NATIONAL CENTER FOR GEOGRAPHIC INFORMATION AND ANALYSIS

University of California, Santa Barbara

State University of New York at Buffalo

University of Maine

and

CALIFORNIA DEPARTMENT OF TRANSPORTATION (Caltrans)

Conference on Object Orientation and Navigable Databases

Report of the Meeting

March 15-16, 1996

Richard Church, UC Santa Barbara

Thomas Cova, UC Santa Barbara

Ramez Gerges, Caltrans

Michael Goodchild, UC Santa Barbara

Technical Report 96-9

INTRODUCTION

Background

In March, 1996 the National Center for Geographic Information and Analysis (NCGIA) and the California Department of Transportation (Caltrans) jointly sponsored a conference on the application of concepts of object orientation to navigable databases. This document is the summary report of that conference. It includes a report on the conference objectives, discussion, and conclusions; a list of the participants; and background papers prepared for presentation at the conference by several of the key participants.

For the purposes of the conference, navigable databases were defined as databases capable of supporting such ITS applications as in-vehicle map display, synthesis of navigation directions, and direction of vehicles to a given street address.

Object orientation (OO) is much more difficult to define, since it consists of a set of concepts that have come to dominate many aspects of computing over the past two decades, including software development, user interface design, and databases. Databases that implement the concepts of object orientation are generally regarded as an advance on earlier varieties, notably relational databases, but the field is in its infancy and while several object-oriented database products can be found on the market, there is relatively little uniformity between them.

Given this uncertainty, Caltrans and NCGIA jointly undertook to sponsor a conference with the following objectives:

- to evaluate the range of products currently available
- to assess the total experience to date with object-oriented databases (OODBMS)
- to compare OO and more conventional approaches with specific emphasis on distributed storage, processing, and maintenance

Invitations to the workshop were issued to ten experts in the fields of OO theory, navigable databases, and object-oriented geographic information systems GIS; Caltrans personnel; NCGIA researchers; and other invited participants with knowledge and interest in the topic of the conference. The experts were Fran□ois Bancilhon, O2 Technology; Kurt Buehler, Open GIS Consortium; Max Egenhofer, University of Maine, Orono and NCGIA; David Fletcher, University of New Mexico; Andrew Frank, Technical University of Vienna; Michael Franklin, University of Maryland, College Park; John Herring, Oracle Corporation; Shashi Shekhar, University of Minnesota; Alan Vonderohe, University of Wisconsin, Madison; and Michael Worboys, University of Keele. The full list of participants can be found in Appendix I.

Structure of the Report

The report follows the structure of the meeting’s agenda, in the form of summaries of presentations, major points made in discussion, and conclusions. Position papers prepared prior to the meeting are included in Appendix II. The meeting opened with two keynote presentations, from David Fletcher on “Navigating in a Digital Object Habitat” and from Andrew Frank on “Object Oriented Modeling for Geographic Information Systems in Transportation”. Ramez Gerges of Caltrans and Richard Church of NCGIA then gave their views of the meeting’s objectives. After lunch, the remaining eight experts were asked to give their views of the morning’s presentations, and to discuss their own position papers as appropriate. The final session of the day was directed at identifying appropriate topics for the following day’s breakout sessions, and was facilitated by Michael Goodchild of NCGIA.

The second day opened with separate discussions in three breakout groups. After lunch, each breakout group reported to the meeting as a whole. The meeting ended with closing remarks from each of the experts.

OPENING KEYNOTES

Navigating in a Digital Object Habitat (Fletcher)

David Fletcher began his presentation by posing the question: What is the current state of object based modeling? Everyone has something to say about objects, but there are two primary views:

- objects are very natural and everyone can comprehend them, and
- objects have to be one of the most bastardized and hackneyed and confusing concepts in the industry; everyone claims to have them.

In reality, object orientation has provided us with a new way of understanding and experiencing transportation and other large scale systems, and its concepts should be incorporated into the next generation of information systems. There are four aspects of objects that extend power:

- they are semantic models instead of syntactic models, and they deal with the meaning of things independent of implementation,
- the problem domain and the digital habitat converge and are congruent with the same model,
- digital objects can be liberated from the natural laws of their physical counterparts, and
- coupling of digital objects with their physical counterparts can lead to a form of symbiotic interaction.

Each of these concepts is explored in greater detail in the following sections.

Our First Experience: Linguistic

Object technology was designed for modeling, and every object is a model of something. Objects can express the essence of experience in human terms, as objects are semantic abstractions that reveal behavioral patterns. For semantic modeling, we need a common language. Also, we need to understand the problem domain and to extract the important issues. There are two primary domains:

- facilities domain (planning, engineering, construction, maintenance, and engineering transportation systems), and
- transportation systems domain (trip planning and traffic control for highways, railroads, public transport, and water transport).

In the facilities management domain there are objects like road segments, highways, connectivity, and routes. In general, it’s a very static world where changes are brought about by construction and administration. In the transportation systems domain there are objects like paths, trips, nodes, links. It’s a very dynamic domain. Transportation is a combination of the facilities management and transportation systems domains. A key issue is how to resolve the subtleties of the two domains. It's difficult to tell the two domains apart by their representations in GIS. A producer-consumer model is useful in understanding the relationship between the two domains, where facilities management is responsible for making a link available, and transportation operations is concerned with using the best link. The transportation system is the connection between the two domains. The ITS domain is a fusion of the two domains. How do we reconcile these two domains? There are many objects and many classes. Some are very persistent, and some are very transient. Here are some example ITS objects, both physical and virtual:

Physical	Virtual
transportation system	network
system components	links
intersections	nodes
signed routes	path
vehicle	network chain
origin/destination	
trip	

Each of these components lives in a physical, spatial, temporal, and functional context (a 5-D space). A major issue is the complex set of behaviors that arise from all the ITS objects. Objects are integrated and have physical, spatial, temporal, and functional characteristics. The state of a component is interdependent with proximate components. The number of interactions between components is enormous and system behavior is non-linear.

Our Second Experience: Convergence

One way to think about objects is in terms of their correspondence to physical objects, but they can also be considered 'real' entities. This correspondence leads to a convergence of the problem and object domains. The same model can reflect physical and digital objects, allowing software objects to represent real-world objects directly, and thus to mirror human thinking. But if this level of transparency can be achieved, it begs the question: What are the appropriate objects for ITS, and what is the appropriate ITS architecture?

Our Third Experience: Objects are weakly homomorphic

The liberation of objects from the laws that control the behavior of their physical counterparts is also a powerful benefit of the object approach. We should think of objects as real entities and use the same model for physical and digital objects. Good software architectures are those that reflect the real world problems. The domain is the problem, and the habitat is the structure in which the solution will be created. The complete system is the real thing, the model, and the software. Digital objects can be made smart, but what are the "natural laws" of the habitat? Through OO principles of inheritance, the system behavior will be distributed to the behaviors of individual objects and state changes in one object will be reflected in other objects. The physical and digital objects can be coupled and digital objects can be attached to each other. Motion is the act of changing state and navigation is the organization of motion from one state to another, where navigation is in space and time. But digital objects can be smarter than their physical counterparts, and this property can be exploited in OO systems.

Our Fourth Experience: Objects can be symbiotic

Objects can become things that think. Changes in the digital domain can be reflected in the physical domain and vice versa. How coupled is a physical object to its digital counterpart (loose vs. tight coupling)? Example natural habitats occupied by physical objects might be a stuffed animal museum, live caged animals, a natural zoo, or the Savanna. Example geographic habitats might be a static feature inventory, a dynamic feature inventory, interactive digital objects, or highly interactive symbiotic objects. Some example transportation habitats include a static link inventory, dynamic links, interactive links and vehicles, and symbiotic vehicles, infrastructure, and environment.

Navigating in a Digital Object Habitat

Navigating in a digital habitat is more than just moving in space. Motion is the act of changing state, when an object changes location in the physical, space, time, and functional dimensions. Any digital object can move, and navigating through this habitat means planning, recording, or controlling the movement between an origin state and a destination state. A trip is a series of choice points defining travel links, where each point represents a spatial (e.g., change direction), temporal (e.g., wait or proceed), and functional decision (e.g., change mode). An issue arises when one tries to grasp what proximity means in this habitat. Some example navigation tasks include point A to point B (navigate space only), walk to bus stop, wait for bus (navigate space and time), wait for conditions to change (navigate time only—time travel), best way to get package from Albuquerque to Santa Barbara by next Tuesday (navigate space, time, function).

Conclusions

Object-orientation is the best tool we have available to analyze and develop complex systems. Semantic analysis, convergence, liberation, and coupling extend the basic object-oriented toolset. Generalized navigation is a non-trivial problem, and there is need for fundamental research in both domain and habitat issues. To summarize the issues for this meeting:

- We need a common language.
- How are multiple views of the same object reconciled?
- There are many classes and many objects.
- Some objects are very persistent; some are very transient.

- The number of potential interactions is enormous.
- System level behavior is non-linear and will emerge from the total interaction state.
- What is the architecture?
- What are the natural laws of digital habitats?
- What are an object's non-physical responsibilities?

Object-Oriented Modeling for Geographic Information Systems in Transportation (Frank)

Object-oriented modeling

Object-orientation (OO) is a software engineering method to solve the software crisis. Software production is taxing the human ability to analyze, understand, and describe complex situations. OO is the trick to solve this problem, as it has something to offer. Computers proceed in a sequence of steps; traditional programming consisted of describing the steps necessary. It followed a procedural pattern. Complex procedures can be learned by humans, but not described by language (e.g. artists, shoe-tying). Languages serve well at describing complex static relations between things, often using spatial relations metaphorically. Procedural descriptions are difficult, but declarative descriptions are easier for humans (e.g., SQL, Prolog). Objects are a fundamental experience: grabbing, moving, observing objects are among the first experiences of babies. This experience is reflected in language: several nouns with qualifiers and relation expressions are combined with one verb to form a sentence. Objects are individual things, which we can manipulate with specific operations (i.e. seat, table, car, road). Objects which respond to the same operations are put in the same category. Objects can be abstract, either physical or non-physical, and philosophers have considered these questions since Aristotle (e.g., road accidents, county boundaries, maintenance level of a road). Nouns usually describe objects and verbs operations.

In regards to the history of OO modeling, simulation (SIMULA) was one of the first OO languages. Graphical user interface toolkits are now typically provided as a set of object classes. OO modeling is currently a completely general technique, widely applicable.

Formalization of object-oriented methods

It is important to clearly define one's language, and computer science has done this. Formalization is necessary to use object-orientation for computer programming, and it is generally useful to avoid misunderstandings and give a defined semantics for descriptions. What is the width of road? Simple question, right? But you get widely different answers depending on who you ask. One might think about boundaries of properties, runoff, or surface.

OO methods are based on algebra, where an algebra is a triple of a set of individual objects, operations applicable to these individual objects, and rules which describe the result of the operations. For example, one possible algebra consists of the integers, with the operations +, -, * and the rules a + 0 = a, a + b = b + a, etc. Also, we might have cars with the operations start, accelerate, brake, stop, get position, and rules like "the current position is the previous position plus the time interval times the speed."

