A Software Architecture Framework for Geographic User Interfaces

Juliano Lopes de Oliveira[1] and Claudia Bauzer Medeiros[2]

[1] Instituto de Informática - UFG
CP 131 - Goiânia-GO 74001-970 Brazil juliano@inf.ufg.br
[2] Instituto de Computação - Unicamp
CP 6176 - Campinas-SP 13081-970 Brazil cmbm@dcc.unicamp.br

Abstract. This work presents an architectural framework to support the design and implementation of user interfaces for geographic applications. It adopts a pragmatic and innovative perspective, considering two key aspects: dialogue with the user and connection with the underlying supporting software (DBMS, interface toolkits and others). The framework covers both the construction of the interface and the mechanisms for its run-time execution, which are based on object-oriented databases and on interoperability concepts. It was applied to the development of interfaces for two large geographic application systems, contributing to reduce their costs and to improve the software development process.
Keywords: **Geographic Application; Software Architecture; User Interface Models for Geographic Applications**

1 Introduction

Geographic Information Systems (GIS) are complex software systems whose main goal is to deal with *geo-referenced* information, stored in a geographic database. This kind of information has three main components: a geographic position; conventional (descriptive) attributes; and the associated time validity frame. There are countless applications for GIS, ranging from urban planning to environmental monitoring. A considerable part of the cost in developing GIS applications is spent with user interface issues [MeHe93,Espr98].

Geo-referenced data are difficult to present and to manipulate, requiring a combination of graphical and textual representations and complex transformations of data from the storage level of the GIS geographic database to the user and vice-versa. Moreover, GIS's users are, in general, experts on many technical areas, but *not* in Computer Science. Therefore, besides being easy to use, the interface must be customizable to the user's needs.

Due to this complexity, inherent to the construction of user interfaces for geographic applications, no appropriate architectural framework has been proposed to give full support to this process. In fact, most systems use an *ad hoc* approach, based only on generic user interface tools and empirical guidelines. We claim that this approach is not adequate, and that to effectively reduce the costs of these interfaces it is necessary to adopt a systematic development approach.

This paper describes such an approach, based on a *domain specific software architecture.*

The term "architecture", when applied in software engineering, is usually considered in an oversimplified manner. For this reason, it often plays a secondary role in the definition of applications and systems. In fact, a *Software Architecture* is more than a diagram with a few connected function boxes; it is an abstraction that help designers to deal with the inherent complexity of software, involving not only the description of elements from which systems are built, but also the interactions among those elements, the patterns that guide their composition, the constraints on these patterns, and the complete specification of run-time behavior [ShGa96,BCK98].

This paper describes a new software architecture, which provides a generic framework for the development of user interfaces for geographic applications. This is based on separating the core of the application (Semantic Component) from its interface needs (Interactive Component). This architecture combines two types of approach: interface specification and construction are based on work developed by the interface software community, whereas modularity and module interoperability within the interface are provided by using a database-centered approach.

A key feature of this architecture is that it supports the development of the user interface of the geographic application through all stages of its life cycle (from requirements through analysis, design and implementation). To accomplish this, it establishes a trade-off between abstract specification, necessary for the earlier stages, and detailed specification, necessary for implementation. This allows distinct types of geographic applications to be developed using the same interface facilities.

Experiences on using this framework to implement geographic applications ([OCM95,OPM97,SPMO98]) have shown a significant improvement on the application modularity, and a consequent reduction of the complexity and of the inter-dependency between the interface code and the code of the geographic application itself. Systems developed according to this approach are easier to mantain and to evolve, and benefit from the reutilization of many user interface components. A prototype of this framework has already been implemented [SPMO98], and some of its features have been incorporated into a large geographic application [OCM95].

This paper is not concerned with implementation details, and concentrates on presenting the main ideas behind the framework. Section 2 discusses the limitations of current approaches and architectures for GIS user interface construction. Section 3 defines the principles of our new architecture. Section 4 specifies constraints and guidelines to orient the software designer in the use of the architecture. Section 5 presents conclusions and extensions.

2 Limitations of Current Approaches

A *Geographic Application* is an interactive program developed on top of a GIS and that manipulates both geo-referenced and conventional data using the facilities of the underlying GIS. Examples of such applications are, for instance, several types of spatial decision support systems (e.g., [MMP98,Gunther98,Mann96]) which allow users to cartographically visualize and interact with different scenarios for facility placement, transportation, environmental control and so on. Other such applications include GIS-based systems for urban planning or utility management. As any interactive program, a geographic application can be logically decomposed in two main parts: *Semantic Component*, which defines the functionality and the semantics of the application, and *Interactive Component*, responsible for the dialog with the user [Edmo92].

