii) navigation independent of context (based on selection), supporting dynamic requirements via transient anchors associated to the Navigation Menu.

4.2.2 Querying the Library

The GDL supports two basic types of queries: (i) those restricting the navigation space, by reducing the data available to browse; and (ii) those expressing specific data retrieval operations.

The first type of queries are contemplated by services associated to the library. These services are: *Map Visualizer*; *Gazetteer*; and *Time Form*. The user can activate these applications using the buttons Location, Time, and Query of the GDL Query Interface. The Location button points to an index whose elements are directed towards the Map Visualizer or to the Gazetteer. The Time and Query buttons activate, respectively, the Time Form and the Query Form. The Query Interface is available in the GDL main menu, and also in all ADVs for navigation classes.

Services are responsible for the definition of temporal and geographic navigation ranges, allowing the user to restrict the default ranges. The Map Visualizer and the Gazetteer define the geographic region the user is interested in while browsing and querying the GDL. The Time Form allows users to specify valid time parameters for searching and retrieving data. The user's choices are added as parameters to the two main data contexts (Geo-Class by Location and Time, and Conventional-Class by Time).

The second type of query is supported by the Query Form. The user accesses this form using the Query button. The queries in the Query Form are restricted to the context from which the form was activated. The information about the current context is a parameter for the activation of the Query Form, allowing the corresponding service to customize its user interface, presenting data and metadata according to the specific class that activated the form.

5 GDL User Interface Framework

Geographic interfaces are particular instances of complex GUI (Graphic User Interface) applications. In complex interactive systems, such as GDL, more than fifty percent of the total amount of code is dedicated to the user interface [MvD91, MR92]. In spite of the great advances in mechanisms for storing and manipulating georeferenced data, the user interface still presents a barrier to the efficient use of geographic systems.

In GDL, besides having to consider the usual interface issues, the control (widgets) area, the interface system has to support cartographic manipulation. We recall that users may interact with the GDL either through the context paradigm or directly via query services. Futhermore, a context node may provide means for navigating via links or switching to the query mode. This flexibility, added to ADV customization, presents problems to interface building. The previous section showed our GDL model and how it supports flexibility in user interaction modes. We now show how present interface architectures are not appropriate to meet our GDL model, and present our framework for building a GUI for the GDL. Conceived as a general framework for building GUI, it can be used for constructing interfaces for the GDL, as will be shown.

Building a user interface within the framework presented in this paper is a two-phased process. In the *design phase*, the interface designer defines templates for interface objects directed to the specific type of dialogue determined by the application. The templates are stored within the database, forming what we call the IMOD DB, which can be regarded as a library containing models of interface objects. In the *implementation phase*, these models are instantiated with data from the GDL, generating complex interface objects. Though stored with the rest of the GDL data, the IMOD DB is treated as a separate entity because of its nature.

This section describes our architecture, showing how it can support the functionality demanded by the GDL model. Due to space limitations, we present here only an overview of the framework, and a brief discussion of its key aspects. The interested reader can find a thorough discussion in [Oli97].

5.1 A New GDL User Interface Architecture

We consider the user interface as an interactive software component to be coupled to the GDL. A user interface architecture for a GDL has to define four important aspects. The first issue is the *integration approach*, i.e., the mechanism for communication with the underlying (GDL) software. Second, the *main modules* of the interface component should be identified, and their functionality and interoperability should be specified. Next, the architecture should define an *intermediate data model* and the mapping mechanisms to and from the underlying GDL data model. Finally, the *division of tasks* between GDB and interface should be clearly defined. These aspects guide the overview of our architecture, presented below.

[Oli97] presents a general geographic interface architecture. Here, this architecture is tailored to a GDL, using two principles. First, we take advantage of the underlying GDBMS to provide functionality which is normally provided by interfaces (e.g., customization mechanisms), thus simplifying the role of the interface developer, and decreasing the cost of system maintenance. Second, we provide a layered interface architecture which clearly separates the tasks of user interaction and display facilities from those of communication with support systems (GDBMS and interface packages). This facilitates reuse of interface modules and helps designing different look-and-feel styles for the same GDL. The user can interact with the GDL in two ways: via hypermedia contexts and via database queries. In both cases, the interface interaction dynamics is specified using ADV.