OO methods are used in computer science for programming languages (Smalltalk, C++, graphical user interface design, object-oriented DBMS (O2, Gemstone), and specification languages (Larch). There exists substantial differences in terminology and concepts used, but there are three basic things: class, data, and instance. A class is a set objects with the operations applicable, including rules describing their effects (but without implementation). Data consists of representations of objects. And instances describe how the operations are carried out for a particular data type.

Example:

```
class Numbers n where
        add :: n -> n -> n
data Num = N Int
instance Numbers Num where
        add (N a) (N b) = N a + b
```

Most current systems do not differentiate between the three aspects: operations, data, implementation. Specification languages prefer a 'behavior' description (the class), programming languages an 'implementation' point of view (the instance). Smalltalk has introduced a particular terminology of 'message passing' and methods, which are not much different from procedure calls and operation. All objects in a class have the described behavior. An object can be in more than one class: a number is also in the class of objects of comparables.

```
class Eq e where
        eq :: e -> e -> Bool
instance Eq Num where
        eq (N a) (N b) = a ==b
```

This is often called multiple inheritance, as the data type Num inherits operations from Numbers and from Eq. Relations between objects follow patterns:

- **classification**: objects of the same behavior form classes,
- **generalization**: classes with similar operations are grouped to generalized classes,
- **aggregation**: groups objects to form new individuals (often separated into aggregation and association).

Objects can play roles: a person is owner of a car, driver of a car etc. All examples in this section are given in Gofer, a functional programming language in the Haskell tradition.

- It is **mathematically clean** and includes language constructs for object-oriented design.
- It is **compact**, which helps specification writing.
- It can be **executed** to observe the behavior, which helps enormously to detect errors (rapid prototyping).

Ontology for GIS-T

An ontology relates the essential elements of our understanding of a problem or phenomenon, and the creation of an ontology is an essential part of object-oriented modeling. In transportation, we might understand the relevant geographically dispersed phenomena to include such elements as:

- concrete objects (roads, bridges, vehicles)
- geographic features (mountains, valleys)
- administrative objects (counties, police beats)
- etc.

Ontologies differ by application. Only those differentiations necessary for a task are included in an ontology. Ontologies can be combined from small ontologies for single objects. Algebraic descriptions of ontologies allow one to describe combinations formally. The description of the ontologies of the objects involved helps to capture the meaning of the application. Ontologies of the base objects can be compared between applications and differences are detected early. Often applications seem to use the same objects but view them slightly differently.

Model of Road Network as Nodes and Edges

Consider the example of finding the shortest path through a network consisting of nodes and edges. The following steps define the base classes, plus the code to construct the network:

Dijkstra's Shortest Path Algorithm:

(1) Initialize: Set $W := \text{empty set},\ U := V,$
$l_U(x) := 0,\ l_U(y) := \infty$ for all $y \neq x, p(y) := *$ for all $y \in V$

(2) Determine the minimum of $l_U(y)$ over all $y \in U$. Choose a node $z \in U$ such that $l_U(z)$ equals the above minimum. Set $d(x,z) := l_U(z)$.

(3) Set $W_1 = W \cup \{z\},\ U_1 := U \setminus \{z\}$, as well as for all $y \in U_1$
$l_{U_1}(y) := \min(l_U(y), l_U(z) + w(z,y))$.
If in this last expression $l_U(y) > l_U(z) + w(z,y)$, then set
$p(y) := z$.

(4) If $W_1 = V$, the algorithm terminates. If $l_U(y) = \infty$ for all
$y \in U$, then the graph is not connected, and there exists no path
from *x* to the nodes in *U*. Otherwise set $U := U_1,\ W := W_1$ and
return to step (2).

Detailed models of road networks for driving must include turn restrictions and one way streets. Models for maintenance must include other objects, such as traffic lights, signs, and mile posts.

Object-oriented modeling methods help to integrate multiple levels of detail and different focuses of applications. But special problems occur due to different levels of detail in the areas of spatial resolution, thematic resolution, and temporal resolution.

Road Movement

The dynamic nature of the road environment poses particular problems. The example here demonstrates the applicability of the object-oriented method to vehicle movement.

```
type Pos = Int      -- meter
type Time = Int     -- seconds
type Speed = Int    -- meter per second

class Vehicles v where
        make :: Pos -> Time -> v
        at :: Time -> v -> Pos
        speedup :: Int -> Time -> v -> v
        slowdown :: Int -> Time -> v -> v
        slowdown a t v = speedup (-a) t v
```

```
data Vehicle = V Pos Time Speed

instance Vehicles Vehicle where
      make p t = V p t 0
      at tt (V p t v) = v * (tt - t)
      speedup a tt (V p t v) = V (at tt
             (V p t v)) tt (v+a)

v1 :: Vehicle
v1 = make 0 0
```

To calculate the position of a vehicle after speeding up and slowing down:

```
at 4 ( slowdown 2 3 (speedup 3 1 v1))
```

Tools

Object-oriented models can be constructed by starting from a prototypical case and then adding additional details.

Conclusions

Object-oriented modeling is a generic tool to describe complex situations. It allows **declarative** descriptions based on objects which are easier for the human cognitive system than complex **procedures**. It can be applied to

- geographic problems,
- graphical user interfaces,
- traffic simulation etc.

Geographic features as they appear in topographic maps are just one kind of object.

Object-oriented modeling is based on algebra, using classes of objects with the same operations and the same behavior.

An object can be in multiple classes and respond to operations from different classes.

Overview of the Meeting Objectives (Gerges)

We would like to move toward developing a background for supporting a digital transportation habitat The goal is to find a link between academic research and day-to-day operations in the field; in short, application of academic research in the real world.

The problem is to:

- Develop a knowledge base at the state DOT.

- Develop understanding for DOT operations problems for academia.
- Deploy and Implement Solutions.
- The missing link between DOT and academia—applied research, a two way transfer of problems and techniques.

The players are Caltrans and local government with many dimensions of interaction.

Many conferences have occurred, mainly centering on implementation issues. The need is to bring agencies, users, and vendors to address the issues of deploying a statewide navigable database. Some issues include:

- Data Collection
- Data Fusion
- Data Dissemination

The Caltrans Center for Interoperability was established as a testbed for applications and concepts to move ahead on these issues I'm discussing. I would also like to emphasize a few key issues:

- We are dealing with an intermodal transportation system.
- We want to decouple the source of the information from the users and agencies.
- Caltrans' goal is a navigable database to share between 12 semi-autonomous districts, and we want interoperability between districts with their own autonomous databases.

The questions to address are:

- Balance problems: what is there, not just GIS, but real-time data and feedback, enabling technologies.
- Is OO the way to go? It is a good tool, but will it help us attack and solve our problems? Do other options exist?
- Maybe a national standard for navigable DB's should exist.

Overview of the Meeting Objectives (Church)

Our objective for these two days is to identify how best to meet Caltrans' needs for a navigable data base system to support intelligent transportation systems and other transportation needs. The primary purpose of the meeting is to pursue the following questions:

- What is the experience to date in OO-DB (especially in terms of transportation or network applications)? For example, what installations, what kinds of success?

- What range of OO-DB products exist and what is best for the development of a navigable database? What will this allow us to do, that other technologies will not? Are there products that can fully support this need?
- How do the traditional relational DB products compare to the OO-DB in terms of the needs of a transportation agency? Why go OODB?
- How should an agency like Caltrans proceed in acquiring a data base system that will support growing ITS needs? Does this future revolve around OODB?

The issues of importance are:

- *Costs of OODB in terms of alternatives*: will it be more efficient?
- *Distributed system*: Caltrans is divided into regions, each maintaining a portion of the corporate database, no one location will likely have the entire database.
- *Interoperability*: Can we build an application that can simultaneously access data in different databases provided by different vendors (or different regions), to what extent are OODB systems interoperable today?
- *Transportability*: Will we have the ability to move an application from one network/hardware combination to another while using the same database?
- *Communication (e.g,. LRS)*: We must be able to easily handle linear referencing systems.
- *Expandability*: We cannot expect that functions and data types will be added over time.
- *Real-time*: Perhaps bank-based transaction systems, large amounts of data, can it handle more data and transactions that other approaches.
- *Scale*: e.g. Do objects represent individual units on the system like a bus or does an object represent a road segment which knows the presence of a bus?

Further details include:

- *Planar vs. Non-Planar Representations*: How to deal with intersections and grade separations?
- *Lane information*: How many systems maintain information on lanes?

Given that we have a need to store, retrieve, and analyze transportation data over time to support traffic operation centers, navigation, and routing, we need to find the most appropriate approach to store, retrieve, and analyze such data and to facilitate effective communication on a real-time basis. The requirements will be that the system be interoperable, distributed, transferable, operate in real-time, and be expandable. If OODB, then why? Are there problems with existing OODB systems? What are the objects?

Is the specific use of GIS technology to support transportation functions, e.g. personal navigation, incidence response, condition and need records, traffic management, traveler information, etc.?

PANEL MEMBER DISCUSSIONS ON POSITION PAPERS

In the afternoon of the first day, each panel member was allowed 10 minutes to briefly present their position paper. Following are the notes taken from this series of presentations.

François Bancilhon, O2 Technology

ODBMS History

1984 - Birth of the ODBMS concept (D. Maier, G. Copeland)

1989 - ODBMS definition: "The Manifesto" (M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S. Zdonik)

1993 - ODBMS standard: ODMG '93 (T. Atwood, J. Duhl, G. Ferran, M. Loomis, D. Wade)

ODBMS Definition - The ODBMS Manifesto, December 1989

DBMS

> Persistence, disk management, data sharing, data reliability, data security, query language

OO System

> Complex object, class, inheritance, overloading, object identity, encapsulation, extensibility

ODBMS is the merging of the two areas above.

ODMG '93 - The ODBMS Standard (http://www.odmg.org)

Founded in 1991 by Rick Cattel and five ODBMS vendors.

Guarantees the portability of applications written for object databases across most of the available object database systems.

Standardized interfaces:

> Object data definition language ODL
> Object Query language OQL
> C++ and Smalltalk bindings

Adopted by OMG (IDL, CORBA), currently studied by ANSI SQL3

The result of this work was a database where there is no need to copy and translate complex objects between the application environment and the DBMS.

Kurt Buehler, Open GIS Consortium

It is hard to agree on object models.

The emphasis must be on the interface and not on implementation
- the abstraction presented by the interface is more important than the implementation details
- i.e. vehicles should be objects even if this makes programming cumbersome

The emphasis must be on mechanisms and not on policy
- allow domain-specific views based on a consensus core of AUF (Andrew U. Frank) classes.

Don't throw away the past
- adapt data and systems to be part of migration to the future

Practice a wise use of resources, a hard learned lesson for OGIS

DMV lessons learned (from BYTE magazine)

Avoid technology for technology's sake

Involve staff (users) throughout the system early on
- consider the future: impending projects
- watch the connections: relationships change
- bottom line: is the mission being addressed?