Two types of functions are available to the user of a geographic application. The *interface functions* are processed by the Interactive Component of the application. The *semantic functions* deal with complex semantic transformations which depend on the Semantic Component of the application. For instance, a geographic application dealing with transportation networks requires the computation of several path functions (e.g., the minimal path between two points), which can be displayed in distinct ways. The computation is performed by the Semantic Component, whereas the display is the responsibility of the Interactive Component. Ideally, the Interactive Component plays two roles: it provides interface functions and it mediates the communication between the user and the application's Semantic Component. Figure 1 shows this high level view of a geographic application.

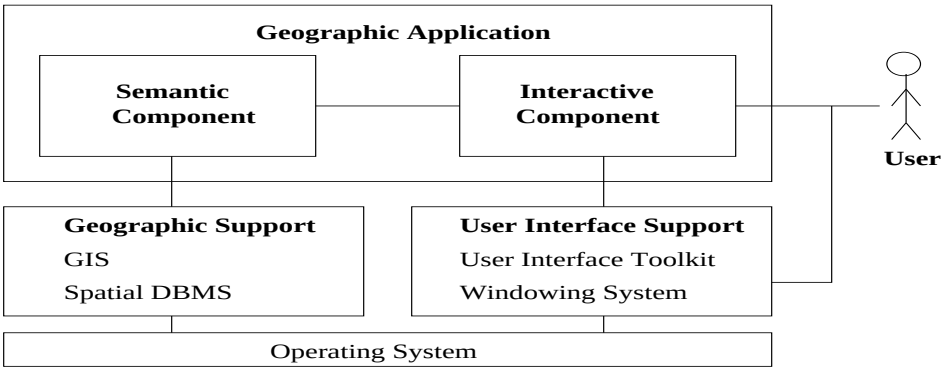

Fig. 1. General context of a Geographic Application

The separation of the Interactive and Semantic components of an application improves the specialization and the optimization of each component. It also makes it easier to construct customized interfaces for the application, if the data

and functions are properly encapsulated. Most user interface software architectures recognize these benefits and adopt this separation as a design principle. However, most of these architectures define the Semantic and Interactive components from a very abstract level, which, from a practical viewpoint, does not help the design and the implementation of the application. Furthermore, this separation is hardly ever found in applications, because of the shortcomings of the underlying GIS.

Current architectures for user interface software can be classified as *Modular* architectures (e.g, the Seeheim architecture [Gree85] and Slinky/Arch [BFL+92]) or as *Agent Based* architectures (e.g, MVC [KrPo88] and PAC [BaCo91]). Modular architectures are imprecise on the definition of the communication protocols between modules, while Agent Based architectures do not define rules for the composition of each agent of the system. These architectures offer good guidelines for generic user interface design, but are not concerned with and cannot deal with the idiossyncrasies of geographic application user interfaces.

Besides the inherent difficulties for presenting and manipulating geo-referenced data, geographic applications need to integrate heterogeneous supporting systems (GIS and user interface toolkits, for instance). Generic architectures are not appropriate for this context because:

- they do not take into account the particularities of geographic applications. For instance, generic architectures deal with *multiple representation* of objects only in the presentation level, but geo-referenced data have multiple representations also in the data level [Riga95];
- They do not fulfill the organizational requirements of geographic software. In particular, generic user interface architectures do not take into account the integration of the application with heterogeneous underlying software. These architectures cope only with user interface packages, but not with GIS.
- They do not guide the transition from the logical to the physical design of the application. Their goal is to be generic, so they cannot take into account the details of specific applications.

The limitations of generic user interface architectures were recognized by the researchers in the GIS community. Some ad hoc solutions have been proposed to build geographic user interfaces for specific GIS (e.g., [Esri92,AYA+92]). Other solutions were GIS independent, but they impose rigid functional constraints either on the GIS or on the Interactive Component of the geographic application:

1. Constraints on the GIS: the architecture can only be used with GIS that offer specific services, such as a particular type of manipulation of geo-referenced data (e.g, the map legend dictionary of [Vois94]) or specific integration approaches (e.g, the interface as a layer of the GIS architecture itself [PMP93]).
2. Constraints on the Interactive Component: this type of solution is based on limiting the set of functions available to the user in the application. The idea is to minimize the interaction between the geographic application and the GIS by reducing the set of functions available in this interaction. The

complexity is also simplified by eliminating functions with side effects in the stored data. In general, only query functions are allowed in this approach (e.g, [Riga95,OlMe95]).

The remainder of this paper presents a new user interface software architecture for geographic applications. This architecture solves the limitations we have just discussed. Similar to the previous architectures, it separates the user interface software in abstract and inter-related sets of components. However, it also specifies how to support the design and implementation of the application, by indicating how to refine the components up to the implementation level. Moreover, this architecture focuses on the exclusive features of geographic applications.

