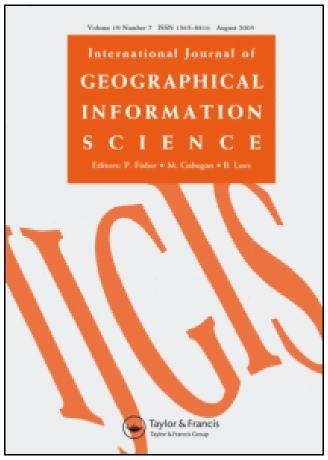

International Journal of Geographical Information Science
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713599799

Object-oriented approaches to geo-referenced information

Michael F. Worboys [a]
[a] Department of Computer Science, Keele University, England, U.K.

Online Publication Date: 01 July 1994
To cite this Article: Worboys, Michael F. (1994) 'Object-oriented approaches to geo-referenced information', International Journal of Geographical Information Science, 8:4, 385 - 399
To link to this article: DOI: 10.1080/02693799408902008
URL: http://dx.doi.org/10.1080/02693799408902008

INT. J. GEOGRAPHICAL INFORMATION SYSTEMS, 1994, VOL. 8, NO. 4, 385–399

Review Article

Object-oriented approaches to geo-referenced information

MICHAEL F. WORBOYS
Department of Computer Science, Keele University,
Keele ST5 5BG, England U.K.

(*Received 11 November 1993; accepted 21 January 1994*)

Abstract. This paper surveys the current state of the object-oriented paradigm as it applies to the handling of geo-referenced information. The model of any computerized system is multi-layered, with a high-level system-independent conceptual model of the application domain supported by increasingly system-oriented models beneath. The author argues that object-oriented approaches can be taken at each of these layers. The major constructs of object-orientation are discussed from this layered viewpoint and in the context of geo-information handling.

1. Introduction

This paper is in a sense an update to an earlier paper (Worboys *et al.* 1990) in this journal. That paper reviewed the state of the object-oriented approach as observed by the authors in 1990. It was argued that such an approach was highly appropriate for geo-referenced information and the paper suggested how it might be applied, using the particular object-oriented model IFO (Abiteboul and Hull 1987, Hull and King 1987). One of the authors of the original paper has now the opportunity to provide an update, four years on. This paper will follow essentially the same structure as its predecessor, outlining the main concepts behind the object-oriented approach and its application to geo-referenced information handling. However, we will not restrict the discussion to any particular object modelling approach.

There are certain striking observations about the object-oriented (OO) field as it has developed over the last few years. In Worboys *et al.* (1990), we wrote that 'there is no clear definition of, or even general agreement in the computing community on what precisely is an object-oriented data model'. Current workers in the field are still making such statements. For example, Ling and Teo (1993) stated that 'although some attempts have been made to forge a common agreement on OO concepts, they have so far been unsuccessful'. On the other hand, a paper (Stonebraker *et al.* 1993), summarizing a recent panel discussion amongst world experts on future topics for information systems research, reported that 'another hostilely received topic was any additional object-oriented data models. The feeling was that lots of models have been invented, and lots more will presumably be discovered, all differing in minor ways'. So there are contradictory signals in this area. Perhaps, while we do not need yet more object-oriented data models, what we do need is standardization of the meanings of commonly used terms. For example, even the meaning of central notions, such as inheritance, can be quite varied.

There are two related causes of this proliferation:

- Lack of general agreement on what features an object-oriented system should possess.
- Lack of general agreement on the definition of each object-oriented feature (for example, what is meant by inheritance).

However, a body of research is emerging that seeks to unify the diverse concepts associated with object-orientation, notable amongst which is the work by Mattos *et al.* (1993).

Proprietary object-oriented systems have continued to develop in the intervening years. The programming language C++ is now well-established as a standard object-oriented language. However, given the amount of hyperbole in support of object-oriented systems, one might have expected a more dramatic upsurge in their use. Commercial object-oriented database management systems (OODBMS) include ONTOS, Gemstone from Servio Corporation, ObjectStore from Object Design Inc. and O_2 from O_2 Technology. However, most of these products have been around for six or seven years now and are still not in use in large-scale or mission-critical commercial applications. Kim (1993) estimates the current market for OODBMS at less than 1 per cent of the current total world market for database products.

Object-oriented approaches have been promoted especially in areas for which the application of pure relational technology has been found to be problematic. Examples of such areas are computer-aided design (CAD), office information systems (OIS), software engineering support (CASE) and GIS. Each area is characterized by one or more non-classical features, such as structurally complex information, specialized graphical requirements, non-standard transactions or version control requirements.

Regarding the specific question of applications of object-oriented ideas to GIS, there has been a progression from the publication of work that introduced the concepts in the context of GIS (Egenhofer and Frank 1987, 1992, Worboys *et al.* 1990, to the use of such an approach in implemented systems. Noteworthy systems that incorporate object technology include Intergraph's TIGRIS (Herring 1991), Smallworld GIS (Chance, *et al.* 1990, Newell 1992) and Geo++ (van Oosterom and van den Bos 1989). There has also been a series of interesting experiments with the object-oriented database system O_2 (David *et al.* 1993, Scholl and Voisard 1992a, b). As yet there are few proprietary GIS that have object-oriented features, but those that do exist have been highly successful for specialized markets and it is likely that more proprietary object-oriented GIS will follow.