Max Egenhofer, University of Maine, Orono; NCGIA

Response to Rick Church's question: Why do we need OO for transportation?

To model complex spatial relations the way that people think of them.

Secondarily semantic reasons
- OO is richer than other approaches in its ability to model complex phenomena
- it offers the prospect of fast implementations
- it can support multiple representations (views), dropping or adding detail

Benefits
- reduce the demands on the user to understand details of the implementation
- can do different analyses—roads vs. lanes vs. links
- the potential for graphic representation
- internal logical consistency

Methods of handling multiple representations
- on the fly: generalize from detail (too expensive / unreliable)
- stored: materialized views

Relation to OODBMS concepts

encapsulation: properties change with scale
overloading: different implementations

Made an analogy to verbally describing the method for tying shoes.

Michael Franklin, University of Maryland, College Park

What problems do OODBs solve?

Statement: “Objects let us think big.”
true, but this doesn’t mean you need an OODB.

So why OODB?
option 1: productivity gains
option 2: performance

OODB exploits client resources (CPU, memory, disk) to improve performance and scalability.

Question: How can your applications exploit the many and varied resources in the distributed, ITS environment (i.e., 12 sites or 12 million)?

Some database fundamentals

a database is a shared resource
it will be used by all existing applications as well as all future ones.

databases always grow
old data does not go away.

No hope if you plan to have a separate representation for every application.

e.g., fine vs. coarse grained access

you must use the finest granularity required by any application

Database systems address these issues using:

views: ANSI SPARC model (external / conceptual / physical)
physical view helps with implementation issue

integrity constraints
associated with the data

John Herring, Oracle Corporation

GIS is changing from file orientation to database orientation.

There is a proliferation of applications, where many applications run on the same data.

OO lets us build bigger and better systems.

OO makes programming easier, even bad programming.

An object structure that is optimal for one application may not be for another.

Indexing is necessary for queries of big databases.

Cannot optimize well with OO queries.

Encapsulation and inheritance are contradictory.

Object databases should be in SQL3.

Look carefully at available technology and decide the appropriate uses.

Are OODBMS extended virtual memory modules?

Shashi Shekhar, University of Wisconsin, Madison

The navigation decision is the basis of the problem—it underlies many ITS services that use navigable databases.

In our IVHS implementation in Minnesota we process images, text, and reports; support in vehicle units, TV cameras, and traffic sensors; and manage these systems through traffic control centers. While the product may be a map, the database is the central issue.

The question "Is OO the answer?" is the wrong one: the correct question concerns how to get interoperability and scalability. The answer must involve evolution, not revolution! We must choose the best technique for each problem, and not put blinders on! The cost of database software is relatively small, but data is expensive. OO may work for some problems and not others. Data is reality, but software will change, so it's important not to mix data with software.

Approaches based on objects and relations are the future. OO is already supported widely, with the exception of polymorphism, for example in SQL 93, Oracle 7, Sybase 10.

Modeling involves several separate levels:

- Conceptual: Entity-Relational
- Logical: object-relation or OO
- Physical: relational

There are too many standards
OGIS - GIS
SQL3/MM - Databases
GIS&T - Transportation
VRML - Virtual Reality

Consensus is hard to achieve in the standards world
Corba/OMG vs. OLE vs. ODBC

Many hard problems are unanswered

e.g. schema integration/conversion
data loss in conversion
quality issues

Conflicts will occur in schema integration: at the entity level, e.g., contours vs. triangulation, conversion; at the attribute level, e.g., position, lat/long, route, mile-post; and at the instance level, e.g., two objects with same location. Conflicts are of different types: synonyms, homonyms (1:1); overlapping concepts (1:N); and conflicts across schemas (M:N). An important question is: Does geography make the problem easier?

Various approaches are available to try to resolve these issues. We can do nothing, e.g., WWW; we can rely on visual integration by the user, through graphic overlays; we can establish a consistent location reference; consistent segmentation; consistent constraints; opt for schema integration and conversion; or develop and impose standards, using a common schema. If the last, co we need bottom-up efforts as well, specific to applications?

In summary: Interoperability means combining schema integration with portability and other properties; geometry can help, e.g., through consistent linear reference; geometry may even be able to simplify problems in other cases; application can help, e.g., WWW/VRML, visualization; and standards are only the first steps to full interoperability.

Alan Vonderohe, University of Wisconsin, Madison

A workshop was held recently in Milwaukee, attended by 42 participants (government, industry, and academia), with the objectives of preparing a draft consensus conceptual data model, at the entity-relationship level, for linear referencing systems; and of seeking a generic core data model that can be extended to meet specific needs of various application areas.

Key notions and significance:

A model describes requirements; it is not a specification.

There should be a clear distinction between a linear referencing method and a linear referencing system.

The number of fundamental operations with a map database is limited: locate, position, place, transform.

It is necessary to support multiple scales and multiple networks.

Cartographic representations have two elements: network representations, and linear references on the network.

Events are of two types: linear and point events. They will be tied to a datum object, composed of anchor points and sections.

There are many linear referencing methods.

An object model includes aggregate object classes, component object classes. other object classes, associations, and attributes

The following elements must be defined in order to implement a LRS in the ODMG-93 standard (based on the OMG Object Model):

Objects

Behavior

Set of Operations

State

Values for a set of Properties
Attributes
Relationships

Object Types

Categories of Objects with Interfaces

Datum, AnchorPoint, AnchorSection

Properties

Extents
Set of all Instances (indexed)

No Extents Declarations for Datum-Related Objects
May be Desirable for Instances of Derived Subtypes (Hydrology, Highways, RR, etc.)

Structured Objects
Collection
Set - no duplicates
Bag - allows duplicates
List - ordered bag
Array

Structure

Sets of AnchorPoints, AnchorSections

Lists of Links - Traversals

Also Versioning.

Conclusion: this framework allows for a generic linear referencing system for an object environment.

Michael Worboys, Keele University, UK

The concept of Resource Tracking provides an appropriate framework for thinking about the problem; the following discussion is based on a system developed for a hospital.

Key Generic Object Classes

Domain

Framework in which a given set of observers, informers and resources are located.

Observer

Associated with a single domain

Source of data about resources in its domain

Informer

Associated with a single domain

Provides information from the system to resources in the domain

Resource

Moves within and between domains

Tracked by observers in their domains

Event

Aggregations of movements of resources.

Primary objects of interest in the application domain.

The Spatial Framework

The underlying spatial framework of a navigable database is a dynamic, labeled, directed graph embedded in a coordinatized surface. Nodes and edges of the graph will have geometrical, topological and other attributes.

Node

geometry (surficial embedding)
topological (connectivity to other nodes, etc.)
other (traffic lights, etc.);

Edge

geometry (surficial embedding)
topology (connectivity to other edges at nodes, direction, turn restrictions)
other (classification, travel time)

Population of the Spatial Framework

Dynamic, collection of objects (vehicles, sensors, traffic lights, etc.)

Objects referenced to the underlying spatial framework

Objects classified as:
- observers
- informers
- resources

There is a requirement that the population is dynamic and the system is responsive to change.

On a longer time-scale, there is a need for history

Responsive GIS

Large amounts of spatially referenced data readily available

A sizeable proportion of the data regularly updated from external data sources

Some of the data are noisy, conflicting and incomplete

Rapid decision-making supported

Database Support for Dynamic Spatial Systems

requirement that the DBMS can
- respond to a dynamic scenario
- retain performance
- maintain integrity
- keep up-to-date

data captured from a heterogeneous collection of observers:
- noisy
- conflicting
- incomplete

deductive capability required to establish a coherent scenario that does not conflict with observations domain rules.

in order to build up such coherent patterns, history of past movements may be important.

Coordination and Control

Requirement to coordinate activities on the network

Controlling network behaviour to optimize desirable outcomes for the domain (e.g., an even spread of entities throughout the domain).

Remaining Questions

What is polymorphism?

Ad-hoc vs. parametric polymorphism

OO vs. OR vs. Wrapper

OO market is not the debit/credit market

MORNING PLENARY SESSION, FRIDAY, MARCH 15TH

Goodchild began asking the group whether every issue was on the table. He noted that the goal of the conference was to build a bridge between transportation and object-oriented database management systems (OODBMS). In pursing this goal, he reminded the participants that they must stay close to Caltrans requirements (e.g., a distributed database, interoperability). OO must make its case in the navigable database and intelligent transportation systems (ITS) arena. Goodchild then posed the following set of questions:

- Are we including all modes of transportation? Yes
- Will we have to support off-network information? Yes
- How will we handle dirty or incomplete data?
- What is the relationship between infrastructure and usage data?
- Will Caltrans need both planar and route information? Yes
- What are Caltrans requirements? Keep them in mind.
- Adopt a traffic operations perspective. The system will be installed in traffic management centers (TMC's).
- Will the database contain all streets or just major routes?
- What are the respective responsibilities of the public and private sectors?
- Is language an issue? How does one unambiguously express the problem?

In the subsequent discussion, it was noted that a modeling specification language for transportation problems had been designed by UC PATH. A description of this language is at http://www.path.eecs.berkeley.edu/nakash/, under "Auto high sys tools interface format".

Also, the point was made that there are many different types of network representations. One participant mentioned that his group had identified 10 network objects in their transportation modeling efforts, and that only four of these objects are currently supported by GIS vendors. Another point was made that Caltrans districts currently vary in their representation of the road network. This led to the point that data-sharing is an essential part of the object-oriented database movement in intelligent transportation systems.

BREAKOUT SESSIONS

Breakout sessions followed the plenary discussion. The group agreed that it would be useful to have further discussion on three topics: identification of the objects of a navigable database; criteria and procedures for evaluating alternative DBMS solutions, including OODBMS; and institutional issues affecting navigable databases and OO. The following sections provide highlights of the breakout discussions.

Breakout Session 1: What are the Objects?

The purpose of this break-out session was to brainstorm an initial list of relevant objects to support the development of a navigable database. The discussion began with the current state of our knowledge regarding objects in a transportation domain. The question was posed as to whether the objects should be viewed as hierarchical or simply related. John Sutton provided a list of initial objects that included:

shape points	vertex
arc	node to node
segment	shape point to shape point
section	one or more contiguous segments
route	multiple arcs
super-route	multiple routes
corridor	abstract collection of arcs

The point was made that there are other relevant objects like people, vehicles, roadways, and others that do not fit into the traditional transportation network view. Also, in regards to traditional transportation network data, there are existing data providers like ETAK and NavTech. The point was made that there are geometric vs. situational representations to consider as navigation is done by individuals in unique situations. Another perspective was offered that attempted to encompass the entire navigation problem. This perspective included object concepts like:

operations
aspatial objects
communications
people
decisions

Another perspective on objects was proposed that included the classes:

proximity
cartographic
flow

The point was made that we must consider the interface between the physical objects and the model objects. Physical objects are abstracted based on a selection of their characteristics. What characteristics of a vehicle need to be part of the vehicle object? Also, what are the roles of an object in the overall model; for instance, dynamic road signs.