5.1.1 Interface Software Organization

Our GDL interface framework is based on the generic geographic interface architecture proposed by [Oli97], which decomposes the interface software into three major layers. Each layer is responsible for the management of a well defined set of tasks: the lower layer handles the communication with the underlying support software (Connection Layer); the intermediate layer manages the data models involved in the interface processing (Data Models Layer); and the upper layer performs the specific user interface functions of each individual application (Application Layer).

Although this generic architecture supports the specification of general purpose geographic application interfaces, we focus here in the design and implementation of the interface for the GDL. To meet the goal of developing the interface component for this target application (the GDL), we redefined the generic architecture, taking into account the features and properties described in sections 3 and 4. Figure 7 shows the software architecture resulting from these refinement process.

Connection Layer The lowest layer of the architecture establishes the communication with the underlying support systems. In geographic interfaces, two support systems are essential: the underlying GDBMS and the user interface packages (graphics and widget toolkits). For each support system, the Connection Layer provides an Adaptor Module, which

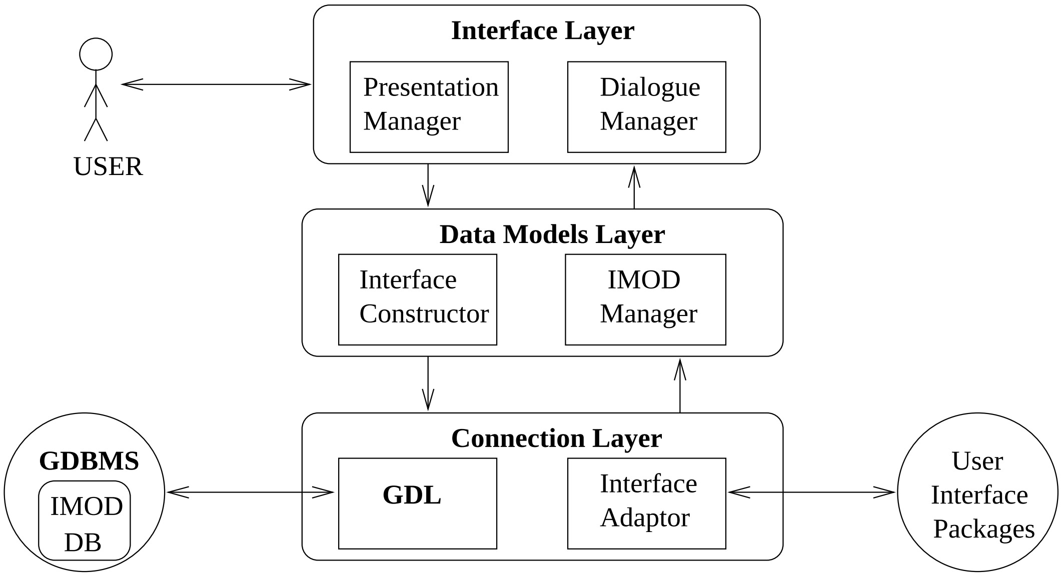

Figure 7: Architecture of GDL User Interface

offers to the other layers of the architecture a normalized access to the support system, independently from the specific software addopted.

The Adaptor Modules have three main objectives: (i) to guarantee portability, by encapsulating heterogeneous support systems in specific Adaptors, which work as device drivers to these systems; (ii) to promote reuse and standardization, by sharing the services of Adaptors in all modules of the GDL, rather than implementing different communication protocols with each specific support system; (iii) to support software maintenance, as a by-product of the encapsulation of the support systems. Each Adaptor defines, basically, an abstract machine with an uniform programming interface. The upper layers use the same communication protocol, regardless of the chosen underlying system. We give now the basic ideas of the two main Adaptors of the architecture.

Different toolkits have distinct representations for widgets, following diverse look-and-feel definitions. However, it is possible to identify a basic set of widget types that are present in most toolkits. The Interface Toolkit Adaptor defines this fundamental set of widgets and implements mapping rules to the specific widget sets of each toolkit. The same idea is also present in the user interface architecture presented in [BFL+92], and in the multiplatform toolkits [Val89, MMM+97]. The Interface Adaptor allows the mapping of abstract user interface objects to real toolkit widgets. The abstract interface objects are defined using the IMOD model (described in the next section).

Data Models Layer This layer is responsible for providing the mapping between the user mental model and the GDBMS, managing the data exchanged between the user interface and the application and supporting the construction of the user interface by means of instantiation of interface templates from the IMOD DB.