2. Object-oriented concepts

There are two broad and opposing classes of models of geographical information. The class of *field-based* models treats such information as collections of spatial distributions, where each distribution may be formalized as a mathematical function from a spatial framework (for example, a regular grid placed upon an idealized model of the Earth's surface) to an attribute domain. Patterns of topographic altitudes, rainfall and temperature fit into this view. The class of *entity-based* models treats the information space as populated by discrete, identifiable entities (or objects), each with a geo-reference. As observed in Couclelis (1992), 'the object-versus-field debate in GIS closely parallels a much more fundamental controversy in the philosophy of science, that between the atomic and the plenum ontologies'. These two model types result in two opposed GIS implementation approaches, namely vector and raster.

Object-oriented models decompose an information space into objects. As stated in (Mattos *et al.* 1993), an object must be

— identifiable
— relevant (be of interest)
— describable (have characteristics)

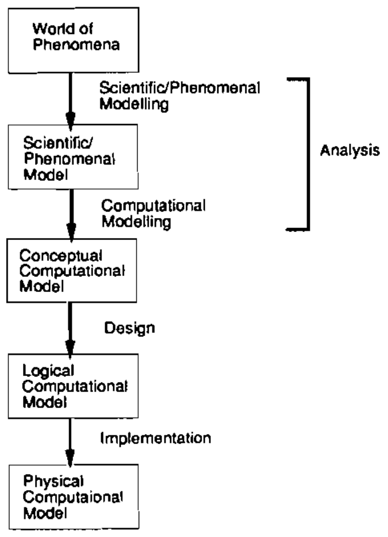
World of Phenomena
Scientific/Phenomenal Modelling
Scientific/ Phenomenal Model
Computational Modelling
Analysis
Conceptual Computational Model
Design
Logical Computational Model
Implementation
Physical Computaional Model

Figure 1 Scientific/phenomenal computational modelling.

With respect to the last item, the description of an object is provided by static properties (such as the name of a city), behavioural characteristics (such as a method of plotting the city at a particular scale) and structural characteristics (placing the object in the overall structure of the information space).

It is clear that entity-based geographical information models fit naturally with object-oriented concepts. Indeed, much of the discussion in this paper will take examples from entity-based models. However the converse is not true, since it is quite possible to apply object-orientation in a field-based context. Tomlin (1990) takes such a view by defining geographical fields as abstract data types. Applications of this approach to environmental modelling are discussed in Kemp (1992).

Before we proceed further, we should ask ourselves a question, 'Object-oriented approach to what?' and it is here that an initial confusion occurs. A GIS is a multi-faceted entity, consisting of hardware, software and data, and going through several processes in its life-cycle, from conception through design and implementation to use. The object-oriented approach can be applied at all of these different levels and has a slightly different meaning at each. The life-cycle begins with *analysis*, when the issue of what the system is required to do is clarified. Following analysis comes *design*, when the problem of how the system will satisfy its requirements is tackled. The translation of the design into something that really works is the *implementation* stage. Following that comes continued *usage* and *maintenance*. This sequencing of activities in system development is not meant to suggest a strict temporal dependency between stages. In fact, in most cases the dependency is logical but not temporal: for example, in an evolutionary approach to system development, further analysis and design might follow a pilot implementation, and this process might continue to loop, iterating to a possibly acceptable solution. However we view the system development process, object-oriented approaches may be adopted at each stage.

We have so far rather neglected what for many is the most important phase of modelling, namely the construction of the scientific or phenomenal model. Figure 1 shows a refinement of the system life-cycle described above, in order to embrace this important activity. The single analysis phase above has been decomposed into two stages:

1. Phenomena in the real world are represented as processes in an appropriate abstract scientific or phenomenal model.
2. Structures and transformations in the scientific/phenomenal model are represented in forms that are computationally tractable but independent of any specific computational paradigm.

Thus, a particular instance of a conceptual computational model will prescribe in non-specific yet computationally meaningful terms the domains and transformations that must be applied in order to model the phenomenological processes as interpreted by the scientific/phenomenal model. The design phase moves from the specification provided within the conceptual computational model to the design of the logical computational model. The logical model is embedded in a particular computational paradigm, whether relational database, object-oriented, deductive, or some other. The *physical computational model* corresponds to implementation and is the eventual representation of the real world phenomena as computational processes inside a physical computer.

Such a staged representation leaves us with an important difficulty, and it is this difficulty that motivates the object-oriented approach at each of the stages. Computer science has developed the concept of *impedance mismatch* to explain certain problems with system development. The idea is that the gap between the constructs that are available at different stages of the development process make the transition from one stage to the next inefficient: information may be lost or a simple concept may get hidden in a complex modelling paradigm. An important instance of this problem occurs if a scientific/phenomenal model, expressed using high-level scientific constructs, must be translated into a low-level conceptual computational model in order that it can be implemented. Thus, the heartfelt cry of many scientists that computers force them to make square holes for their round and fuzzy edged pegs. However, it is not the computers that force scientists into unnatural modes of expression, but the primitive computational models in current use. In a nutshell, object-orientation is one of several attempts to raise the level of the computational modelling environment so that the impedance mismatch problem is lessened. We now discuss some of the concepts upon which this approach is based.