The discussion moved to a series of questions regarding transportation in general:

• What are we transporting?	people/commodities
• What is doing the transporting?	conveyances
• What are we transporting on?	travelways (connectivity/infrastructure)
• When are we transporting?	time (dynamic/transaction/data/world)

The point was made that information granularity must be considered. Is the transportation world comprised of vehicle fleets or traffic flows?

The question of the current state of transportation modeling was reintroduced. The vendors have a particular viewpoint, but there are many scenarios that they don't support like non-planarity and hierarchical objects and tasks.

A point was made that there are no formal semantic specifications of the problem and that there are many levels to consider (geometric, mathematical, conceptual, real world). Object orientation is valuable because it buys the ability to generalize about complex problems when they are well specified. The objects must reflect the complexity of the system being modeled.

The group decided that Caltrans should develop an initial object-oriented model to support navigation and that the information exchange regarding this development should be high.

Breakout Session 2: Comparative DBMS Evaluation

The purpose of this breakout session was to examine the relative merits and drawbacks of the various database management systems (DBMS) options, primarily object-oriented vs. relational.

The discussion began with a series of questions:

- What is the state of the art?
- What are the DBMS alternatives?
- What can the research/commercial communities do for us?
- What recommendations can we make?

The point was made to focus on navigable databases. A working definition of a navigable database was given as a database comprised of fused information from a variety of sources to solve pathfinding problems, which includes real-time update and is a standalone part of a larger database.

A key distinction between a traditional GIS database and a GIS-T database is the temporal update. GIS databases are regarded as relatively static, where GIS-T databases are highly dynamic. A navigable database would have to support real-time update.

The point was made that a navigable database is generally a network model of the real world and this should play an important role in the evaluation of database alternatives regarding performance and also database design and administration reasons.

The focus shifted to a discussion of the definition of real-time. One approach is to differentiate hard real-time from soft real-time. In hard real-time, if the information is not where it should be by time X, we crash. This is not really the navigable database situation. In soft real-time, we get 10 bazillion sensor readings every 30 seconds, so we must be able to process 10 bazillion updates per 30 seconds. This is more the situation with a navigable database.

A discussion of the differences between a navigable database and a common corporate database ensued. The question came up as to how the functionality should be divided regarding centralization and distribution. This lead to a discussion of the key entities and functionality and how to define them. It might be useful to separate information to help the traveler from information for traffic management.

The discussion turned to the issues of distribution, replication, and heterogeneity of systems, and what needs to be considered to support a large-scale navigable database in these areas.

The point was made that there is also an issue of accuracy in location referencing, and this brought the discussion back to defining a navigable database. A navigable database 1) is optimized for one application or one group of interrelated applications 2) has a location referencing system, and 3) has on-line updates. It provides a materialized view of the real-world for pathfinding.

Breakout Session: Institutional Issues

The purpose of this breakout session was to examine the institutional issues that might arise in Caltrans efforts to deploy a statewide navigable database to support intelligent transportation systems (ITS).

The discussion began with the point that there are no state funds for local transportation projects. This led to the question as to Caltrans' role regarding ITS projects. Basically, Caltrans' role is to support ITS projects. These projects are generally funded as 10% local money, 10% state money, and 80% federal money. Caltrans' role is to deploy and demonstrate these ITS projects. The question arose as to whether Caltrans is in a coordinating role, which led to a discussion of regional planning councils (RPCs). These RPCs manage federal monies, where 80% of the federal money is passed through 6 RPCs. But what is the role of RPCs regarding ITS?

Regarding many of the transportation technologies already deployed, the point was made that 60-70 percent of all smart street lights are currently in "dumb" mode. It was noted that Caltrans is a state agency acting as a coordinating body for RPC technological advance. Tele-commuting, tele-shopping, and tele-education are three example areas where this sort of coordination is already taking place.

The question arose as to how much coordinating needs to be done. For example, assume an RPC wants to implement an OODB. The issue of data-sharing immediately arises. How difficult is it to share data? What are the advantages of OODBMS sharing over file sharing? The point was made that a lot of ITS strategies tend to be local, excluding commercial vehicle operations (CVO).

The discussion turned to the goal of ITS: to improve safety, health, and the efficiency of existing transportation systems. To accomplish these goals, the data need not be in one place. Example ITS projects were discussed including one with the goal of integrating an automated traveler information system (ATIS) with an automated transportation management system (ATMS). ITS kiosks were also broached, where the goal is to optimize individual trips via a distributed database with various search engines.

The focus shifted to open architectures and their necessity in supporting a navigable database. What are the hardware problems? The software problems? How do these problems relate to the institutional problems? The prediction was made that databases will have to become truly generic.

The focus was redirected to the current state-of-the-art in database management systems. The point was made that semantics are the real bottleneck in data-sharing. That means that we must have standardized meaning. In this regard, the emphasis should be on developing the method of sharing and not the means.

Another point was made that we must make use of what we already have in place (i.e. transportation management center elements). The counter-point was put forth: should we start from scratch? Another approach was proposed where we examine what information we need and compare it to what information we can get. A common assumption was broached: technology will solve its own problems. The group agreed that institutions should not adopt standards that are based on current technology. In this regard, institutions should focus on standards that are invariant to technological change. One example of this would be conceptual views over physical views. This would lead to the recommendation to minimize investment in current technology, but maximize adoptability to current technology.

A concrete strategy was proposed where an institution would invest in data over investing in hardware. A simple classification was proposed of investment strategies for data, hardware, software, and current off-the-shelf (COTS) technology. Data should be a long-term investment, but hardware and software should be short-term investments. COTS technology should be favored over in-house development.

The point was made that different databases may be needed at different levels. There are jurisdictions that abut, overlap, and are related hierarchically. A common database with a common data dictionary is needed to support multiple applications.

There are issues regarding several standards efforts ongoing (OGIS, ISO, STDS). For example: what is a road in the real world and as a digital representation? In general, OGIS is not a network model. For Caltrans, the data dictionary a key issue.

Also, the issue of static data exchange versus dynamic data exchange. What will be the static base map to support ITS? Database time and real-world time will also complicate the database (e.g., historic traffic information). To address many of these problems, metadata will be critical.

Caltrans is expected to have a coordinating role toward the regional planning councils (RPC) and metropolitan planning organization (MPO).

What do we mean by data dictionary? What is the appropriate investment strategy? What will be our measure of success? Standards need to be addressed. Where will the definitions come from?

There are resource problems for representing transportation networks (modeling). Custodianship is also a key institutional issue. Maintenance of data should be close to originator, configuration management should be adopted, and agreement should exist among the various data providers as to what the objects are.

Quality assurance regarding data and metadata should be in place.

Breakout Reports

Following the morning break-out sessions, the group reconvened to listen to the group leaders present the results of the break-out discussions. Here are the three reports.

What are the Objects? (David Fletcher)

There are two primary domains that exist regarding objects: physical objects and model objects. Model objects have proximity, cartographic representations, and attributes (e.g., flow). Example physical objects might be a place, sign, person, roadway, and vehicle.

Transport is concerned with movement:

- What is being moved? People and commodities.
- Where is it being moved? Travelways between places.
- How is it being moved? Conveyances.

Granularity is a significant semantic issue to resolve. Are we moving vehicles, fleets, or flows?

Where are we?

- Transport scenarios are not supported in many existing products.
- Transport = (people/goods) moving between (places) using (conveyances).
- The problem suggests multiple levels of granularity.
- Granularity is determined by the task and responsibility.
- No formal semantic definition exists for this domain.
- We need to separate "real world" objects from geometric mathematical objects.
- OO is necessary to understand complex problems.
- Tasks are dynamic and interdependent, and so are the objects.

Where do we need to go?

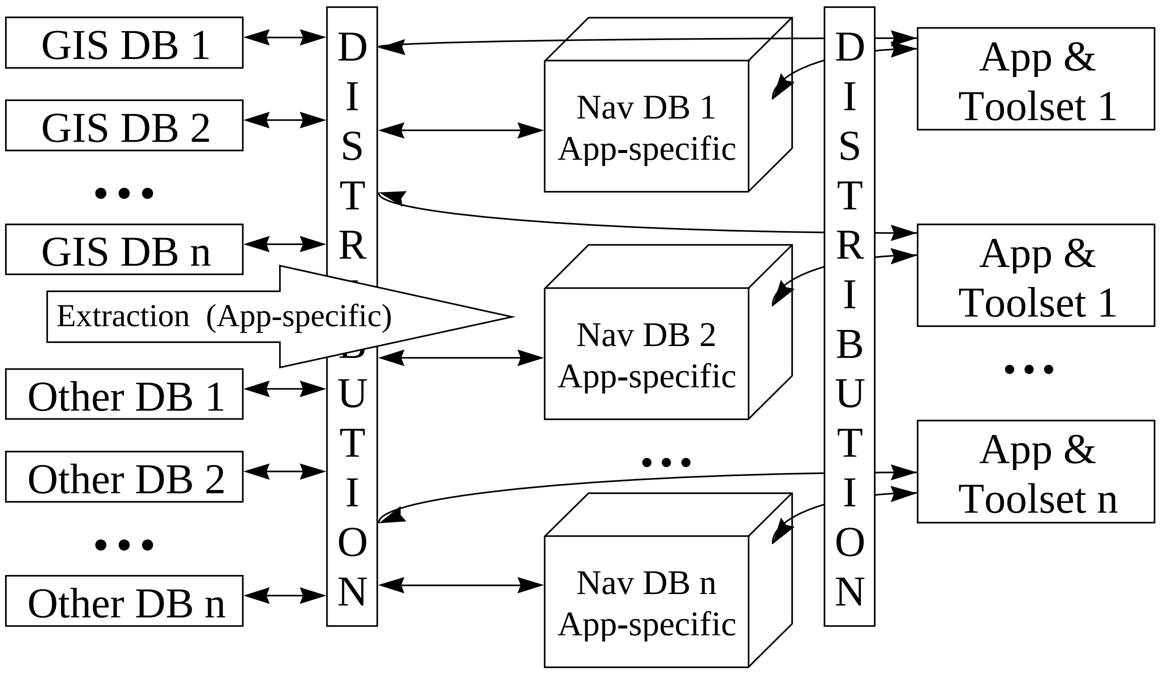
- Develop formal semantic specifications.
- Reconcile/integrate existing models
- Identify fundamental objects.

Question posed to entire group for discussion: What should Caltrans do?

Comparative Databases: The good, the bad, and the ugly (Kurt Buehler)

We should pursue a generalized architecture (see figure below), one that encompasses multiple GIS and non-spatial databases alike. This generalized architecture would support a navigable database capable of serving multiple applications. The myriad GIS and non-spatial databases would be integrated via a distributed layer that would exist between the component DBs and the applications that are served. Another layer above the applications would distribute separate toolsets.

A Generalized "Architecture"



Scenario #1 - Tourist Information

In a typical tourist information scenario, an in-car traveler information applications would access a traveler information navigable DB. The navigable DB would then query a traveler information navigable DB extraction tool that would access the distribution layer above the component DBs to assemble the information necessary to serve the original tourist query.