The IMOD model supports the (automatic) creation of presentations for data described in GDB schemata. It offers abstractions for the representation of the basic set of widgets (specified by the Interface Toolkit Adaptor), and for associating these interaction objects with data stored in the GDBMS. A Constructor module, associated to IMOD, allows the composition of complex interface objects from simple abstract widgets stored in the IMOD DB. This Constructor maps complex interface objects into generic widgets available from the Interface Toolkit Adaptor.

Application Layer The upper layer of the architecture implements the conventional user interface components. Here, the user interface component is logically subdivided into Dialogue Manager module and Presentation Manager module. The Presentation Manager module has two main tasks: transforming users' actions (i.e., users' events) into operations on interface objects of the IMOD DB; and managing the graphic and textual presentations of data.

The Dialogue Manager module is responsible for managing the dynamic behavior of the user interface, including the behavior of the Presentation Manager module. For instance, an interaction object of the IMOD DB may be associated to more than one geographic object (defined in a GDB). The Dialogue Manager module keeps track of these associations, and propagates events and actions for the appropriate objects.

5.1.2 Interface Construction Model: IMOD

Our GDL User Interface Framework is based on two models: the GDL data model and an interface data model. The first, described in section 3.1, is used as the interface intermediate model. The latter, IMOD, is used to built the interface for these data.

IMOD is an object-oriented data model allowing the recursive specification of complex interface objects from a hierarchy of classes that define basic interface objects. Building an interface within our framework involves the creation of IMOD objects and the instantiation of these objects with GDL instances stored in the GDB. The kernel components of the IMOD model are shown on figure 8.

The basic components of IMOD can be considered *abstract widgets*. They provide the basic features of widgets from most user interface toolkits. These abstract widgets can be mapped to real widgets by the Interface Adaptor of the architecture.

The main means of constructing the interface look-and-feel is via the *window* concept, i.e., the screen is a window, which can be composed of several elements (e.g., windows, buttons, menus) and the whole can be described by progressively building complex interface objects described in terms of IMOD. An interface model in IMOD is a composite and recursive template of basic interface components, with a single composition element: the Window class. A composite window contains other windows, which may be composite or simple. A simple window contains only basic IMOD objects (i.e., they are not window objects). Every component of IMOD is contained in a window, which is called the component's ancestor. Windows that are not nested may be associated with each other through the *associated window* relationship. This is useful, for instance, to determine the sequence of windows in a browsing session. Thus, in the GDL, the window is the basic template.