2.1. *Objects, types and classes*

As might be expected, the concept of *object* is central to the object-oriented approach. It arose out of a desire to treat as primitive not just the static data-oriented aspect of information but also its dynamic behaviour. Dealing first with the static aspect of an object, this is expressed by a collection of named *attributes*, each of which may take a value from a pre-specified domain. A city might have **name, centre, population** among its attributes. A particular city might take the value 'London' for the **name** attribute. The totality of attribute values for a given object constitute its *state*.

The dynamic, behavioural side of an object is expressed as a set of operations that the object will perform under appropriate conditions. For example, the idea of a region is captured not just by specifying a set of points or curves giving the boundary of its spatial extent (the data), but also the operations that we expect a region to support, such as operations to return real numbers giving the region's area and perimeter, an operation to plot the region at different scales and levels of generalization, operations that create and delete regions from the system, operations that return the lineage of the region (when it was created, by whom, to what accuracy, etc.). The key notion is that

object = state + functionality

This is the minimum, without which we do not have the object-oriented approach. In fact, many researchers would say that state plus functionality is below the minimum for the definition of an object (merely defining an abstract data type). Essential properties would include object identity and inheritance. As is stated by Fong *et al.* (1991), an object is something 'which plays a role with respect to a *request* for an *operation*. The request invokes the operation that defines some service to be performed'. Exactly how the object will respond in any particular case is dependent on its state. A request to print a region may result in a picture that depends on the scale and the boundary data values.

An object is thus characterized by its behaviour, which is the totality of its responses to requests. Objects with similar behaviours are organized into *types*. Thus, we might have the type **region**, as above. At the design and implementation stages of the system life-cycle, such semantic notions must be represented by specific computational objects. Each such object will have data structures holding its attribute values and methods that implement specific operations. At this level, we talk about object classes, which are groupings of objects with corresponding data structures and methods. Thus, an object class is an implementation construct while an object type is a semantic notion. The parallel between type and class with intension and extension is useful here.

2.2. *Further properties of objects*

As has already been noted, different object-oriented models emphasize different constructs. The list below itemizes some of the more common ones. We have tried to distinguish between the roles of the construct at different stages of the modelling process, especially between the scientific/conceptual levels (sometimes called the real world) and the logical and physical levels (system levels).

Object identity: primarily a system construct, although it does have resonances at all stages in the modelling process. At the scientific/conceptual level, the issue is whether objects have unique identities, independent of their attribute values. This corresponds to a natural idea, which can be simply be expressed by means of an example.

> My name is Mike, but I am not my name. My name may change, indeed every cell in my body may change, but my identity remains the same.

Object identity may be implemented at the system levels by insisting that each system object has a unique identifier, created when the object is created, never altered and only dropped when the object is destroyed. This identifier is additional to other attributes of the object. Almost all self-respecting object-oriented systems support object identity.

Encapsulation: very much a system construct (but also a programming philosophy). One of the primary original motivations behind object-oriented system design is 'reuse'. The idea is that systems are more quickly developed if they can be constructed by bolting together a collection of well-understood and specified sub-components. The key to successful reuse is that sub-components should behave in a predictable way in any setting, and this can only be achieved if their internals are insulated from the external environment. This is the principle of encapsulation, where an object's state and the methods that it uses to implement its operations are not directly visible externally, but only by means of a clearly defined protocol prescribed in the definition of the object.

Inheritance: an important system and real-world construct. Reuse is also a motivating factor here. In the systems context, inheritance involves the creation of a new object (or object type) by modifying an existing object (or type). In real-world modelling, inheritance acts in two ways, one the inverse of the other:

Generalization: abstracting the properties of several types. For example, **village, town** and **city** generalize to **settlement**.

Specialization: differentiating types according to specific roles. For example, type **traveller** specializes to **road_user** and **rail_user**.

Object-oriented models will differ in the support they provide for these constructs. The end-result of either will be an *inheritance hierarchy* with subtypes further down the hierarchy than their corresponding supertypes.

Composition: an important system and real-world construct allowing the modelling of composite objects with complex internal structure. There are several ways in which a collection of objects might be composed into a new object. Some of the possibilities are briefly mentioned below.

Aggregation: An aggregate object is one that is made up of component objects. For example, an object of type **property** might be composed of objects of type **land parcer, dwelling** and **boundary**. To quote Rumbaugh *et al.* (1991), 'an aggregate object is semantically an extended object that is treated as a unit in many operations, although physically is made up of several lesser objects'. Sometimes, an extra constraint is imposed that the existence of the aggregate object is dependent upon the existence of its sub-objects, in that it cannot exist without them (and also sometimes *vice versa*).

Association: An associated or grouped object is one that is formed from a homogeneous set of other objects. For example, an object of type **districts** might be an association of individual **district** objects.

Ordered association: When the ordering within the object set is important, we have an ordered associate object. For example **point_sequence** might be an object structured as a linear ordering of **point** objects.

Polymorphism: primarily a system construct, allowing the same operation to be implemented in different ways (by different methods) in different classes. For example, the operation **perimeter** may have different implementations for classes **square, triangle** and **circle**. Polymorphism becomes powerful in combination with inheritance. It provides flexibility for execution of processes in information systems, since operations need only be bound to implementations at run-time.