Scenario #2 - Traffic Management

Similarly, in a traffic management scenario, a traffic management application would make a request to the navigable DB, which would then rely on a traffic management extraction tool to assemble the necessary information from the component DBs below the distribution layer to serve the original query.

Where is DBMS Capability Required?

DBs required in several places

- Extraction Source DBs
- Real-time update/filter DBs
- NavDB
- Client (application side) DBs

What are the Required DBMS Characteristics?

- Distributed
- Replicated
- Heterogeneous
- Active (at least support a trigger type mechanism)
- Asynchronous (for some application contexts)

What is a Navigable Database?

- A DB optimized for path-finding and/or evaluation
- Location transparent
- Capable of on-line (real-time) updates

Legend for the Subsequent Database Comparison Summary Charts

<u>Database Types</u>

- OO - object oriented database
- OR - object-relational database
- R - relational database
- C - composite

<u>Ratings</u>

- G - good
- B - bad
- U - ugly

Subjective* Ratings of Suitability for NavDB by Functionality

Functionality/Type	OO	OR	R	C
A lot of fine grained access	G	G	U	B
Reliability	G-	B-G	G+	B+
Availability	G-	B-G	G+	B+
Integrity checking	U[G]	G-	G	U
Value update (realtime)	G	G	G+	G
Network update	G+	G	G-	U

Subjective* Ratings of Suitability for NavDB by Functionality

Functionality/Type	OO	OR	R	C
Value Query	G	G+	G+	U
Navigation Query	G+	B-G	U+	U
Back-end processing	G-U	G-U	U	G
Push-based data delivery	U	G	G	U
Scaleability - Data Size	U	G	G	U
Scaleability - Data Complexity	G	G	U	U

Subjective* Ratings of Suitability for NavDB by Functionality

Functionality/Type	OO	OR	R	C
Scaleability - # clients	U-G	G	G	U
Views	U	G	G	U

Other comments

Maybe there are application or interoperability architectures that provide the capabilities/functionality required without forcing a DBMS decision.

Location referencing standardization needs to be addressed somewhere.

Institutional Issues (Alan Vonderohe)

What do we know about the context for developing a navigable database to support ITS?

Institutional context

Funding breakdown

Fed - 80%
State - 10%
Local - 10%

In California

Regional Planning Councils (RPC) manage Federal $$
80% goes through 6 RPC's
Caltrans has a coordinating role
Southern Cal. Corridor Steering Committee
Caltrans districts and headquarters participating

Data Sharing (Jurisdictions abut and overlap)

Local <-> State <-> National all share static databases
Local <-> Local share real-time and static databases
Local <-> Vehicle share real-time data

Technological Context

Traveler Information System (TIS)

Public Kiosk
Point A to Point B
Optimize individual trips across distributed databases
Search Engines are Required

Integration of TIS and Automated Traffic Management (ATM) Systems

Open Architectures
Interoperability becoming a commodity
Internet is lowest common denominator
Need for higher common denominator
Fully distributed but more tightly coupled

Recommendations to Caltrans

Assumption: Technological advances will solve interoperability problems.

Do not adopt standards based on current technology.

Maximize adaptability

Investment Priorities

Data (long-term)

Hardware (short-term)

Software
Current off-the-shelf (COTS) first
In-house second

The key to success is the determination of what data (objects) need to be shared.

Focus on mission

Focus on Semantics

Invariant under technological change

Develop conceptual views rather than physical views

Strive for common conceptual data model (based on semantics)

Custodianship

Maintenance as close to originator of data as possible

Configuration management

Quality assurance / metadata

- May need minimal standards beyond “truth in labeling / fitness for use”

Schema Transfer

Closing Remarks

Following the presentations by the break-out leaders, the panelists, keynote speakers, and organizers of the conference were asked to make closing remarks. Following is a summary of this portion of the conference.

François Bancilhon

There is a distinction between "object-orientation" and "complex object manipulation. "Object orientation" means you support C++, Smalltalk, or OMG, or functions/languages of that type, whereas complex object manipulation is a more general reference to performance.

What needs to be done

- Definition of a class library to support a navigable database
- Definition of bench marks:
  - database
  - application
  - metrics (time)

These will help focus on the goals and issues involved in developing a object interface on a database engine.

Kurt Buehler

Important points

- Focus on interface, not on implementation
  - abstraction above implementation details
  - mechanism, not policy
- allow domain-specific views based on a consensus core of "AUF" (Andrew U. Frank) classes.
- Don't throw away the past
  - adapt data and systems to be part of migration to the future.

Remember the DMV lessons learned:

- Avoid technology for technology's sake.
- Involve staff (users) early on.

Consider the future: impending projects.

Watch the connections: relationships change.

The bottom line: is the mission being addressed?

Max Egenhofer

I would like to note a few problems:

We have been mixing conceptual, implementation, and performance issues. We should define what the benchmarks should indicate before deciding what they are.

I have noticed major difficulties in language/semantics in our discussion. The semantics of the problem need to be addressed first.

There is a clear lack of tools to accomplish our goals.

Positives:

Our application domain is specific and focused.

We have recognized the complexity of problem: from many different perspectives, all of which must coexist in an application context.

Michael Franklin

Everyone has their own concerns and own perspective. His perspective, for example, is as one who deals with database issues.

We need to understand:

How fast does data change? This will affect design. Need to determine how fast it will change (at least order of magnitude). Example: replication is feasible if data is static, not if changing rapidly.

How tight are timing constraints? How important is it that data is up to date? This will affect which technology is appropriate.

What is consistency? How much tolerance for error is there? Will be problems from sensors, etc. Connects with timing issues.

What resources are available to the system? Solution for PDA will be different than base station workstation.

How to deal with heterogeneous data?

These issues need to be addressed before a system can be judged for appropriateness

Also, we need boundaries. Definition of what is and is not a navigable database, for example.

Alan Vonderohe

What generally comes out of meetings like this is more questions that need to be answered. We need to start working on the semantics and the definition of the conceptual data model.

Shashi Shekhar

Reiterating points:

A conceptual data model is central. Once the conceptual model is right, then one can translate to objects.

Visual representations should be included in the model, and be consistent.

There is a distinction between continuous (reality) and discrete data (coverage). A connection between two should be in the model.

Objects are inherently discrete. New concepts are needed to model continuous roads.

There is a need for data models, manipulation functions, and query languages for graphs.

Other issues should be secondary

John Herring

We need to realize that the goal here is a very complex system, so we can't go out and just design it. Because it:

must be distributed

must be "heterogenous up the gazoo"

must be dynamic, both data and system, because it will be unpredictable

has requirements that will grow to meet its capabilities

What is useful?

Object modeling must be used.

Distributed computing and interoperability.

Both object-oriented and object relational approaches are necessary.

Need integrated development environment.

“The right tool at the right place.”

"If you look at the world in terms of nails, you're going to get screwed."

David Fletcher

What is object-orientation? It is a way of thinking about things.

Difficulty is changing perspectives, of how we look at processes and data.

How to engineer complex software to deal with problems

A navigation database includes policy, procedure, process, products, for assisting the movement of people and goods from one place to another safely, efficiently, and equitably.

Object orientation includes the perspective, principles, methods, tools, technologies for understanding complex problems and engineering complex software.

Navigational database:

results from public policy

needs collaboration among many people
- policy makers
- citizens
- transportation planners and engineers
- system developers & technology providers
- shippers & carriers
- researchers

Michael Worboys

It appears that we have a primary domain that is complex and permanent that interacts with an observer, an informer, and inhabitants.

We require a means of conceptualizing and reasoning with these entities.

An unambiguous specification must be pursued that is OO.

Data fusion is the goal: bringing together all the real-time, inconsistent, and incomplete data from the informers and providing it in a seamless manner to the observers.

Andrew Frank

What was the problem?

Data exchange

Integration at different levels of aggregation

Shared database is necessary because information must be shared—distributed database

may be the result of this requirement.

It is not well understood how hierarchies and processes interact.

Rick Church

The original problem was object-orientation in a navigable database. It is clear from the discussion of the comparable databases group that databases (plural) are required. This may have been the reason for the problem in the object definition group—the goal was not clearly defined.

Important issues:

- Dirty data
- New meaning to the term "wrapper"
- Legacy issues cannot be ignored

APPENDIX I: LIST OF PARTICIPANTS

Conference Organizers:

Dr. Michael F. Goodchild
NCGIA/UCSB Geography
3510 Phelps Hall
University of California
Santa Barbara, CA 93106-4060
good@ncgia.ucsb.edu

Dr. Richard L. Church
UCSB Geography
3611 Ellison Hall
University of California
Santa Barbara, CA 93106-4060
church@geog.ucsb.edu

Ramez Gerges
Testbed Center for Interoperability
1701 South Hall, UCSB
University of California
Santa Barbara, CA 93106-4060
ramez@introp.transcal.ca.gov

Danette Coughlan,
NCGIA
3510 Phelps Hall
University of California
Santa Barbara, CA 93106-4060
coughlan@geog.ucsb.edu

Tom Cova
NCGIA
3510 Phelps Hall
University of California
Santa Barbara, CA 93106-4060
cova@geog.ucsb.edu

LaNell Lucius
NCGIA
3510 Phelps Hall
University of California
Santa Barbara, CA 93106-4060
lanell@ncgia.ucsb.edu

Core Panel:

Dr. Francois Bancilhon
O2 Technology
3600 West Bayshore Road, Suite 106
Palo Alto, CA 94303
francois@o2tech.com

Kurt Buehler
Open GIS Consortium
357 B West Morgan St.
Spencer, Indiana 47460
kurt@ogis.org

Dr. Max Egenhofer
National Center for Geographic Information and Analysis
and Department of Spatial Information Science and Engineering
Department of Computer Science
University of Maine
Orono, ME 04469-5711
max@spatial.maine.edu

Dr. David Fletcher
Alliance for Transportation Research Institute
University of New Mexico
Albuquerque, NM
fletcher@unm.edu

Dr. Andrew Frank
Geoinformation E127
Technical University of Vienna
Gusshausstrasse 27-29
A-1040 Vienna Austria
frank@geoinfo.tuwien.ac.at

Dr. Michael Franklin
Department of Computer Science
A.V. Williams Building
University of Maryland
College Park, MD 20742
franklin@cs.umd.edu

Dr. John Herring
Oracle Corporation
196 Van Buren Street
Herndon, Virginia 22070
jrherrin@us.oracle.com

Dr. Shashi Shekhar
University of Minnesota
4-192 EE/CS Bldg.
200 Union Street S.E.
Minneapolis, MN 55455
shekhar@cs.umn.edu

Dr. Alan Vonderohe
1208 Engineering Hall
1415 Engineering Drive
Department of Civil and Environmental Engineering
University of Wisconsin - Madison
Madison, WI 53706
vonderoh@coefac.engr.wisc.edu

Dr. Michael Worboys
Department of Computer Science
Keele University, Keele,
Staffordshire ST5 5BG, U.K.
michael@cs.keele.ac.uk

Other Participants:

Teresa Adams
Civil and Environmental Engineering
University of Wisconsin - Madison
1415 Engineering Drive
2208 Engineering Hall
Madison, WI 53706
adams@antigo.cee.wisc.edu