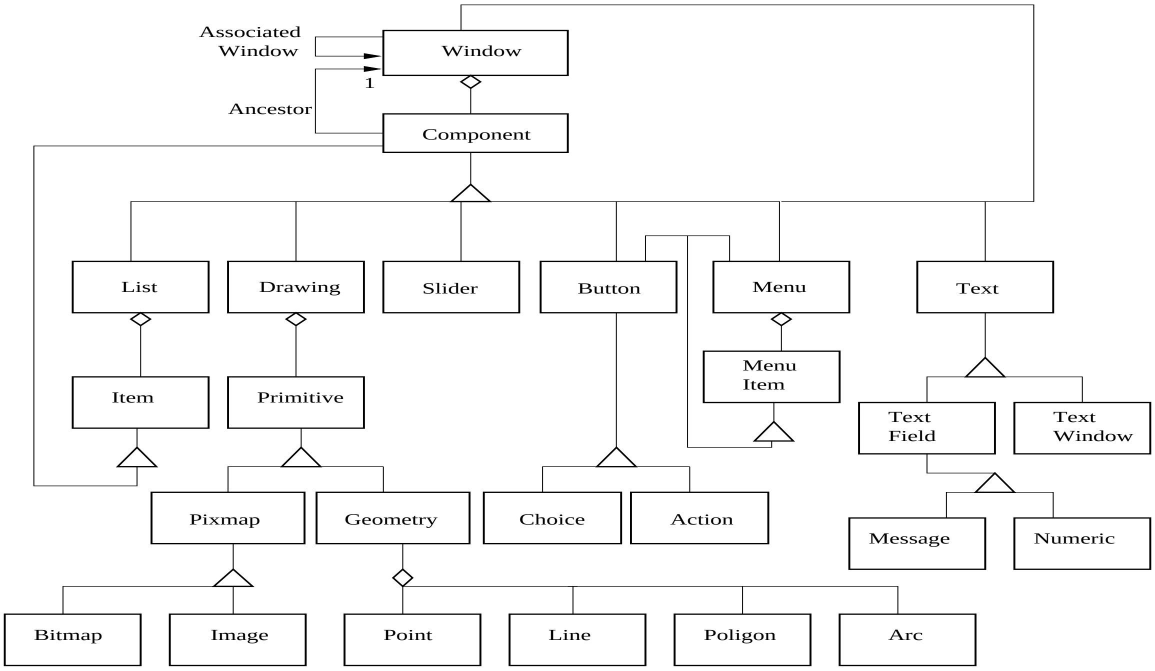

Figure 8: Interface Data Model (IMOD)

One important characteristic of IMOD is that it is extensible. New interface objects can be specified using the model by adding and specializing classes in the kernel model presented in figure 8.

Another important concept of the IMOD kernel is the support to graphic drawings. The model allows the definition of complex drawings by combining geometric (e.g., points, lines, polygons and arcs) and pixmap (e.g., bitmaps and images) primitives. The combination of these primitives is essential for cartographic display in GDL.

Graphic drawing primitives can be used to construct arbitrary maps in the following way: a *graphic instance* is an IMOD drawing object representing a geographic object (stored in the GDB). Graphic instances in a given geographic region are grouped in a *graphic layer*. The interface model supports the selection of instances that compose a graphic layer (e.g., the instances retrieved by a query to the GDB). A *map window* is a specialization of the IMOD basic window that allows the visualization and manipulation of graphic layers and their graphic instances.

5.1.3 Interoperability and Functionality of the Modules

Rather than giving a complete formal description of the functionalities of each module and of the way they interact, we present an example that shows most types of interactions, and the typical functionality of each layer of the architecture. Assume the user starts interaction with GDL by selecting the context Image by Theme for a given spatio-temporal range, for theme = vegetation. The following illustrates, in a schematic way, a dialogue between a user and the GDL.

- The Presentation Manager module translates the user's action into a IMOD object operation, which is forwarded to the Dialogue Manager module. This module maps the IMOD operation into a GDL operation, triggering a GDBMS action.
- The GDBMS is called to build the corresponding context, and returns the set of vegetation images within the given spatio-temporal range.
- The Interface Constructor module is called to build a presentation for the given GDL objects. The type of presentation (in this case an entry point for each vegetation image in the view). When the user selects one of these entry points, the Contructor generates an image window (context node – implemented according to the Geo-Class ADV).
- The type of window to be built is passed as a parameter to the Constructor, which generates a presentation, defined in terms of IMOD interface objects. This objects are forwarded to the Interface Toolkit Adaptor, which creates the actual widgets.
- The Dialogue Manager module changes the "state" of the interface, to record the presence of another window.
- The user can thus navigate along several vegetation images, following simple links within the context. Changing contexts (e.g., to Image by Version) may be performed in a similar way: again, the user's event is translated into an IMOD operation and after into a GDL operation.
- At some point in time, the user may decide to perform a database query involving, for instance, metadata about a given image. This corresponds to invoking an ADV Query Form within the Image by Theme context. Again, this will result in a similar sequence of translations, between GDBMS data and IMOD templates, mediated by the Constructor.

We recall that the change of interaction mode (from browsing to querying) may be transparent to the user. In fact, the activation of a Query Form may be obtained by clicking on context dependent information (e.g., a context specific anchor).

5.2 Customization of Geographic User Interfaces

There is often a mismatch between the users' understanding of the world and the models and operations provided by particular geographic user interfaces. A trade-off solution, based on customization capabilities, is adopted in many systems. However, in current customization approaches, the users' choices are usually limited to a small number of predefined interaction mechanisms. Thus, the users have to adapt themselves to one of the available interface paradigms.

Our approach brings the interface into the GDBMS, by extending existing GDBMS modules in order to augment the power of interface development mechanisms. The basic ideas of this approach were presented in [OMC97]. The mechanism is based on the active database paradigm, associated with the use of the IMOD DB. Unlike traditional approaches, the customization options are not limited to a fixed number of interface styles, and interfaces can be built dynamically.

The notion of using active DBMS for user interface facilities has been previously proposed by [DJPaQ94]. In this particular aspect, it is similar to our approach. However, their emphasis is on dynamically reflecting database state changes in the interface (akin to a view refresh). We, on the other hand, concern ourselves with customization of interface control and display components.