3. Object-oriented databases

For GIS, it is especially pertinent to examine the role of object-orientation in information systems. Products labelled object-oriented databases have been available for purchase for some years. An object-oriented database management system (OODBMS), in addition to supporting the object constructions described above, must also provide a safe environment for the management of persistence in the object class hierarchies inhabiting it. Ideally, all the features available in a modern database should be provided by an OODBMS. These include

— Schema management, including the ability to create and change class schemata.
— Providing a usable query environment, including automatic query optimization and a declarative query language.
— Storage and access management
— Transactions management, including control of concurrent access, data integrity and system security.

Unfortunately there are technical problems in matching some of these facilities with the system object model. For example, the provision of query optimization (and therefore the possibility of a non-procedural query language) is made difficult by the complexity of the object types in the system. No longer do we have a few global and well-defined operations, such as the relational operations of project, select and join, but a multitude of methods implementing operations, for each of which there might be no measure of implementation cost.

Another problem is the seeming incompatibility of indexing with the notion of encapsulation and object-identity. Here, the difficulty is that in most relational systems, indexes rely on direct access to attribute values; but an object is only accessible via messages through its protocol and identified by its object-id. This obstacle is more practical than theoretical and may be overcome by the use of active databases, associative languages, or the embedding of index updates in method code, to trigger updates without direct access to attribute values.

On the other hand, an OODBMS has advantages. Some of these are technical and involve reductions in joins and possible speed-up using object-identifiers. However, the main advantage for the purpose of this account is the use of a system that is at the same level as the conceptual object model. To implement a rich semantic data model using a limited database model is like playing a Beethoven piano sonata on the trumpet, the instrument does not match the conception and much of the intended meaning will be lost in the execution. However, it must be said that even today, object-oriented databases are still rarely used for realistically large-scale applications. Many of the solutions found by research and development have yet to find their way to the market-place.

4. Applications to geo-referenced information

This section examines the applications of object-oriented approaches to geo-referenced information. Entities in such a system will have several categories of dimension along which attributes may be measured. These include spatial, graphical, temporal and textual/numeric (see figure 2). To give a simple example a land parcel may have a polygon representing its boundary in the real world (spatial), a polygon and point representing its cartographic form at differing levels of generalization (graphical), times when it was created in the real world and in the system (temporal), and attributes describing its area, owner and name (textual/numeric). In practice, the dimensional categories do not arise in such an unmixed form. For example, a land parcel may change shape and have different owners at different times. Modelling approaches must allow these categories to be unified. We now expand on the spatial, graphical and temporal dimensions.

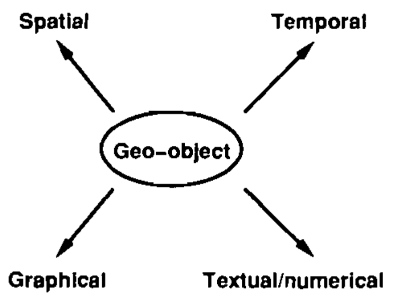

Figure 2 Dimensions of geographical information.

4.1. *Spatial objects*

The specification of a spatial object depends upon the structure of the real-world space in which the object is embedded. The spatial dimensions have always been the major focus of activity for GIS research. Earlier work (Egenhofer *et al.* 1989), (Güting and Schneider 1993) and (Worboys 1992), has discussed the nature of objects in the underlying spatial structures. The following subdivision is often useful.

Euclidean: admitting measurements of distances and bearings between objects. Given a suitable frame, objects in Euclidean space may be represented using set of coordinate tuples.

Metric: admitting measurements of distances (but not necessarily bearings) between objects. An example of a metric space is a travel-time surface (assuming that travel-time is symmetric, in the sense that the travel-time between x and y is the same as the travel-time between y and x).

Topological: admitting topological (but not necessarily metric or Euclidean) relationships between objects. Topological relationships are those that are preserved under 'rubber-sheet' transformations, which continuously deform the underlying space. Examples of topological relationships are connectivity, adjacency and incidence.

Set-theoretic: admitting only general set-theoretic relationships, such as membership, containment, union and intersection. The containment structure of a hierarchy of administrative regions is an example.

In order to define an object, there is a requirement to specify state and behaviour. The usual situation is that the underlying space is Euclidean and each spatial object is specified by a set of coordinate tuples or computable equations. Another approach is to specify a set of primitive objects, out of which all others in the application domain can be constructed, using an agreed set of operations. Primitive spatial object classes that have been suggested include closed half-planes (although these suffer from the un-intuitive property of being infinite in extent), simplical complexes (the disadvantage here being computational expense in building with such primitive objects) and point-line-polygon primitives (common in existing systems).

Having defined an object's states, object behaviour is defined by operations, which may be broadly classified as above into Euclidean, metric, topological and set-theoretic. The provision of methods to implement operations will depend upon the operations and upon the representation of the objects. Egenhofer and co-workers (see, for example, (Egenhofer 1991)) have written several papers exploring some of the background and connections between these operations, which they call spatial relationships. The

Table 1. Temporal dimensions supported by system types.

	No support for transaction time	Support for transaction time
No support for database time	Static	Rollback
Support for event time	Historical	Bi-temporal

author (Worboys 1992), assuming a Euclidean embedding in 2-space, has provided a formal description of common spatial object classes and operations, using simplicial complexes.