Divyakant Agarwal
Dept. of Computer Science
University of California Santa Barbara
Santa Barbara, CA 93106
agrawal@cs.ucsb.edu

David Arctur
Laser-Scan, Inc.
45635 Willow Pond Plaza
Sterling, VA 20164
arctur@acm.org

Masatoshi Arikawa
Hiroshima City University
Japan (NCGIA visitor)
arikawa@rsru.geog.ucsb.edu

Scott Bell
UCSB Geography
3611 Ellison Hall
University of California
Santa Barbara, CA 93106-4060
bell@geog.ucsb.edu

Stephen Bespalko
Sandia National Laboratories
PO BOX 5800, MS 0718
Albuquerque, NM 87185-0718
sjbespa@sandia.gov

Rob Bootsma
Testbed Center for Interoperability
1701 South Hall
University of California, Santa Barbar
Santa Barbara, CA 93106
rob@transcal.ca.gov

Daniel Carlson
National Engineering Technology Corporation
14320 Firestone Blvd.
La Mirada, CA 90638
dcarlson@la.nateng.com

Vicki Cobb
Caltrans, New Technology and Research

Wils Corrigan
UCSB Geography
3611 Ellison Hall
University of California
Santa Barbara, CA 93106-4060
corrigan@geog.ucsb.edu

Kevin Curtin
NCGIA/UCSB Geography
3510 Phelps Hall
University of California
Santa Barbara, CA 93106-4060
curtin@geog.ucsb.edu

Craig Dean
Sandia National Laboratories
P.O. Box 5800 MS 0449
Albuquerque NM 87185-0449
cddean@sandia.gov

Craig Denison
Caltrans, District 7 Office
MS: DOB RM 127
120 S. Spring St.
Los Angeles, CA 90012
cdenison@trmx3.dot.ca.gov

Zachary Deretsky
Etak, Inc.
1430 O'Brien Drive
Menlo Park, CA 94025
zach.deretsky@etak.com

Akash Deshpande
California PATH, UC-Berkeley
Richmond Field Station, Bldg. 452
1301 S. 46th St.
Richmond, CA 94804
akash@shadow.eecs.berkeley.edu

Amr El Abbadi
Department of Computer Science
University of California
Santa Barbara, CA 93106
amr@cs.ucsb.edu

Pete Fohl
NCGIA/UCSB Geography
3510 Phelps Hall
University of California
Santa Barbara, CA 93106-4060
fohl@geog.ucsb.edu

Reginald Golledge
UCSB Geography
3611 Ellison Hall
University of California
Santa Barbara, CA 93106-4060
golledge@geog.ucsb.edu

Cecil Goodwin
Oak Ridge National Lab
goodwinc@utk.edu

Dale Honeycutt
Environmental Systems Research Institute
dhoneycutt@esri.com

Karen Kemp
3510 Phelps Hall
University of California
Santa Barbara, CA 93106-4060
kemp@ncgia.ucsb.edu

Philip Kilby
CSIRO/DIT
GPO Box 664
Canberra ACT 2601
AUSTRALIA
pjk@cbr.dit.csiro.au

Mei-Po Kwan
Department of Geography
The Ohio State University
1036 Derby Hall
154 N. Oval Mall
Columbus, OH 43210
mkwan@magnus.acs.ohio-state.edu

John Landis, UC TC
University of California, Berkeley
landis@ced.berkeley.edu

David Lemberg
UCSB Geography
3611 Ellison Hall
University of California
Santa Barbara, CA 93106-4060
lemberg@geog.ucsb.edu

David Lively
Caltrans Traffic Operations Program
PO Box 942874 MS-36
Sacramento CA 942874.
dlively@trmx3.dot.ca.gov

Hong Lo
Department of Civil and Structural Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
cehklo@uxmail.ust.hk

Hideo Makino
Niigata University
Japan
makino@psych.ucsb.edu

Jeff McRae
Caltrans, Electrical Systems
Division of Traffic Operations
1120 N Street   MS-36
Sacramento, CA 95814
jmcrae@trmx3.dot.ca.gov

Yusuf Ozturk
Navigation Technologies
10400 W. Higgins Road, Suite 400
Rosemont, IL 60018
yozturk@ssi.navtech.com

Joe Palen
Senior Research Engineer
ATMIS Development
Caltrans New Technology
Box 942873, MS 83
Sacramento, CA 94273
japalen@dcn.davis.ca.us

Alex Rodarte
National Engineering Technology Corporation
14320 Firestone Blvd.
La Mirada, CA 90638
arodarte@la.nateng.com

Asfand Yar Siddiqui
Department of Transportation
New Technology and Research Program
Office of Advanced Highway Systems, MS 83
1227 "O" Street
Sacramento CA 94273-0001
asiddiqu@trmx3.dot.ca.gov

Steve Smyth
Microsoft Corporation
stevesmy@microsoft.com

Kermit Snelson
Manager, Field Data Capture Tools
Etak, Inc.
1430 O'Brien Drive
Menlo Park, CA 94025 USA
kermit.snelson@etak.com

Bruce Spear
Director of Geographic Information Services
Bureau of Transportation Statistics
400 Seventh Street, SW
Washington, DC 20590
bruce.spear@bts.gov

Jon Speigle
Psychology Department
University of California, Santa Barbara
Santa Barbara, CA 93106
speigle@psych.ucsb.edu

Robert Steinke
Mailbox 117
Dept. of Computer Science, UCSB
Santa Barbara, CA 93106
steinke@cs.ucsb.edu

John Sutton
GIS/Trans, Ltd.
2081 Business Center Drive, Suite 145
Irvine, CA 92612
jsutton@gistrans.com

Agatha Tang
ESRI
380 New York Street
Redlands, CA 92374
atang@esri.com

Nectaria Tryfona
University of Maine, Orono
nectaria@spatial.maine.edu

Paul Van Zuyle
NCGIA/UCSB Geography
3510 Phelps Hall
University of California
Santa Barbara, CA 93106-4060
vanzuyle@geog.ucsb.edu

Ali Zaghari
Caltrans District 12 - Traffic
2501 Pullman Street
Santa Ana CA 92705
azaghari@trmx3.dot.ca.gov

Ming Zhang
UC Berkeley
zhang@uclink2.berkeley.edu

APPENDIX II: POSITION PAPERS

Teresa M. Adams, Alan P. Vonderohe, Nancy K. Wiegand

Department of Civil and Environmental Engineering
University of Wisconsin
1415 Engineering Dr
Madison, WI 53706

Abstract

This paper explores implementation of a Linear Referencing System (LRS) model using features of next generation Database Management Systems (DBMSs) and the proposed ODMG-93 standard for Object-Oriented DBMSs (OODBMSs). The paper presents preliinary concepts being developed through research in progress.

The Linear Referencing System

A data model for a Linear Referencing System (LRS) was developed by transportation professionals at a workshop convened for this purpose (Vonderohe et al., 1995) The data model addresses the problems of 1) integrating data referenced by alternative linear methods, 2) associating these data with multiple network representations for analysis, and 3) associating these data with multiple cartographic representations for display and linkage with two- and three-dimensional data. The LRS can be used in network-based applications such as transportation, hydrology, and utilities management. The purpose of the LRS is to coordinate applications and business data in which different linear referencing methods have been used. Examples of linear referencing methods for transportation are milepoint, reference post, and engineering stationing. Examples of transportation business data (events) are accidents, stop signs, or stretches of pavement needing repair.

The central notion in the LRS model is that of a linear datum, providing the fundamental linear space for transformations among referencing methods, linkage to multiple networks, and linkage to multiple cartographic representations. The linear datum allows field addresses for events, based on various linear referencing methods, to be standardized using offsets along linear objects (referred to as anchor sections) in the datum (Fletcher, 1995). Anchor sections begin and end at anchor points that can be identified and recovered on the ground. A field address for an event consists of an offset from a traversal reference point that, in turn, is located by an offset from the initial node of a traversal (an ordered and directed collection of links). Nodes, in turn, are mapped by offsets onto anchor sections. Standardized system addresses are calculated from successive offsets between or along events, traversal reference points, nodes, and anchor sections. A given linear referencing method consists of a collection of traversals, each with its own collection of traversal reference points.

Multiple network representations can be associated with a single datum because individual nodes are mapped to anchor sections. In a similar way, a single datum can be associated with multiple cartographic representations from different sources or at different scales. Individual linear cartographic objects are mapped to anchor sections by offsets. Thus, it is possible to link diverse application domains such as infrastructure management, transit emergency, and intelligent transportation systems (ITS). We feel the model can be generalized to accommodate numerous other linearly referenced civil and environmental applications.

IIDBMS Implementation

The LRS model provides a basis for a software system that manages large stores of persistent data. The motivation for considering next generation database management (Cattell, 1991) is the desire to benefit from the features of extensibility. Extensibility is the ability of a DBMS to include application-specific code, including new data types and operators. User-defined types are required to define spatial data types.

In this paper, we describe modeling the LRS using features of the ODMG-93 (Cattell, 1993) standard for OODBMSs. The ODMG-93 (Object Database Management Group) standard uses the OMG (Object Management Group) Object Model as the basis. The ODMG comprises member database and software companies committed to supporting the standard.

In the OMG Object Model, the basic modeling primitive is the *object*. Objects are categorized into *types*. Each type has one *interface*. The interface defines the external properties and operations supported by instances of the type. In the LRS model, for example, interfaces are defined for Datum, AnchorPoint, AnchorSection, Network, Link, and Node object types.

Each type is itself an ojbect with properties. One property is the supertype/subtype graph. Another is the *extent* of an object type. The extent designates the set of all instances of a given type and causes an index on the member of the set to be automatically created and maintained. The initial model has one Datum, AnchorPoint, and AnchorSection class. There are no extent declarations for Datum, AnchorPoint, and AnchorSection because some are for roads, hydrology, RR, etc. It may be desirable to specify derived subtypes for hydrology, roads, RR, etc. The decision may be driven by the desire to have extents for the instances of each derived subtype. Network, Link, and Node are not grouped by extents; they are grouped by relationships with other object types.

All objects of a given type exhibit common behavior and a common range of states. The behavior is defined by a set of *operations* that can be executed on an object of the type. The state of objects is defined by the values they carry for a set of *properties*. The properties can be either *attributes* of the object itself or *relationships* between the object and one or more other objects.

The relationships in which objects of a given type participate are specified as a set of *relationship signatures*. Each signature defines the type of the related object or set of objects and the name of a function used to refer to the related object or set of objects. The cardinality of a relationship can be one-to-one, one-to-many, or many-to-many.