5.2.1 Current Approaches to Customization

A *generic user interface* mechanism is software that provides basic, standard, user interface look-and-feel characteristics for a specific type of application, such as DBMS and DL. Section 4 defines the generic user interface for our GDL. However, this generic user interface must be customizable.

Many solutions have been presented to deal with this necessity of customization. These solutions can be summarized in three main approaches:

1. *Multiple interaction paradigms*: the interface offers many interaction paradigms, letting users decide which one is best for their needs;

2. *Stereotypes*: the interface is adapted to each user's mental model. The interaction with the system classifies the user into an stereotype, which defines the interface behavior;

3. *Toolkit*: the interface is formed by a generic interface basis, and a toolkit for customization. The user must rely on a programmer to get a full-customized interface.

None of these approaches solves the problem. In the first and second cases, the user is limited by the predefined interaction modes or stereotypes. In the third case, customization cost is increased due to the need of an application programmer to develop completely new interface code. Our solution is closer to the third approach. However, the programmer is helped by modules which are embedded in the geographic database.

5.2.2 Active User Interface Customization

Our approach to the problem of GDL interface customization is based on two main components: the GDL interface mechanism and an active database mechanism. The architectural and functional relationships among these components are shown in figure 9. The dynamic construction of user interfaces based on IMOD templates is the keystone of the customization process.

The generic user interface mechanism provides the framework for *ad hoc* interaction with the GDL. User interactions are interpreted by the interface and transformed into *database events* – requests for queries or updates – which are then passed on to the underlying geographic database. In active DBMS, specific events are interpreted by an active mechanism,

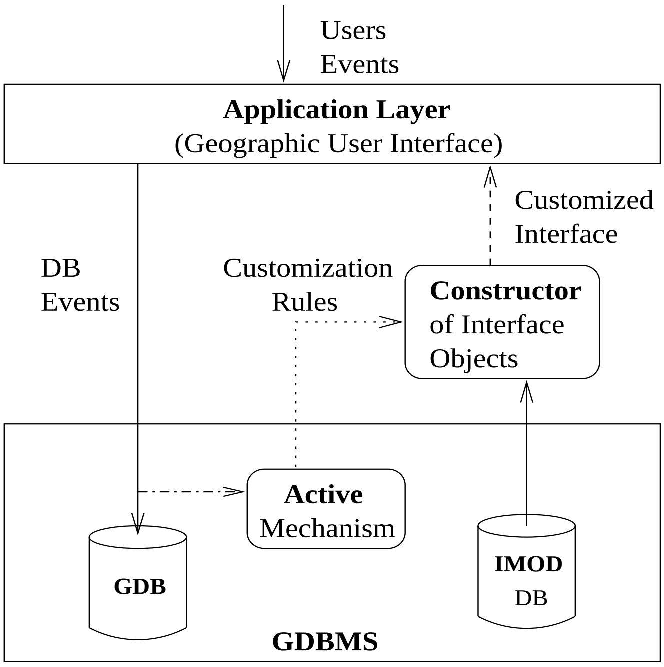

Figure 9: Customization of Geographic User Interface

which then coordinates subsequent actions. In our case, we are interested in interface customization. Thus, the active mechanism is now responsible for interface customization. Now, the DB events (on data and metadata) are intercepted by the *active database mechanism*, which activates appropriate *interface customization rules*. (It may also activate semantic data management rules, but this does not concern us).

These rules are declaratively specified through a *customization language*, which allows the user interface programmer to define specific look and feel behavior for the interface. Each rule acts on the appropriate interface objects of the IMOD DB. The result is processed by the *constructor* module of the user interface framework, which generates a *definition of a customized interface*. This definition is used to dynamically generate the output screen objects (e.g., maps and/or widgets).

A declarative customization may spawn several customization rules, which look like traditional E-C-A (Event, Condition, Action) rules [Buc94]. However, customization rules define new semantics for the E-C-A components:

- Event: the conventional database event is now composed by user interface (e.g., mouse click) and database (data semantics) events.

- Condition: the traditional database condition is replaced by a *use context* condition, which refers to the user and the application accessing the GDL. The condition clause specifies either some user profile or the navigation context, corresponding respectively to interaction via database query or to hypermedia browsing (and thus to some database view whose interface look-and-feel was specified by some ADV).