4.2. *Graphical objects*

The distinction is not always made between the spatial and graphical dimensions. The idea is that spatial objects exist in the application domain, while graphical objects are their presentational form. This echoes back to our earlier distinction between modelling at the phenomenological level and the systems level. Graphical objects are required for systems modelling. The relationship between geo-spatial and graphical objects is the subject of cartography. The transformation from spatial to graphical objects introduces abstraction and generalization processes. Graphical objects themselves are required to hold methods for rendering on appropriate presentation media.

4.3. *Temporal objects*

Temporality is an inherent aspect of geo-information, but until recently has been relatively neglected by GIS research. This history of neglect is probably due to the inadequacy of the technological support, both in terms of hardware and software. This situation is now changing. The speed and capacity of hardware, along with the software that is now becoming available, make temporal information systems possible. There has been a history of research amongst the database community into support for the temporal dimensions (see (Tansel *et al.* 1993) for a recent collection of papers and (Soo 1991) for a briefer survey of the field). Research into the management of spatially and temporarily referenced information in a unified system is relatively novel (see (Al-Taha *et al.* 1993) for recent list of papers). Future GIS will support temporal as well as spatial aspects of geographical information. In this section we discuss temporality from an object perspective, after providing some general background to temporal systems.

Time in information systems is measured along at least two separate dimensions.

Database time: also called transaction or system time. This is the time when transactions take place with an information system.

Event time: also called real-world or valid time, when the events actually occur in the application domain.

Temporal information systems can be subdivided into four types: *static*, *historical*, *rollback* and *bi-temporal* (Snodgrass 1992). Each type supports zero to two temporal dimensions. *Static* systems support neither database nor event time; *historical* systems support only event time; *rollback* systems support only database time and *bi-temporal* systems (sometimes termed just *temporal*) support both database and event times. Table 1 shows the temporal dimension(s) that are supported in each case. For work in the area of bi-temporal systems see (Ariav 1986) and (Snodgrass 1987).

It is possible to distinguish different structural types for each of the temporal dimensions.

Discrete or continuous: Time may be measured as a discrete or continuous variable. A continuous temporal variable is used where processes require the measurement of time to be possible at arbitrary levels of precision. For example, the position of a marker on a glacier may be measured at various time points, and the underlying model might assume a theory of interpolation for other time points. A discrete temporal variable is used where time is measured at certain points or intervals and the variation is discontinuous between these points. For example, an administrative boundary may occupy one position at time *x* and another position at time *y*, but it probably does not make sense to say that the boundary occupies some intermediate position between times *x* and *y*. It does not creep continuously but jumps discretely.

Topology: Differences in structure arise from differing temporal topologies. Time may be linearly ordered, such that for any two different temporal events, one will be simultaneous with or earlier than the other. Time may be partially ordered, allowing the possibility of alternative futures, pasts, or both. Events may form a temporal cycle, exhibiting periodicities.

Dimensionality: Temporal events are usually associated with time points or intervals (durations).

The object class hierarchy for purely temporal objects is in general simpler than the spatial case, at least for the special case where the temporal dimensions are assumed linear and discrete. An analysis of uni-dimensional temporal relationships was given in (Allen 1984). In order to model the case of two orthogonal dimensions, such as database time and event time, we may think of a two-dimensional array, where the elements in the array indicate times in the two dimensions. The array is composed of rectangles whose length and breadth are associated with durations in the two orthogonal dimensions.

Just as we have made the distinction between spatial and graphical object classes, so should we do the same for time. There is a distinction between time in the real world (whether of events or interactions with the information system) and time as presented to the user of the system, for example in animations.

4.4. *A unified spatio-temporal object model*

There are sound reasons to construct a conceptual model that brings together the purely spatial with the purely temporal by forming an aggregated object consisting of a member of the spatial hierarchy and a bi-temporal reference. The class of such aggregates we term the class of spatio-temporal objects. Such a class is sufficient to model many of the situations that are required in a temporal GIS. An approach taken by the author is to form a class of primitive spatio-temporal objects by associating a two-dimensional temporal element with a primitive spatial object. An algebra for manipulating such unified objects is the subject of current research. Early results are reported in (Worboys 1994).

5. Object-oriented GIS

There is now a moderately sized literature on the applications of the object-oriented approach to GIS, some of which has been discussed earlier. In this section, we discuss some of the published work on implemented object-oriented GIS. We have made a deliberate choice not to consider here proprietary systems, but to focus on systems reported in the research literature.

Choi and Luk (1992) extend a generic object-oriented database system by providing two additional layers above it, namely:

> **G-Kernel,** consisting of a geographic object model with high-level geographic object classes and operations, and a geometric object model holding implementation level geometric object classes and operations.
>
> **Query language processor,** which manages a functional query language invoked directly through a graphical interface by using a mouse or other pointer.

The authors give most of their attention to the query language processor, in particular the development of strategies for query optimization in this environment.