Relationship declarations for the AnchorPoint class are illustrated in Figure 1 using the Object Definition Language (ODL) of the ODMG-93 standard. Verbally, the relationships are expressed as : each AnchorPoint is part of a Datum; each AnchorPoint is the beginning of a set of AnchorSections; and each AnchorPoint is the end of a set of AnchorSections. In the ODMG-93 standard, optional system-maintained *inverse* relationships are also supported. For example, the inverse of the relationship "is_part_of" is "includes". Inverse relationships and their cardinality are defined in the target object class definition.

```
AnchorPoint         //relationship signatures
{
  relationship Datum is_part_of
    inverse Datum::includes;
  relationship Set<AnchorSection> is_beginning_of
    inverse AnchorSection::begins_at;
  relationship Set<AnchorSection> is_end_of
    inverse AnchorSection::ends_at;
}
```

Figure 1: Relationship Signatures for the LRS AnchorPoint Object Type

The OMG Object Model accommodates built-in *collection* types. A collection may be, for example, a *set* containing no duplicates, a *bag* which allows duplicates, a *list* which is an ordered bag, or an *array*. Named instances of these types can also be used to group objects. We envision the *list* collection type as useful for the relationship between traversals and links of a network. In the LRS, each traversal is associated with an *ordered* set of network links.

AN important consideration for the LRS model is the necessity to provide for updates. For this, we are considering the use of *versions*. Versioning is a DBMS mechanism to reference previous states of data. For example, in the LRS, a datum with its anchor points and anchor sections is considered to be static. However, if a roadway is re-aligned, the underlying datum changes. To accommodate such changes, datum objects could be versioned. Each version of the datum contains pointers to the then-current anchor sections and anchor points. It is desirable for some component objects to remain active through many versions.

Summary

This paper presents some capabilities of next generation DBMSs to support a Linear Referencing System. In particular, we focus on identified object types, extents, and relationships of the LRS. Our schema uses the ODMG-93 standard for Object-Oriented DBMSs, and takes advantage of the capabilities not available in standard relational DBMSs.

Acknowledgments

This work was partially supported by NSF award CMS 94-02478; Dr. Ken Chong, cognizant program director.

References

Cattell, R.G.G. (1993) *The Object Database Standard: ODMG-93.* Morgan Kaufman, San Mateo, CA. With contributions by T. Attwood, J. Duhl, G. Ferran, M. Loomis, and D. Wade.

Cattell, R.G.G. (1991) What are next-generation database systems? *Communications of the ACM* **34**(10): 31-33.

Fletcher, D. (1995) Personal communication.

Vonderohe, A.P., C-L. Chou, F. Sun., and T.M. Adams (1995) Results of a workshop on a generic data model for linear referencing systems. *AASHTO Symposium on GIS-T, Sparks, NV, April*.

François Bancilhon
O2 Technology
3600 West Bayshore Road, Suite 106
Palo Alto, CA 94303
francois@o2tech.com
http://www.o2tech.com

OBJECT DATABASES

Object databases appeared in the 80's, first as prototypes, then as commercial products. This new generation of database systems differs from the relational one by (1) supporting an object data model instead of a relational one and (2) integrating closely with object programming languages, instead of separating the database from the program. This paper covers the main concepts of object databases: object data model and classical database features, query language, programming language binding, persistence, performance and standards.

Object database concepts: the object model

An object database system is a database system that supports and object model. These object models all include the following features, [Atkinson et al. 89]:

Complex objects: Objects are built from atomic objects (integer, real, Boolean, string, bit, etc.) by recursively applying object constructors: tuple, set, bags, lists and arrays.

Relationships: The tuple constructor allows for the definition of relationships between objects. These relationships can be one-way or two-way.

Object identity: Each object has its own identity, which is independent from its value. Through object identity, objects can share subcomponents and objects can even be cyclic.

Encapsulation: An object consists of an identity, a value and a set of operations (the methods). The object can actually be manipulated through the operations. The only way an outside operator can update or read an object is through the methods, thus the internal structure of the object is not visible from the outside

Classes: Objects with the same internal structure and the same set of methods can be grouped in classes. Thus a class has an interface description and an implementation description.

Inheritance: Classes can be specialized by adding new attributes, new relationships and new methods. The set of classes forms a directed cyclic graph.

Message passing: The binding of the name of the method is actually done at run time (late binding) so as to insure that the proper implementation of the method takes into account the actual class to which the object belongs.

Classical database concepts

Because they are database systems, object database systems also support the same type of functionality as traditional database systems:

Transactions and concurrency: the minimum level of functionality is the support for ACID transactions. More elaborate modes of transactions can be offered to support design applications.

Recovery: warm (in case of main memory failure) and cold (in case of disk failure) restart are provided. On line back up of the database is also a key feature.

Distribution: client/server architecture is the minimum required, support for two phase commit and replication are other options.

Version management: support for object version creation, maintenance and use must be provided.

Schema evolution: the database administrator must be able to modify the database schema, while keeping the same database data.

Object Query Language

Object databases, like traditional databases, need a query language. The SQL query language cannot be used for that purpose because it lacks support for (i) complex objects (ability to query nested structures and arbitrary collections) and (ii) method invocation.

The OQL language is currently accepted as the standard object query language.

It supports typical join/select/project queries: For instance, this query prints the manager of Employee John:

```
select d.manager
from employees e, department d
where e.name = "John" and e.dept = d.name
```

It also supports navigation in objects: This query extract the first name of every person who lives in France.:

```
select p.name.first_name
from persons p
where p.address.country = "France"
```

Of course, it allows method invocation on objects: This query prints the age of every person in the database. This is computed from the birth date by a method "age":

```
select p->age
from persons p
```

Due to the relationships between objects, pointer joins are supported: This query prints the name of the students attending a course taught by Turing.

```
select s.name
from professors p, p.courses c, c.students s
where p.name = "Turing"
```

And finally, the language allows for complex structures construction: This query builds a set of complex objects with attributes "name" (of type string) and "children" (of type set of strings).

```
select struct(name: p.name, select c.name from p.children c)
from persons p
```

Language bindings

Language bindings provide the ability (i) to write the methods of a class in a given programming language and (ii) to develop an application using an object database. Object databases, instead of introducing new programming languages, chose to provide language bindings to existing object programming languages, essentially C++ and Smalltalk. The language bindings specify how (i) to declare persistence, (ii) to manage persistent objects, (iii) to embed queries, (iv) to represent and manipulate all the features of the data model and (v) to manage databases and transactions.

Persistence

In a traditional database, there is a "wall" between the database system and the application program. The database contains only persistent data, i.e., every modification done to the database by a committed transaction remains until another transaction eventually modifies it again. The application program only manipulates transient data, i.e., all the data it operates on will disappear after the end of the execution of the program. Interaction between the application and the database is always done through read an write statements. The program can read data from the database, insert new data in the database or update existing data in the database.

Object databases solve this problem differently: they use the persistent programming approach. In this approach, there is no separation between the database and the application work area: the application program manipulates both persistent and transient data. Persistent data is updated in transactions and its modifications are persistent once committed. Transient data has the life span of the application. Orthogonal persistence means that (i) the type system of the persistent and transient space are the same (in other words, every object of every type can be made persistent) and (ii) the same operations can be done on a transient and a persistent object. A model of persistence specifies how objects are created in each space and whether they can change their persistence status.

Performance

Object databases are currently targeting specific application areas, mainly those manipulating complex data. A number of benchmarks have been proposed to represent this type of work load, the best known are the 001 benchmark [Cattell and Skeen 92] and the 007 benchmarks [Carey et al 93]. 001 was mainly used to compare object databases to relational databases. It shows that on this type of workload, object database systems outperform relational systems. 007 was mainly used to compare object database systems to one another.

Performance improvement over relational systems can be easily explained

- The database engine is specifically designed to support the object model. It offers native support for complex objects and avoids the join operation of relation systems when building these objects.
- Object databases were designed with a client-centric architecture in mind. Thus they transfer as much as possible of the workload onto the client, and support client caches for objects.

Standards

In 1992, ODMG, the Object Data Management Group, was created at the initiative of the object database vendors with the goal of defining and promoting a portability standard for object databases. This will ensure that developers can port their applications from one compliant system to another at a minimal cost.

The standard has five main elements:

- a data model, compatible with the OMG data model;
- ODL, an object definition language for schema declaration. ODL is an extension of the OMG IDL language;

- OQL, the Object Query Language, was adopted by ODMG as the standard query language;
- bindings to the C++ and Smalltalk languages.

The ODMG model supports both a model of persistence on creation (persistence status is determined at object creation and never changes) and a model of persistence by reachability (objects reachable from persistence roots are persistent).

Several drafts of the standard were published starting in 1993, the last one being ODMG 1.2 in 1995 [ODMG 95]. Compliant products began to appear in 1995 and it is safe to predict that most companies will soon be shipping ODMG compliant products.

Conclusion

Object database technology has now reached a level of maturity, with a complete set of features supported by a variety of commercial products. The systems are in use in a set of specialized applications: CAD, Telecom, GIS, Financial Industries, and Multimedia.

References

[Copeland and Maier 83] "Making Smalltalk a Database System", G. Copeland and D. Maier, Proceedings ACM SIGMOD Conference, June 1984.

[Atkinson et al. 89] "The Object-Oriented Database System Manifesto", M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier and S. Zdonik. Proceedings of the First DOOD Conference, Kyoto, December 1989.

[Cattell and Skeen 92] "Object Operation Benchmark", R. Cattell and J. Skeen, ACM Transaction on Database Systems, 17(1), March 1992.

[Carey et al 93] "The 007 Benchmark", M. J. Carey, D. J. DeWitt and J. F. Naughton, Proceedings ACM SIGMOD Conference, Washington D.C., 1993.

[ODMG 95] "The Object Database Standard: ODMG-93", R. G. G Cattell, ed., Morgan Kaufman Publishers, 1995.

Kurt Buehler

Vice President, Technology Development
Open GIS Consortium, Inc.

I feel compelled to try to contribute to this effort by writing generally about object orientation and, specifically, object oriented modeling and design, the way that these system development approaches can be utilized to provide interoperability and thus helping to improve the situation of

large government entities wrestling with a mission that is bigger than their budget, and last, but not least, stimulating thought. I cover these topics here, and leave the rest to the other panelists who are far more qualified to speak about them.

What is the question we are asking? And is it the right question?

I think we should start by examining whether we are asking the right question, but that is clearly not possible since we are here to answer that question that was already asked. In other words, the someone who asked the question thinks it is the right one. The question, as I understand it, is "What are the capabilities and benefits of object orientation in navigable databases for Intelligent Transportation Systems (ITS)?"

The first thing that I note is that there is no statement of a problem. I would also note that part of the solution seems to be a foregone conclusion. That is that object orientation (and maybe even OO database) is part of the solution. Given that no problem statement was provided, I propose to take a shot at a fuller question that I think has part of an answer in the smaller question that is at hand.

Given the current situation:

- the policy naming Caltrans as the maintainer of up-to-date information concerning the highways and byways of California on behalf of the citizens of California;
- the operational databases, hardware, and software within Caltrans;
- the prospects for better-cheaper-faster data collection within Caltrans;
- the need of the highly-mobile citizens of California for navigable databases for use in in-vehicle navigation systems, synthesis of navigation directions, and the direction of vehicles to a given street address; and
- the near and long term prospects for funding and personnel to provide these databases to the citizens of California,

What systems development approach will best enable Caltrans to deal with this massive amount of distributed data in terms of storage, processing, and maintenance?