Again, looking at Figure 1, we point out that it gives a high level view of concepts which we use throughout the text, separating an application from the *support software* (GIS, spatial DBMS, interface toolkits, etc). Our framework is based in establishing appropriate communication protocols between application and support layers, thereby helping the design and development of the Interactive Component of a geographic application.

3 Architecture Overview

The main goal of our architecture is to guide the design and the implementation of geographic user interfaces, that is, of the Interactive Component of geographic applications. To reach this goal, the architecture has to define, among other aspects:

- design guidelines to aid the designer on the construction of a geographic interface;
- an overall run-time behavior for the architecture, specifying the protocols and the rules for interaction among its software components;
- a communication protocol between the Semantic and the Interactive components of the geographic application, maximizing their mutual independence;
- additional modules to support the development of the interface, producing services that can be reused by different geographic applications;
- additional modules to deal with the integration with the underlying GIS, enhancing the portability of the geographic application.

Our key idea to support interoperability and modularization borrows from database research. It is based on construction of an intermediate layer (the Data Model Layer) between the application and the supporting software. Data used by an application is not brought directly to the interface's working memory. Rather, it is "imported upon request" into a reserved data space constructed according to the GMOD geographic data model [OPM97]. The GMOD data model is what interface literature calls the *interface intermediate model* – i.e., a model closer to the user and which can be at the same time manipulated by the interface functions. Similarly, another reserved data space contains templates for constructing interface objects. These templates are built according to the IMOD

geographic interface model. These templates correspond to what interface literature calls *interface object models.* A given data element defined using GMOD is formatted according to a IMOD template in order to be presented to the user.

3.1 Main Components

The architecture has three main layers. Each layer is responsible for the management of a well defined set of tasks, providing services to the other layers. The main tasks performed by this framework include:

- communication with the underlying software (geographic information system and user interface tools) - Connection Layer, at the bottom;
- mapping geographic and interface data models - the intermediate Data Model Layer; and
- supporting the implementation of the geographic application, separating interface from semantic functions – the Application Layer.

Connection Layer The Connection Layer handles the communication with the underlying supporting systems. Among these systems, two are fundamental for a geographic application: the GIS and the user interface toolkit.

The Adaptor Modules of the Connection Layer are responsible for offering to the upper layers a standard access to the supporting systems, regardless of the specific features of these systems. An adaptor has three main objectives:

- To guarantee portability, playing the role of a "device driver" for each underlying system. This means that the geographic application should be able to execute on any system that can be connected to the adaptor module.
- To improve software reuse: this happens in the global context of the architecture since all geographic applications built according to this framework can share the same adaptors.
- To support maintainability and software evolution. Maintainability is a consequence of the encapsulation of the supporting systems. In fact, any modification on the supporting software affects only the adaptor module (in particular its mapping model) and has no side effect on the rest of the architecture.

An adaptor module defines, basically, an abstract machine with an uniform application programming interface. Therefore, the upper layers use the same communication interface regardless of the underlying system is. There are specific adaptors to the geographic and to the interface supporting systems.

GIS Adaptor The GIS adaptor is based on the interface integration mechanisms presented in [OlAn93]. The main idea is to establish a standard protocol between the adaptor and the underlying GIS, based on a set of primitive operations defined on the intermediate data model.

It performs the translation of schemata, data and operations from the intermediate model (GMOD) to the data model of the GIS, and vice-versa. Communication with the GIS can be performed via its application programming interface (API), or directly with its data manager modules, if it does not define an API. This poses the problem of strong integration (the interface code is mixed with the GIS code) versus weak integration.

Our approach presents a trade-off solution to accomodate both types of integration: only one module of the interface code is dependent on the underlying GIS. The degree of dependency varies according with the GIS: with *open* GIS the dependency is limited to its API, while in other cases part of the GIS code has to be directly controled by the interface. The key point here is that the GIS adaptor offers a standard API to the upper layers of the architecture (in the figure, GAPI- Geographic Application Programming Interface). All the geographic applications use a single communication protocol, and the GIS adaptor translates this protocol to the specific communication mechanism of the adopted GIS.

User Interface Toolkit Adaptor The User Interface Toolkit Adaptor Module allows the definition of interface and interaction objects according to a generic user interface object model (IMOD). This model contains definitions for most common types of interaction objects provided by the main existing user interface toolkits.

The adaptor implements mapping rules from toolkit's widgets to the interaction objects in IMOD. These rules defines the correspondence between the specific toolkit API and the standard API provided by the Toolkit Adaptor (TAPI - Toolkit Application Programming Interface).