Milne *et al.* (1993) discuss the construction of an object-oriented GIS based on the general-purpose object-oriented DBMS, ONTOS. They use the US Spatial Data Transfer Standard (SDTS) to provide a model for the basic spatial object classes. ONTOS (1991) provides a persistent class library, written in C++, and allows new persistent classes to be constructed using C++ code. The authors extend ONTOS with a graphical user interface, map module, graphic display module and geographical database module. Database design is effected using the IFO notation described in (Hull and King 1987, Abiteboul and Hull 1987). Milne *et al.* compare the performance of ORACLE (a proprietary relational database) and SIRO-DBMS, (an extended relational DBMS with special GIS capability (Abel 1989) and implemented on ORACLE), with their extended ONTOS. Their paper reports an impressive performance gain for their the test data.

David *et al.* (1993) describe the implementation of an object-oriented GIS using the OODBMS O_2 from O_2 Technology (Bancilhon *et al.* 1992). Their paper describes one of several efforts to build an OOGIS using O_2. The authors' system, called GeO_2, distinguishes between model (semantic geographical model and localization model) and structure (spaghetti, network and map structures). The authors note inadequacies in O_2 in this context, such as lack of an array constructor.

6. Extended relational systems

The current paradigm for general-purpose information management is the relational model. It is now clear that current standard corporate relational databases are unsuitable by themselves for the management of geo-information. The majority of GIS implementing a relational model typically adopt a hybrid architecture where attribute data and their spatial references are stored and managed in independent structures. Such systems maintain the attribute data in a conventional DBMS but organize and manipulate the spatial data using conventional file-handling techniques and special-purpose software. The hybrid approach has shortcomings. In failing to maintain an entire map base within a single DBMS, geometric data are not subject to the same rigorous management as might be applied to the non-spatial data. Because of this, the handling of spatial data lacks such fundamental DBMS facilities as data security, integrity control, multiple user access and concurrency management. Also, and perhaps most importantly, if benefits are to be derived from distributed database technology, then GIS must be developed that deal with the particular complexities of distributed database management without being hindered by an approach that stores some data outside the DBMS, inaccessible to any distributed functions provided therein.

There is therefore a good case for developing unified extended-relational systems with spatial and non-spatial data under the same architecture. Unfortunately, a number of problems are known to inhibit the development of such systems, largely as a result of the special characteristics of spatial data (Healey 1991). There are at least two major projects, both in the U.S.A., that are exploring the suitability of extensible relational databases as unified management systems for geographic information. The Starburst project, from the IBM Research Laboratory at Almaden, California, has been examining extensions to standard relational database systems (see, for example, (Lohman *et al.* 1991)). The Postgres system (Rowe and Stonebraker 1987) is a generic extensible relational database, formed by expanding the traditional relational system Ingres. (It has many of the characteristics of an OOBDMS and is sometimes classified as such). Postgres is being used in Sequioa 2000 (Stonebraker and Dozier 1991, Guptill and Stonebraker 1992) to handle large images and spatial data associated with global change research.

We should not leave this section without mentioning the proposed SQL3 specification. SQL is the principle standard database query language for interaction with relational systems. The language is in a state of evolution, as evinced by the acceptance of the SQL-92 standard. The SQL3 specification moves to unify object and relational data. SQL3-style databases are becoming available in the market-place. It is clear that, as relational technology advances, it is taking on many of the constructs associated with the object-oriented approach and in turn is having an impact upon object-oriented technology.

7. Conclusions

We may summarize the argument developed in this paper as follows. In order to construct and effectively use information systems, several models of the application area are required. We may make a broad dichotomy:

— Models of the application area, taking into account the nature of the computational system in which the model will be implemented.
— Computational system models, taking into account the nature of the application area that they are serving.

The art of system modelling is getting these two extremes to merge and so produce a system that runs well and is appropriate for its use.

Research into the application of object-oriented ideas to GIS has come a long way in the past few years, from pioneering work by Egenhofer and Frank (1987), through the explorations of this paper's precursor (Worboys *et al.* 1990), to current work on conceptual modelling, system modelling and implementation. We have seen that object-oriented approaches can be taken at both ends of the modelling spectrum. The high-level constructs in these approaches gives both application experts and system modellers more scope to handle and represent their ideas about the real world and the application requirements. We have noted that although the conceptual modelling constructs exist, they are not yet fully supported by generic proprietary products. Even where such systems exist, they are still used only for a minority of applications involving geo-information. However, it is generally agreed that a standard corporate relational database is unsuitable for the management of geo-information. Looking into the future, there are two major options: to extend current relational systems with specialized functionality for geo-information; or to move to radically new approaches, amongst which are object-oriented systems.

This article has surveyed the range of concepts associated with the object-oriented approach. Naturally, such a review will reflect the author's tastes and biases. Space restrictions (not only restricted to computer storage) have led to some difficult choices over material. In particular, we have given little attention to specific examples of the application of object-oriented ideas but concentrated on the general principles. Other areas that would have been useful to discuss if space permitted are the possibilities offered by open environments.

The general literature describing the application of object-oriented models in the field of geographic information is fairly sparse. A notable exception is (Egenhofer and Frank 1992) which is complementary in nature and detail to the present work. The reader may find it useful to have some further references, which provide a coverage of specific areas within the field. One of the best recent books on object-oriented databases is (Khoshafian 1993). An earlier work by Khoshafian and Abnous (1990) deals in detail with the different semantics of the inheritance construct. For detailed discussions of database questions, see the object-oriented manifesto (Atkinson *et al.* 1990). A general and comprehensive treatment of object-oriented data modelling is given in (Booch 1994) with a complementary approach (Rumbaugh *et al.* 1991).