- Action: instead of activating a database query language code, the action part of the customization rule redefines a given aspect of the IMOD template for the corresponding context node.

Traditionally, the use of an active mechanism can cause conflicts, since rules can trigger other conflicting rules. This is not the case for interface customization rules because the action part of a rule is limited to defining a customization for an interface object.

The same example of section 5.1.3 can be followed, with an additional step. Now, instead of generating a presentation by combining GDBMS data to the default IMOD template, the Constructor:

- receives the data from the GDBMS;
- receives the customization code from the $< Action >$ part of the customization rule (which is stored in the GDB);
- merges data and IMOD templates according to the customization code.

Thus, in theory, each user can have a different interface built just by having appropriate rules added to the active database.

6 Conclusions

The need for dissemination and sharing of information on the net has prompted the appearance of digital libraries. However, electronic documents need not follow the same restricted uses of traditional library documents. Thus, interfaces to DLs should offer their users a wide degree of flexibility in library access and usage patterns. This, in turn, presents many challenges to designing and implementing adequate user interfaces. GDL augment these challenges by adding to the existing problems the issues of data spatiality and temporality, as well as questions of cartographic presentation and geographic data representations.

This paper presents a solution to this problem, which consists of combining facilities from three domains: database systems, software engineering and user interfaces.

The flexibility in data access and usage patterns is provided by a specific GDL organization, which is based on combining hypermedia navigation to database querying, using a geographic data model. Different user needs can be accomodated by the specification of contexts, which allow filtering information in a natural way. Contexts have their interface presentation and interaction dimensions specified by means of ADV and ADVCharts, which allow customizing contexts to different users, and support interchangeable interaction modes with both data and metadata.

The flexibility in interface design and programming, from a software engineering point of view, is provided by the GDL interface framework, which is based on clearly separating the user interaction layer from the underlying support services (GDBMS and interface toolkits). The interface intermediate model is the GDL model itself, which in turn approaches the user mental model. The use of an extensible template library (IMOD) supports interface customization, thereby implementing ADVCharts and providing dynamic interface construction facility.

Parts of this interface framework have already been developed. The use of the IMOD library has shown considerable advantages, reducing by 1/3 the amount of code dedicated to implement a geographic user interface [OCM95].

Our geographic user interface framework solves important drawbacks of user interface construction. As the previous architectures, we separate the user interface and the semantic kernel of the geographic application. Unlike the existing architectures, we restrict neither the functionality of the underlying GDBMS, nor the functionality of the GDL applications. Moreover, the abstract model in which our architecture relies is directly mapped to an implementation model, using common user interface development tools currently available.

Future work will be geared towards implementing the gdl model, using biodiversity data. At the same time, experiments are being conducted on extending the interface capabilities of an environmental application facility, to obtain performance data on the effectiveness of our interface architecture proposal.

References

[AB91] S. Abiteboul and A. Bonner. Object and views. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, pages 238–247, 29-31, may 1991.

[AFY94] N. R. Adam, B. S. Fordhamm, and Y. Yesha. Some key issues in database systems in a digital library setting. In *Lecture Notes in Computer Science*, volume 916, pages 9–20. Springer Berlin, 1994.

[AHN94] N. R. Adam, M. Halem, and S. Naqvi. Promissing research directions in digital libraries. In *Lecture Notes in Computer Science*, volume 916, pages 21–30. Springer Berlin, 1994.

[ALD+95] D. Andresen, L.Carver, R. Dolin, C. Fischer, J. Frew, M.Goodchild, O.Ibarra, R. Kothuri, M. Larsgaard, B. Manjunath, D. Nebert, J. Simpson, T. Smith, T. Yang, and Q. Zheng. The www prototype of the alexandria digital library. In *Proceedings of the International Symposium on Digital Libraries*, Tsukuba, Japan, 1995.

[All95] R. B. Allen. Two digital library interfaces wich exploit hierarchical structure. In *Proceedings of Electronic Publishing and the Information Superhighway (DAGS95)*, 1995.

[AYA+92] D. Abel, S. Yap, R. Ackland, M. Cameron, D. Smith, and G. Walker. Environmental Decision Support System Project: an Exploration of Alternative Architectures for Geographical Information Systems. *International Journal of Geographical Information Systems*, 6(3):193–204, 1992.