The main advantage of the standard API provided by the adaptor is the independence from a specific user interface toolkit. The same idea has been used in the generic user interface software architecture presented in [BFL+92], and also in the implementation of multi-platform user interface toolkits, such as [MMM+97].

Data Model Layer This is the core layer, responsible for maintaining two independent data spaces, treated from a database perspective. The GMOD database contains application-related data; the IMOD database contains interface objects and primitives.

More specifically, the GMOD database is constructed to support the GMOD geographic data model [OPM97], an object oriented data model which offers primitives for designing geographic applications. It contains the data exchanged between the Interactive and the Semantic components of the geographic application. The IMOD database supports the geographic user interface, containing the data exchanged between the user and the geographic application.

Each database has its own data model and is associated with a database server that manages the access to the respective data. Although they are logically presented in the Data Model Layer of the architecture, these database servers can be implemented directly on the underlying geographic DBMS. In an existing implementation of our framework, they are actually stored under a single DBMS [SPMO98].

Using GMOD The GMOD database contains the data requested by the application's Semantic component, and which are manipulated in the Interactive component. Both the Semantic and the Interactive components can access the data managed by the GMOD server through a Semantic-Interactive Adaptor Module. This adaptor defines the main aspects of the interaction between the two components of the geographic application, such as the abstraction level of the transferred data, the mechanism for describing the exchanged data, and the allowed access mode to the data.

The Semantic and Interactive components communicate with the adaptor via two specific APIs: the ISPI - Interface Services Programming Interface; and the SSPI - Semantic Services Programming Interface.

Using IMOD The second data space managed by the Data Model Layer is the interface objects model (IMOD), which provides facilities to create different presentations for the data in the GMOD database. The IMOD database contains the definitions of the interface objects used in the Interactive Component of the geographic application. This component requests these objects from the lower lower layers using the ICPI - Interface object Constructor Programming - which communicates with the Interface Object Constructor module.

This latter module manages the creation of complex interface objects by repeatedly composing basic interaction widgets – e.g., buttons, windows – and templates. Widgets and templates are retrieved from the IMOD database and are used to create high level specifications of the interface. These specifications are sent to the Toolkit Adaptor via the TAPI in order to build actual interface objects using the underlying user interface toolkit. In [Oliv97] there is a complete description of the IMOD data model and examples of using this model for building dynamic geographic user interfaces.

Application Layer The two main modules of the Application Layer are the Semantic and Interactive components of the geographic application.

Interactive Component This component is responsible for the dialog with the end user and has two main modules: Dialog Control and Presentation Control.

The Presentation Control Module performs two fundamental tasks: (1) the translation of user actions (ie, user events) into operations on the interface objects which are stored in the IMOD database; (2) the management of graphical and textual presentations contained in this database.

The Dialog Control Module keeps track of the interaction between application objects and interface objects. It is responsible for the dynamic behavior of the user interface, including the management of the Presentation Control Module. For instance, an interaction object can be associated to more than one application object defined in the GMOD database.

The Interactive Component of the geographic application represents the part of our software architecture that must take into account the complex problems of presenting and manipulating geographic data. Due to space limitations, we do

not discuss here the details of this component. Our solutions for these problems are described in depth in [Oliv97].

Semantic Component As we have already discussed, the semantic and interactive components of the geographic application need to exchange data and events and, therefore, it is necessary to take the semantic component into account in the definition of the user interface architecture. Nevertheless, neither the semantic component nor the GIS should be part of the user interface software. Thus, in an ideal context, the user interface should impose no constraints on these modules.

In a real software context, however, this complete independence is not possible. Recognizing this, we tried to limit the mutual interference between the user interface and the other components of the geographic software. In fact, the only constraints we have imposed on the GIS and on the Semantic component of the application are those that specify their communication with the user interface (the Interactive component of the architecture). This autonomy of the user interface with regard to the rest of the software is an important and desirable feature, which is known in the literature as *dialog independence* [HaHi89].

4 Design Guidelines

The layered organization of our architecture establishes a client/server relationship between one layer and the underlying layers. This structure forms the basis for the development of new geographic interfaces according to specific guidelines that determine (a) the components that need to be redefined in new projects, and (b) the reusable components that are already available. These guidelines define three main stages for the development of a geographic interface environment, as we show in the following example.

1. Construction of the Connection Layer;
2. Construction of the Data Model Layer;
3. Construction of the Application Layer.

4.1 Example of Interaction

Before discussing the process of constructing each layer, we must have a clear view of the run-time interactions among these layers. The example we show here illustrates, in a schematic way, a typical interaction of a user with a geographic application built according to our architecture. The objective of this example is to show the exchanges of data and control and the interdependencies among the componentes of the architecture.