Acknowledgments

The author thanks the three anonymous referees, whose constructive comments enabled significant improvements to be made to the paper before final submission.

References

Abel, D. J., 1989, SIRO-DBMS: A database toolkit for geographical information systems. *International Journal of Geographical Information Systems*, **3**, 103–115.

Abiteboul, S., and Hull, R., 1987, IFO: A formal semantic database model. *Association for Computing Machinery Transactions on Database Systems*, **12**, 525–565.

Al-Taha, K. K., Snodgrass, R. T., and Soo, M. D., 1993, Bibliography on spatiotemporal databases. *Association for Computing Machinery, SIGMOD RECORD*, **22**, 59–67.

Allen, J. F., 1984, Towards a general theory of action and time. *Artificial Intelligence*, **23**, 123–154.

Ariav, G., 1986, A temporally oriented data model. *Association for Computing Machinery, Transactions on Database Systems*, **11**, 499–527.

Atkinson, M., Bancilhon, F., Dewitt, D., Dittrich, K., Maier, D., and Zdonik, S., 1989, The object-oriented database system manifesto. In *Proceedings of the First International Conferences on Deductive and Object-Oriented Databases*, edited by W. Kim, J. Nicolas and S. Nishio (New York: Elsevier), pp. 40–47.

Bancilhon, F., Delobel, C., and Kanellakis, P. (editors), 1992 *Building an Object-Oriented Database System: The Story of O_2* (San Mateo, California: Morgan-Kaufmann).

Booch, G., 1994, *Object-Oriented Analysis and Design with Applications* (Redwood City, California: Benjamin/Cummins).

Chance, A., Newell, R., and Theriault, D., 1990, An object-oriented GIS: Issues and solutions. In *Proceedings European Geographical Information Systems (EGIS) Annual Conference* (Utrecht: EGIS Foundation) pp. 179–188.

Choi, A., and Luk, W. S., 1992, Using an object-oriented database system to construct a spatial database kernel for GIS applications. *Computer System Science and Engineering*, **7**, 100–121.

Couclelis, H., 1992, People manipulate objects (but cultivate fields): Beyond the raster-vector in GIS. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, volume 639 of *Lecture Notes in Computer Science*, edited by A. U. Frank, I. Campari and U. Formentini, (Heidelberg: Springer-Verlag). pp. 65–77.

David, R., Raynal, I., Schorter, G., and Mansart, V., 1993, GeO_2: Why objects in a geographical DBMS. In *Advances in Spatial Databases (Proceedings of the Third*

Symposium Databases), volume 692 of *Lectures Notes in Computer Science*, edited by D. Abel and B. C. Ooi (Heidelberg: Springer-Verlag), pp. 264–276.

EGENHOFER, M. J., 1991, Reasoning about binary topological relations. In *Advances in Spatial Databases (Proceedings of the Second Symposium on Spatial Databases)*, volume 525 of Lecture Notes in Computer Science, edited by O. Gunther and H.-J. Schek (Heidelberg: Springer-Verlag), pp. 143–160.

EGENHOFER, M. J., and FRANK, A., 1987, Object-oriented databases: Database requirements for GIS. In *Proceedings of the International GIS Symposium: The Research Agenda*, Vol. 2 (Washington DC: U. S. Government Printing Office), pp. 189–211

EGENHOFER, M. J., and FRANK, A., 1992, Object-oriented modeling for GIS. *Journal of the Urban and Regional Information Systems Association*, **4**, 3–19.

EGENHOFER, M. J., FRANK, A., and JACKSON, J., 1989, A topological data model for spatial databases. In *Advances in Spatial Databases (Proceedings of the First Symposium on Spatial Databases)*, volume 409 of *Lectures Notes in Computer Science* edited by O. Gunther (Heidelberg: Springer-Verlag), pp. 271–286.

FONG, E., KENT, W., MOORE, K., and THOMPSON, C., (editors), 1991, X3/SPARC/DBSSG/OODBTG, Final technical report, National Institute of Science and Technology, Gaithersburg, MD.

GUPTILL, S., and STONEBRAKER, M., 1992, The Sequoia 2000 approach to managing large spatial object databases. In *Fifth International Symposium on Spatial Data Handling*, edited by D. Cowen (Columbia: International Geographical Union), pp. 642–651.

GÜTING, R. H., and SCHNEIDER, M., 1993, Realms: A foundation for spatial data types in database systems. In *Advances in Spatial Data (Proceedings of the Third Symposium on Spatial Databases), volume 692 of Lecture Notes in Computer Science* edited by D. Abel and B. C. Ooi (Heidelberg: Springer-Verlag), pp. 14–35.

HEALEY, R. G., 1991, Database management systems. In *Geographical Information Systems: Principles and Applications*, edited by D. J. Maguire, M. F. Goodchild and D. W. Rhind (Harlow, England: Longman), pp. 251–267.

HERRING, J., 1991, TIGRIS: A data model for an object-oriented geographic information system. *Computers and Geosciences*, **18**, 443–452.

HULL, R., and KING, R., 1987, Semantic database modelling: Survey, applications and research issues. *Association for Computing Machinery, Computer Surveys*, **19**, 201–260.