[BAN94] K. Böhm, K. Aberer, and E. Neuhold. Administering structured documents in digital libraries. In *Lecture Notes in Computer Science*, volume 916, pages 91–118. Springer Berlin, 1994.

[BC91] L. Bass and J. Coutaz. *Developing Software for the User Interface.* Addison-Wesley, 1991.

[BCGP97a] M. Baldonado, C. K. Chang, L. Gravano, and A. Paepcke. Metadata for digital libraries: Architecture and design rationale. Technical report, Stanford University, 1997.

[BCGP97b] M. Baldonado, C. K. Chang, L. Gravano, and A. Paepcke. The Stanford digital library metadata architecture. *International Journal of Digital Libraries*, 1(2), February 1997.

[BFL+92] L. Bass, R. Faneuf, R. Little, N. Mayer, B. Pellegrino, S. Reed, R. Seacord, S. Sheppard, and M. Szczur. A Metamodel for the Runtime Architecture of an Interactive System. *SIGCHI Bulletin*, 24(1):32–37, January 1992.

[BI95] M. Bieber and T. Isakowitz. Designing hypermedia aplications. *Communications of the ACM*, 38(8):26–29, August 1995. special issue in Hypermedia Application Design.

[Buc94] A. Buchmann. Active Object Systems. In A. Biliris A. Dogac, M. Oszu and T. Sellis, editors, *Advances in Object oriented Database Systems*, pages 201–224. Springer Verlag, 1994.

[BW97] M. Q. W. Baldonado and T. Winograd. Sensemaker: An information-exploration interface supporting the contextual evolution of a user's interests. In *Proccedings of CHI'97*, 1997.

[CCH+96] G. Câmara, M. Casanova, A. Hemerly, G. Magalhães, and C. Medeiros. *Anatomy of Geographic Information Systems.* Décima Escola de Computação, July 1996. In portuguese.

[Com] Federal Geographic Data Committee. Content standards for digital geospatial metadata. http://geochange.er.usgs.gov/pub/tools/metadata/ standard/metadata.html.

[Cou96] S. B. Cousins. A task-oriented interface to a digital library. In *Proceedings of CHI 96 Conference*, 1996.

[CPW+97] S. B. Cousins, A. Paepcke, T. Winograd, E. A. Bier, and K. Pier. The digital library integrated task environment (DLITE). In *Proceedings of the 2nd ACM International Conference on Digital Libraries*, pages 142–151, 1997.

[DA97] P. Duguid and D. Atkins. Digital libraries- report of the santa fe planning workshop on distribued knowledge work environments. Technical report, University of Michigan School of Information, March 9-11 1997.

[DJPaQ94] O. Diaz, A. Jaime, N. Paton, and G. al Qaimari. Supporting Dynamic Displays Using Active Rules. *ACM SIGMOD Record*, 23(1):21–26, 1994.

[Edm92] E. Edmonds. *The Separable User Interface*, chapter The Emergence of the Separable User Interface, pages 5–18. Academic Press, 1992.

[Env92] Environmental Systems Research Institute, Inc. *ARC-INFO: GIS Today and Tomorrow*, 1992.

[FCF+95] J. Frew, L. Carver, C. Fischer, M. Goodchild, M. Larsgaard, T. Smith, and Q. Zheng. The alexandria rapid prototype: building a digital library for spatial information. In *1995 ESRI User Conference Proceedings*. Environmental Systems Reserach Institute, 1995.

[GBA+94] H. M. Gladney, N. J. Belkin, Z. Ahmed, E. A. Fox, R. Ashany, and M. Zemankova. Digital library: Gross structure and requirements. In *Proceedings of Digital Libraries'94*, 1994.

[Gon97] M. A. Gonçalves. Use of hypermedia models in digital libraries for geographic data. Master's thesis, Instituto de Computação – Unicamp, December 1997. In portuguese.

[Goo92] M. Goodchild et al. Integrating GIS and Spatial Data Analysis: Problems and Possibilities. *International Journal of Geographical Information Systems*, 6(5):407–424, 1992.

[Gre85] M. Green. Report on Dialogue Specification Tools. In G. E. Pfaff, editor, *User Interface Management Systems*, pages 9–20. Springer-Verlag, 1985.

[GT96] K. Gronbaek and R. Trigg. Toward a dexter-based model for open hypermedia: Unifying embedded references and link objects. In *Proceedings of Hypertext '96*, Washington, D.C., March 16-20 1996.