Consider a geographic application that allows the user to query and update geo-referenced information by selecting (e.g., with a mouse) relevant objects in a map presented in a interface window. We emphasize that in order for an object to be presented in the interface two steps must be followed: (1) first, it must be retrieved into the GMOD database by an application request; and (2) a presentation function must be applied to it, using IMOD templates.

Let us assume the user is working on a telecommunications cadastral problem, and that the geographic objects manipulated are parcels, streets and utility equipment (poles, cables, transformers). Data and control flow in the framework are as follows.

- The user activates an operation by selecting some objects presented in the interface window. The Presentation Control Module captures the user request and communicates to the Dialog Control Module the occurrence of a user event. For instance, it can translate a double click event on a parcel into a select operation over the graphical representation of the parcel.
- The Dialog Control Module manages the relationships between interface objects (IMOD database) and application data objects (GMOD database). This module also keeps track of the interaction context, i.e.,the state of the interface, allowing the identification of the sequence of operations to be performed in response to the user action. In this example, the Dialog Control knows that the selection of a parcel in the current context activates the following operations:

  1. A request to the Semantic-Interface Adaptor (via the SSPI) for querying the current state of the parcel.
     The Adaptor can directly perform this operation, since the data are already in the GMOD database (because the parcel was already presented in the interface map window).
     NB: If this had not been the case, the Adaptor would have had to forward the query to the Semantic Component (via ISPI), and the latter would have to ask the GIS Adaptor for the requested data (via GAPI). This Adaptor would have to translate the query to the corresponding command in the underlying GIS. The result of the query would, then, be sent in the reverse path (GIS → GIS Adaptor → Semantic component → Semantic-Interface Adaptor → GMOD database).
  2. A request for presentation of the current state of the object to the Interface Constructor (via ICPI).
     The Constructor receives from the Dialog Control a description of the type of presentation (in this case, an edition window) and the data values corresponding to the current state of the object. Using these informations, the Constructor searches the IMOD database to find an interface object template corresponding to the requested type of window. Based on this template, the Constructor builds a virtual composite window which is sent to the Toolkit Adaptor (via TAPI).
     The Toolkit Adaptor translates the definitions of the window to the API of the underlying toolkit, creating the real widgets of the composite window. These are next sent via the reverse path to the Dialog Control and Presentation Control modules, which manage the interface windows.
  3. An update of the state of the interface, indicating the presence of the new window to be presented, and setting up possible actions over this presentation.

The Dialog Control is responsible for maintaining the interface context. In each state many different types of functions may be available, and many different sets of interactive objects may be active. In the example, it defines the necessary modifications to represent the state resulting from creation of the new window.

- The user performs the desired modifications on this presentation of the parcel and confirms the operation. Again, the user event is translated by the Presentation Control into a call to the Dialog Control, which is responsible for transforming this operation into a update request in the GMOD database, via Semantic-Interface Adaptor.
- The Adaptor performs semantic validation of the requested update, according to the constraints expressed in the GMOD schema. It can also activate the Semantic component for further semantic validation. If an error occurs, the Dialog Control is warned; otherwise, the Adaptor processes the request in the following way:
  1. Updates the state of the parcel in the GMOD database;
  2. Signals to the Semantic component the update has been performed.
- The Semantic component takes the action necessary to guarantee that the modification be reflected in its internal structures as well as in the GIS (via GAPI).

The design guidelines, indicated in sections 4.2 through 4.4, take into account these control and data flows, in order to obtain software modules that are generic enough to be reused and, at the same time, efficient in their tasks.

4.2 Building the Connection Layer

The Connection Layer is the only part of the architecture that depends on features of the supporting softwares, i.e., the softwares that provide run-time services for the geographic application. The two modules of the Connection Layer are completely independent of each other. Each module needs to be defined once for each underlying support system, specifying the conversion mechanisms from the Adaptor's generic API to the real API of the underlying system. All the upper layers use this standard API provided by the Adaptor and should not be affected by changes in the underlying support systems.

A different GIS Adaptor must be built for each GIS, to provide a mapping model between the GIS data model and the GMOD data model, due to the variety of geographic models existing in current GIS.

The Toolkit Adaptor, on the other hand, works with different types of syntax to express the same concepts (e.g, widgets, events, and callback functions). However, the implementation of this Adaptor is also difficult, since it has to ensure, for instance, a standard look-and-feel for the geographic application with different underlying toolkits.

It is expensive to design and implement the modules of the Connection Layer, but it is worth the effort, since this layer can be shared by all the geographic applications that use the same underlying systems. This layer can be omitted from

the architecture, as a result from a design decision: for instance, if portability is not a major requirement, or if there is no possibility of changing the underlying systems, the Adaptor's API's are replaced by those of the underlying softwares.