KEMP, K. K., 1992, Environmental modelling with GIS: A strategy for dealing with spatial continuity. In *Proceedings, GIS/LIS Annual Conference* (Bethesda: ASPRS and ACSM), pp. 397–406.

KIM, W., 1993, Object-oriented databases: Promises, reality and future. In *Proceedings of the 19th VLDB Conference*, (San Mateo, California: Morgan Kaufmann Publishers), pp. 676–687.

KHOSHAFIAN, S., 1993, *Object-Oriented Databases* (New York: Wiley).

KOSHAFIAN, S., and ABNOUS, R., 1990, *Object-Orientation: Concepts, Languages, Databases, User Interfaces* (New York: Wiley).

LING, T. W., and TEO, P. K., 1993, Toward resolving inadequacies in object-oriented data models. *Information and Software Technology*, **35**, 267–276.

LOHMAN, G., LINDSAY, B., PIRAHESH, H., AND SCHIEFER, K. B., 1991, Extensions to Starburst: Objects, types, functions and rules. *Communications of the Association for Computing Machinery*, **34**, 94–109.

MATTOS, N. M., MEYER-WEGENER, K., and MITSCHANG, B., 1993, Grand tour of concepts for object-orientation from a database point of view. *Data and Knowledge Engineering*, **9**, 321–352.

MILNE, P., MILTON, S., and SMITH, J. L., 1993, Geographical object-oriented databases: A case study. *International Journal of Geographical Information Systems*, **7**, 39–56.

NEWELL, R., 1992, Practical experience of using object-orientation to implement a GIS. In *Proceedings, GIS/LIS Annual Conference* (Bethesda: ASPRS and ACSM), pp. 624–629.

ONTOS INC., 1991, *ONTOS Developers Guide* (Burlington: Three Burlington Woods).

ROWE, L. A., and STONEBRAKER, M. R., 1987, The Postgres data model. In *Proceedings of the 13th VLDB Conference*, edited by P. Stocker and W Kent (San Mateo, California: Morgan Kaufmann Publishers), pp. 83–96.

RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., and LORENSEN, W., 1991, *Object-Oriented Modeling and Design* (Englewood Cliffs, New Jersey: Prentice-Hall).

SCHOLL, M., and VOISARD, A., 1992a, Geographic applications: An experience with O_2. In *Building and Object-Oriented Database System—The Story of O_2*, edited by F. Bancilhon, C. Delobel and P. Kanellakis (San Mateo, California: Morgan Kaufmann), pp. 585–618.

SCHOLL, M., and VOISARD, A., 1992b, Object-oriented database systems for geographic applications: An experiment with O_2. In *Geographic Database Management Systems*, edited by G. Gambosi, M. Scholl and H.-W. Six (Heidelberg: Springer-Verlag), pp. 103–137.

SNODGRASS, R. T., 1987, The temporal language TQuel. *Association for Computing Machinery, Transactions on Database Systems*, **12**, 247–298.

SNODGRASS, R. T., 1992, Temporal databases. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, volume 639 of *Lecture Notes in Computer Science*, edited by A. U. Frank, I. Campari and U. Formentini (Heidelberg: Springer-Verlag), pp. 22–64.

SOO, M. D., 1991, Bibliography on temporal databases. *Association for Computing Machinery, SIGMOD Record*, **20**, 14–23.

STONEBRAKER, M., AGRAWAL, R., DAYAL, U., NEUHOLD, E. J., and REUTER, A., DBMS research at the crossroads: The Vienna update. In *Proceedings of the 19th VLDB Conference*, edited by R. Agrawal, S. Baker and D. Bell (San Mateo, California: Morgan Kaufmann Publishers), pp. 688–692.

STONEBRAKER, M., and DOZIER, J., 1991, Large capacity object servers to support global change research. Sequoia 2000 1, Technical Report, Computer Science, University of California at Berkeley, USA.

STONEBRAKER, M., ROWE, L., LINDSAY, B., GRAY, J., CAREY, M., and BEECH, D., 1990, Third-generation database system manifesto. In *Proceedings of IFIP DS-4 Workshop on Object-Oriented Databases*, pp. 571–584.

TANSEL, A. U., CLIFFORD, J., GADIA, S., SUSHIL, J., SEGEV, A., and SNODGRASS, R. T., (editors), 1993, *Temporal Databases: Theory, Design and Implementation* (California: Benjamin/Cummings).

TOMLIN, C. D., 1990, *Geographic Information Systems and Cartographic Modelling* (Englewood Cliffs, New Jersey: Prentice-Hall).

VAN OOSTEROM, P., and VAN DEN BOS, J., 1989, An object-oriented approach to the design of geographic information systems. *Computer and Graphics*, **13**, 409–418.

WORBOYS, M. F., 1992, A generic model for planar geographical objects. *International Journal of Geographical Information Systems*, **6**, 353–372.

WORBOYS, M. F., 1994, A unified model for spatial and temporal information. *The Computer Journal*, **37**, 26–34.

WORBOYS, M. F., HEARNSHAW, H. M., and MAGUIRE, D. J., 1990, Object-oriented data modelling for spatial databases. *International Journal of Geographical Information Systems*, **4**, 369–383.