[Har87] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[HS89] W. Hurley and J. Sibert. Modelling User Interface–Application Interactions. *IEEE Software*, 6(1):77–77, 1989.

[KP88] G. Krasner and S. Pope. A Cookbook for Using the MVC User Interface Paradigm in Smalltalk. *Journal of Object-Oriented Programming*, 1(3):26–49, 1988.

[LR93] J. Lagrange and A. Ruas. Etat de l'art en generalization. Technical report, IGN/DT/SR/COGIT, April 1993.

[MMM+97] B. Myers, R. McDaniel, R. Miller, A. Faulring, B. Kyle, A. Mickish, A. Klimovitski, and P. Doane. The Amulet Environment: New Models for Effective User Interface Software Development. *IEEE Transactions on Software Engineering*, 23(6):347–365, June 1997.

[MR92] B. Myers and M. Rosson. Survey on User Interface Programming. In *Proc. Human Factors in Computing Systems*, pages 195–202, May 1992.

[MvD91] A. Marcus and A. van Dam. User-Interface Developments for the Nineties. *IEEE Computer*, 24(9):49–57, September 1991.

[Mye95] B. Myers. User Interface Software Tools. *ACM Transactions on Computer-Human Interaction*, 2(1):64–103, March 1995.

[NMK91] H. Nakatsuyama, M. Murata, and K. Kusumoto. A New Framework for Separating User Interfaces from Application Programs. *SIGCHI Bulletin*, 23(1):88–91, 1991.

[OCM95] J. L. Oliveira, C. Q. Cunha, and G. C. Magalhães. An Object Model for Dynamic Construction of Visual Interfaces. In *Proc. 9th Brazilian Symposium on Software Engineering*, pages 143–158, October 1995. In portuguese.

[Oli97] J. L. Oliveira. *Design and Implementation of User Interfaces for Geographic Applications.* PhD thesis, Instituto de Computação – Unicamp, December 1997. In portuguese.

[OM95] J. L. Oliveira and C. B. Medeiros. A Direct Manipulation User Interface for Querying Geographic Databases. In *Proc. 2nd International Conference on Applications of Databases*, pages 249–258, December 1995.

[OM96] J. L. Oliveira and C. B. Medeiros. User Interface Architectures, Languages, and Models in Geographic Databases. Tutorial. In *Proc. 11th Brazilian Symposium on Databases*, pages 20–42, October 1996.

[OMC97] J. L. Oliveira, C. B. Medeiros, and M. A. Cilia. Active Customization of GIS User Interfaces. In *Proc. IEEE International Conference on Data Engineering*, pages 487–496, April 1997.

[Pir97] F. Pires. *A Computational Environment for Modeling Geographic Applications.* PhD thesis, Instituto de Computação – Unicamp, December 1997. In portuguese.

[Rig95] P. Rigaux. *Interfaces Graphiques pour Bases de Données Spatiales: Application à la Représentation Multiple.* PhD thesis, CEDRIC - Conservatoire National des Arts et Metiers, 1995.

[Ros96] G. Rossi. *An Object-Oriented Model for Designing Hypermedia Applications.* PhD thesis, Departamento de Informática – PUC-RJ, 1996. In portuguese.

[RSLC95] G. Rossi, D. Schwabe, C. J. P. Lucena, and D. D. Cowan. An object-oriented model for designing the human-computer interface of hypermedia applications. In *International Workshop on Hypermedia Design*, Montpellier, France, 1995.

[SCB95] L. Shklar, C. St. Charles, and C. Behrens. Geoscope detailed design. Technical report, Bellcore - Bell Communications Research, September 1995.

[SRB96] D. Schwabe, G. Rossi, and S. D. J. Barbosa. Systematic hypermedia application design with oohdm. In *Proceedings of ACM Conference on Hypertext (Hypertext'96)*, Washington, DC, march 1996.

[Val89] J. Valdez. XVT, a Virtual Toolkit. *Byte*, 14(3), 1989.

[WGMD95] S. Weibel, J. Godby, E. Miller, and R. Daniel. OCLC/NCSA metadata workshop report. http://www.oclc.org:5046/oclc/research/ conferences/metadata/dublin_core_report.html, 1995.

[Wil96] R. Wilensky. Toward work-centered digital information services. *IEEE Computer Magazine*, 29(5), May 1996.