4.3 Building the Data Model Layer

The Data Model Layer provides modules to support both the design and the runtime execution of the geographic interface. These modules are handled by the designer in the development of new interfaces, and by the components of an application, to perform its run-time functions. The layer is shared by all geographic aplications in a given site.

The services offered by this layer allow the management of the two databases which are fundamental for our architecture: the geographic database containing data used by the application using the interface intermediate model (GMOD) and the interface objects database (IMOD). The Data Model Layer is composed by two orthogonal modules: the Semantic-Interface Adaptor, responsible for all exchange of information between the two main components of the geographic application, and the Interface Constructor, which manages the interface objects used in the dialog with the user.

The communication between the Semantic and the Interactive components of the geographic application is mediated by the Semantic-Interface Adaptor, which uses data stored in the GMOD database. This Adaptor allows the two components of the application to share these data.

The task of the Interface Constructor Module is to manage complex interface templates, i.e, definitions of virtual complex widgets. To achieve this goal, the Constructor must keep a library of specification of interface objects, organized according to the IMOD interface data model; each application may use different templates from this library.

4.4 Building the Application Layer

The Interactive component of a geographic application is implemented in the Application Layer. The two main modules of this layer, Dialog Control and Presentation Control, reflect the user mental model as it is perceived by the application interface designer. The user mental model involves, for instance, the definition of the reaction an interface must provide to a given user action.

It is necessary to define the Interactive component once for each different geographic application, since this component is completely dependent of the specific goals of each application. It is worth noting, however, that more than one instance of this component may be created for the same Semantic component. This allows, for instance, to provide customized interfaces to heterogeneous groups of users of a single application. For instance, the same application can be used by technical and administrative users. Methods for customization and reuse of interfaces can reduce the cost of development of these alternative interfaces for the same application. In [OMC97] we show examples and applications of these methods.

4.5 The Interface Development Process

The directives discussed guide the development process. Although the Data Model and the Connection layers are complex, their costs are amortized by their reuse in a potentially large number of geographic applications.

The Application Layer, in turn, is relatively easier to develop in term of interface functions because it can count on the services of the two other (supporting) layers. However, the development cost of the application layer cannot be underestimated, because this layer deals with complex problems related to the interactive manipulation of geo-referenced data. In [Oliv97] we present methods and tools for reducing the effort necessary to build the application layer of the geographic application.

Our experience in the development of geographic interfaces according to the approach described in this paper has shown considerable improvement of the development process. Besides promoting modularization and specialization of the components, the architecture makes it possible to reuse all the components that deal with the organization of data and interaction with the underlying systems. This reusable framework is an important by-product of our architecture. In traditional approaches, all these components have to be completely defined, in an ad hoc way, to each and every geographic application.

5 Conclusions

The design and implementation of geographic applications is a complex task. The user interface component of the geographic application contributes with a major part of this complexity, since for the user the interface *is* the application. This paper presented a software architecture aimed at reducing this problem. The main goal of the architecture is to support the design, implementation and execution of the Interactive component of the user interface of a geographic application.

The existing frameworks to support user interface development and execution can be classified in four categories [Myer95]: predefined application structures (i.e, domain specific architectures); model based interface generation tools; interface definition languages; and interactive interface editors. Our framework is a variation of the first category, since it specifically focuses on interfaces for geographic applications. We also adopted ideas from the second category in our mechanism for constructing complex interface objects from virtual (toolkit independent) models.

Each layer is responsible for managing a well defined set of tasks: communication with the underlying software (Connection Layer); data model operations (Data Model Layer); and supporting the implementation of the geographic application, i.e, of the semantic and interaction functions (Application Layer).

There are four main advantages to this framework in comparison with previous architectures presented in the literature. First, it supports the construction of both static and dynamic user interfaces, i.e, interfaces which can be redefined and modified at run-time. Second, it is generic, in the sense that it can be

applied to most of the existing supporting systems (both user interface development tools and geographic information systems). Third, the framework supports the complete life cycle of a geographic user interface, from design (including model-based facilities for iteractive refinements) through implementation (based on a complete architectural specification) and execution (supported by run-time mechanisms defined on the framework). Finally, the software architecture supports geographic interface customization mechanisms.

We proposed a schema for reusing supporting components, promoting moreover the independence and the specialization of components. In particular, we presented a well defined separation between the interactive component and the semantic component of the geographic application, obtaining the so called *dialog independence*. Furthermore, following the ideas of [GHK+96], we adopted a database solution to guarantee data independence for the geographic interface with regard to the underlying geographic system. Thanks to this approach, our architecture allows the development of generic geographic user interface software (and not just for querying a geographic database).

The paper presented only a general overview of the architecture and its communication protocols, mechanisms, components, and design guidelines. An in depth discussion of all these aspects or of implementation details was not possible here, due to space limitations. All these details can be found in [Oliv97], while examples of use of (parts of) our framework can be found in [OlMe95,OCM95], [OMC97,OPM97,SPMO98].

Acknowledgements

This work was developed with financial help of CNPq, FAPESP and MCT/FINEP PRONEX II program 76.97.1022.00 SAI (Advanced Information Systems), as well as a joint CNPq/NSF project on GIS interoperability.

References

[AYA+92] D. Abel, S. Yap, R. Ackland, et al. Environmental Decision Support System Project: an Exploration of Alternative Architectures for Geographical Information Systems. In *International Journal of GIS, 3(6):193–204*, 1992.

[BaCo91] L. Bass and J. Coutaz. Developing Software for the User Interface. Addison-Wesley, 1991.

[BCK98] L. Bass, P. Clements and R. Kazman. Software Architecture in Practice. Addison Wesley, USA, 1998.

[BFL+92] L. Bass, R. Faneuf, R. Little, et al. A Metamodel for the Runtime Architecture of an Interactive System. In *SIGCHI Bulletin*, 24(1):32–37, Jan. 1992.

[Edmo92] E. Edmonds. The Emergence of the Separable User Interface. In *The Separable User Interface*, pages 5-18. Academic Press, 1992.

[Espr98] ESPRIT/ESSI Project no.21580. Guidelines For Best Practice In User Interface For GIS. GISIG - Geographical Information Systems International Group, Italy, 1998.

[Esri92] Environmental Systems Research Institute. Arc-Info: GIS today and tomorrow, 1992.

[Gunther98] O. Gunther. *Environmental Information Systems*. Springer Verlag, 1998

[GHK+96] N. Gopal, C. Hoch, R. Krishnamurthy, et al. Is GUI Programming a Database Research Problem? In *Proc ACM SIGMOD*, 517–528, 1996.

[Gree85] M. Green. Report on Dialogue Specification Tools. In *User Interface Management Systems*, pages 9-20. Springer-Verlag, 1985.

[HaHi89] H. Hartson and D. Hix. Human–Computer Interface Development: Concepts and Systems for its Management. In *ACM Computing Surveys*, 21(1):5–92, 1989.

[KrPo88] G. Krasner and S. Pope. A Cookbook for Using the MVC User Interface Paradigm in Smalltalk. In *Journal of Object-Oriented Programming*, 3(1):26–49, 1988.

[Mann96] S. Mann. Spatial Process Modelling for Regional Environmental Decision Making. In *Proc. 8th Annual Colloquium of Systems Information Research Centre*, 126-135, 1996.

[MeHe93] D. Medyckyj-Scott and H. M. Hearnshaw, editors. Human Factors In Geographical Information System. Belhaven Press, 1993.

[MMM+97] B. Myers, R. McDaniel, R. Miller et al. The Amulet Environment: New Models for Effective Interface Software Development. In *Trans. on Software Engineering*, 23(6):347–365, 1997.

[MMP98] V. Maniezzo, I. Mendes and M. Paruccini. Decision Support for Siting Problems. Decision Support Systems, 23:273-284, 1998

[Myer95] B. Myers. User Interface Software Tools. In *ACM Trans. on Computer-Human Interaction*, 2(1):64–103, Mar. 1995.

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

[OlAn93] J. L. Oliveira and R. Anido. Integrating a Browsing Interface to Different Object-Oriented Database Systems. In *Proc. 13th Brazilian Computer Society Conference*, pages 61–75, 1993 (in portuguese).

[Oliv97] J. L. Oliveira. Design and Implementation of Interfaces for Geographic Application Systems. PhD thesis, IC – Unicamp, Dez. 1997 (in portuguese).

[OlMe95] J. L. Oliveira and C. B. Medeiros. A Direct Manipulation User Interface for Querying Geographic Databases. In *Int. ADB Conf.*, 249–258, 1995.

[OMC97] J. L. Oliveira, C. B. Medeiros, and M. A. Cilia. Active Customization of GIS User Interfaces. In *IEEE ICDE Conference*, 487–496, 1997.

[OPM97] J. L. Oliveira, F. Pires, and C. B. Medeiros. An environment for modeling and design of geographic applications. In *GeoInformatica*, 1(1):29–58, 1997.

[PMP93] N. Pissinou, K. Makki, and E. Park. Towards the Design and Development of a New Architecture for GIS . In *Int. CIKM Conference*, 565–573, 1993.

[Riga95] P. Rigaux. *Interfaces Graphiques pour Bases de Données Spatiales: Application à la Représentation Multiple*. PhD thesis, CEDRIC - CNAM, 1995.

[ShGa96] M. Shaw and D. Garlan. Software Architecture - Perspectives On An Emerging Discipline. Prentice Hall, USA, 1996.

[SPMO98] M. A. Salles, F. Pires, C. B. Medeiros and J. L. Oliveira. Development of a Computer Aided Geographic Database Design System. In *Proc. 13th Brazilian Symposium on Databases*, 235–250, 1998.

[Vois94] A. Voisard. Designing and Integrating User Interfaces of Geographic Database Applications. In *ACM AVI Workshop*, 133–142, 1994.

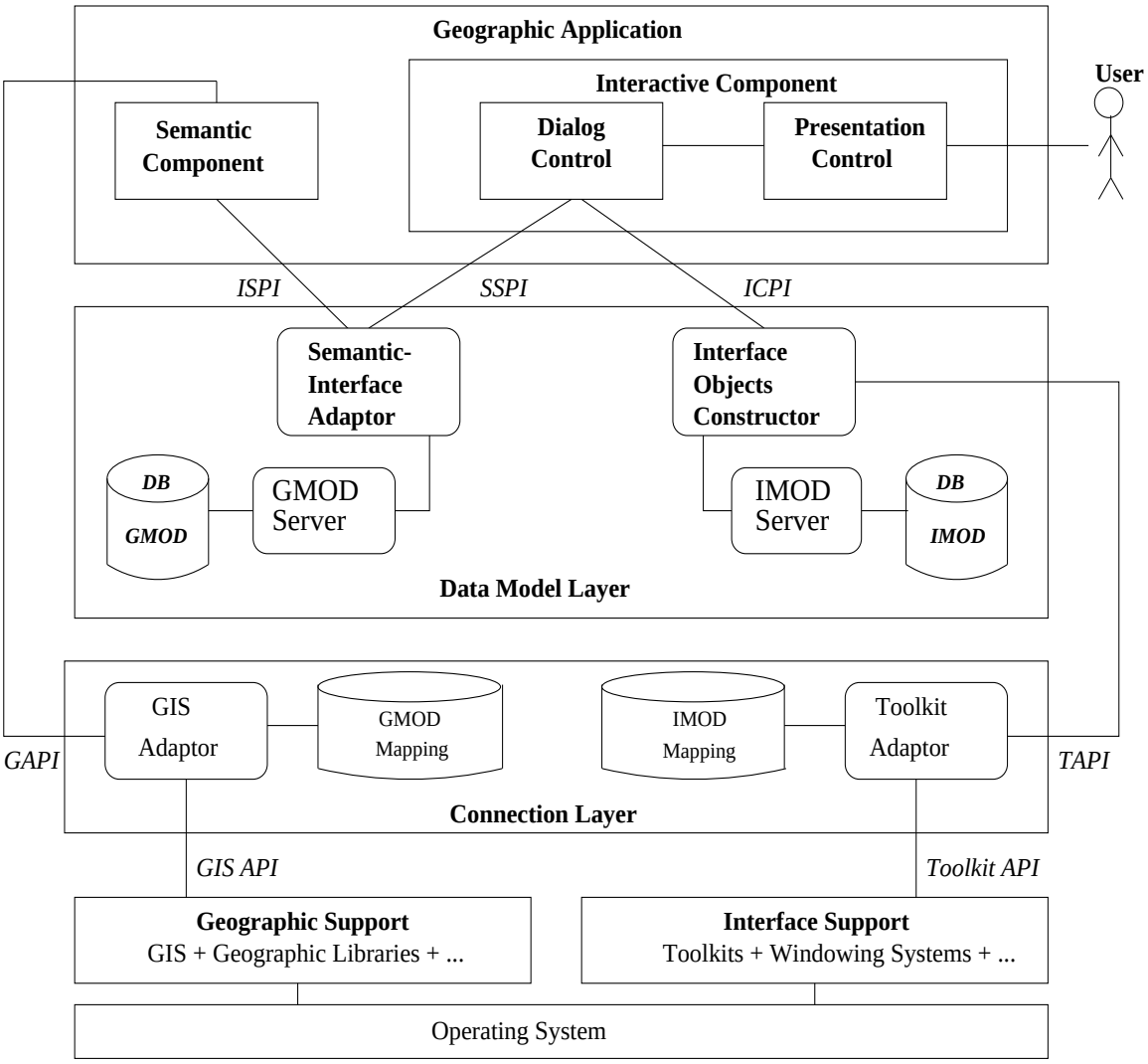

Fig. 2. Main aspects of the geographic user interface software architecture