Let us break down the assumptions a bit to try to understand what is being asked.

The first part says that Caltrans will maintain data about the California transportation system. It is in their mission and the data they generate, collect, and use throughout the life cycle of the transportation system can be used for many purposes including very many more outside the strict mission of Caltrans than for purpose within the mission. If Government as a whole is to be effective, the components must learn to interact the way businesses do in a competitive environment. Businesses aim for virtual corporations, partnerships, and consortia to leverage each others' strengths and/or resources. Thus if Caltrans is to be the maintainer of California transport system data, then they must figure out how to share it and make it more useful to others. They need interoperability.

Caltrans has a past. The legacy of years of road building, operation, and maintenance is a lot of data (some in database management systems, some not), a lot of hardware (used over the years as platforms for automation), and a lot of software (some off-the-shelf, some not, but all used to implement various automation programs and initiatives). This past (if it is anything like the agencies that I have dealt with) has been expensive and there is a large amount of inertia associated with this investment. Furthermore, they cannot afford to be non-operational, even for a little while. Thus Caltrans must figure out a way to leverage this legacy as it migrates to a better (presumably more object oriented) future.

They need a migration path that includes interoperability and platform independence.

With the advent of the commercial remote sensing market (initially Spot and EOSAT, soon to be Space Imaging, EarthWatch, and OrbView, among others) and the long established aerial photography industry as well as established (or soon to be established) programs of automated planning, design, engineering, operations, and maintenance within Caltrans, comes a vast prospect for data collection. There are many more sources of digital data then there were even five years ago. Thus Caltrans must figure out how to store, process, maintain, and, as I stated earlier, to share and add value to this data such that it is more useful, in more ways than it has been in the past or is currently. They need scaleability, extensibility, and portability.

California is well known to those of us outside of California as "The place with the REALLY big highways and even more traffic." It is common knowledge that ALL Californians have a Thomas Guide in their back seat. It seems obvious to me that Caltrans would be worried about finding ways to reduce the overall on-the-road time for every single Californian and, interestingly, so would all Californians. ITS seems to be an emerging technology area that can help in this regard. Not to mention the benefits it can have for emergency service vehicles and Caltrans itself in its pursuit for a higher quality of service to the citizen. Thus Caltrans must figure out how to feed these systems with up-to-date, accurate navigable databases in a timely and efficient manner. They need a robust, simple, efficient systems design.

Caltrans must accomplish all of this under budget and within current staffing constraints. They need a miracle, they need object technology and a well-designed and executed plan.

Know that we better understand the question, let's examine the role OO might play in providing an answer.

Can OO Help?

The answer is, "Of course it can help." How do I know? Look at a big company with the problem of delivering a continuous stream of desktop software products to an ever smarter and more sophisticated mass of Personal Computer users while at the same time dealing with the reality of their (the PC owners') existing data, hardware and software, limited budgets, desire to share data and applications with each other, etc. Microsoft used object orientation to help.

Look at Wall Street, who, when faced with the problem of building systems that supported trading activity in a world of complex data structures, simulation, and process control, utilized

object technology to ease the burden and to make the transition from the past smoother. Look at Boeing, who, faced with the difficulties of automating the entire production, operation, and maintenance of their aircraft, turned to object technology.

But, Put Design First!

Examples abound, and to be honest, counter examples abound. But in all cases, success and failure are determined (after factoring out poor budgeting and management) by the amount of time and energy spent on analysis and design.

So what am I saying? I am saying that much time and energy must be spent looking at the problem to determine what the objects are before trying to implement systems. Should there be billions of special purpose objects, one for every possibility in one centralized design? NO, there should be many simple, smaller, interesting designs that serve the needs of specialists while making connections to the collective whole. In other words, we need a few simple, but powerful, design patterns that will be reused over and over.

What are some of the other considerations?

- Thought should be given to transactions and concurrency. How many times will objects be accessed within a given time period and for how long? What is the granularity of access? Is it at the level of routes and route segments, on-ramps and bridges, or potholes and signage? How often will these databases be updated? Yearly, monthly, weekly, by the hour, minute or second? When and how often will the vehicle-based systems obtain information?
- Thought should be given to performance. Will the processing, update and maintenance occur? In the vehicle, in the local, region, or state offices, or in some combination?
- Thought should be given to distribution. Where will the data be stored? In the vehicle, in the local, region, or state offices, or in some combination? How will the vehicle-based systems obtain information?

Yea...But What Are The Objects?

I could tell you that the objects are the nodes and edges of a graph that represents a transportation system. I could tell you they are road segments, bridges, on-ramps, guardrails, signs, and culverts that make up the physical infrastructure. I could tell you that they are the landmarks, buildings, and topographic features that humans use to recognize past visited locations. I could tell you that they are districts and regions that make up the administrative units. I could tell you that they are planning, engineering, construction, maintenance, procurement, administration, clerical, research and development, and public relations departments that make up the organization. In all of these cases, I would be correct (or at least in the ball park). The point is that all of these different views need to be examined, each in the context of the place where it is needed to perform a mission.

For our navigable database, we could probably get away with some variant of the first three views: the physical infrastructure and navigation aids, and their association with some representation that enabled navigation. The variant would add such things as reference systems, accuracy, coordinates, and, probably, quite a bit of descriptive information. Object oriented models must, however, go beyond the semantic data model, using the data and behavior merging characteristics of objects to describe functions to navigate through the model as well as its structure. In fact, one of the foundational dogmas of object technology is that the structure of data is veiled by a procedural interface.

The benefits of specifying a navigational database in terms of an object-oriented model expressed as procedural interfaces are many. I describe some of them here for your consideration.

In the OO software environment, objects interact strictly on the basis of their interfaces, which neither programs nor programmers change. Encapsulation controls modification to the state of an object by dictating that changes to the object's data may be made only via the methods of the object. By providing a modular way to keep state and methods together, encapsulation also tends to facilitate more natural thinking in design and construction of systems.

Hidden behind an object's interface, the exact implementation of the object's methods and the format and layout of the encapsulated data is private to the object itself. This isolation allows users of the object (that is, programmers integrating the object into systems) to concentrate on the capability provided (as defined by the interface) and not on the details of the inner implementation. This concept of encapsulation allows systems to be defined using interfaces. In the case of navigable databases, this is highly desirable because navigable database developers and developers incorporating navigable database processing into systems are thus free to provide different implementations and to update the implementations as long as they meet the interface specification. These are the foundations of components, a powerful concept that make extensibility of functionality real.

The ability of one object to subclass or inherit some of the functionality of another object while overriding other functionality is called inheritance. Inheritance of the actual implementation of an object is very useful in the context of application building. However, it is only possible if the implementation language provides this capability. Inheritance of interfaces, the ability of one object to reuse the interface of another object (without reusing the implementation), is possible using most programming languages. Interface inheritance should be employed, providing developers the freedom to implement using object or non-object languages, while retaining the modeling benefits of inheritance.

Summary

Once a well designed, simple, yet effective model is designed and presented as a procedural interface, we can begin the task of implementing the model in software. Now the answers to the many questions about transactions, concurrency, performance, and distribution must be revisited in the context of implementation. I feel it best to leave this with the other panelists.

Max J. Egenhofer
National Center for Geographic Information and Analysis and
Department of Spatial Information Science and Engineering
Department of Computer Science
University of Maine
Orono, ME 04469-5711
max@spatial.maine.edu
http://www.spatial.maine.edu/~max/max.html

OBJECT-ORIENTED NEEDS FOR NAVIGABLE DATABASES

Three years ago, I challenged the database community to consider vehicle navigation systems as a demanding database application (Egenhofer, 1993) for which then-current database technology deemed insufficient. Among the requirements I raised were

- real-time access to spatial data,
- complex data models,
- spatial query languages, and
- multiple representation.

I think the underlying requirements have not changed since, nor have I seen the technological advances that would make the challenge obsolete.

In this position paper, I want elaborate on the issue of multiple representations. I have chosen this aspect because it is an area that I believe is often overlooked when intelligent vehicle navigation systems are investigated. Multiple representations are a critical issue for any data model applicable to intelligent vehicle navigation, and only object-oriented concepts such as generalization (with inheritance) and aggregation (with propagation) support them appropriately (Egenhofer and Frank, 1992) . These are not the only reasons for having database management systems with advanced semantic modeling capabilities, but they are among the most demanding ones.

Under *multiple representations* I understand the co-existence of geometric representations covering the same geographic extent at increasingly coarser detail (Buttenfield, 1989) . These versions cover the same area, but unlike temporal versions (which describe alternatives), they *co-exist*. Also, they are semantically distinct from history, which refers to different times. Multiple representations refer to different *levels of detail*. An example of a detailed representation is a road as an extended areal object with detail about lanes, shoulders, rest areas, on- and off-ramps, etc. At a coarser geometric level, these details are aggregated into a single object that is now represented as a line. At even less detail, certain aspects of the roads are dropped leading to a simplified representation of a road network (Timpf et al., 1992) . Simplification within a road network may further eliminate less important connections while preserving the main arteries.

Multiple levels of detail are necessary for a variety of tasks for two main reasons:

- Reduction of cognitive overload when graphically presenting spatial information (McMaster and Shea, 1992) ; and
- Performance of efficient large-scale analyses (Frank and Timpf, 1994) .

Usually one would take the approach of maintaining a most-detailed geometric representation (Beard, 1987) from which any coarser representation gets derived. In standard database terminology these derived representations would be considered *views*. The derivations often are semantically complex processes and commonly have to undergo extensive manual quality control. For this reason, the derivatives are stored—something the database community would call a *materialized view*.

The major bottleneck is the generation of these views. No established sets of reliable procedures exists for generating such spatial views; therefore often tedious manual quality control has to be applied in order to ensure that critical spatial information was preserved.

There is a need to assess:

- whether a materialized view is consistent with the most-detailed representation, and
- whether two or more increasingly coarser views are consistent among each other.

These assessment are not limited to checking individual objects and how they were simplified, but they have to include an examination of whether the spatial relations with respect to other objects were maintained.

Multi-resolution requires the following object-oriented concepts:

- Encapsulation—meaning that each object knows to which operations to respond—is extremely important for multi-resolution databases. If objects can change their representations, this may imply that some operations are impossible on the coarser representation.
- Overloading matters since different implementations may apply depending on the representation of the objects.
- The notion of identity plays a particular role in multi-representation spatial databases where a view may represent the same object (generalization) or an aggregate of different objects.

References

Beard, K. (1987) How To Survive A Single Detailed Database. in: N. Chrisman (Ed.), *AUTO-CARTO 8, Eighth International Symposium on Computer-Assisted Cartography*, Baltimore, MD.

Buttenfield, B. (1989) *Multiple Representations: Initiative 3 Specialist Meeting Report*. National Center for Geographic Information and Analysis, Santa Barbara, CA, Technical Report 89-3.

Egenhofer, M. (1993) What's Special about Spatial—Database Requirements for Vehicle Navigation in Geographic Space. *SIGMOD '93, SIGMOD Record* 22(2): 298-402.

Egenhofer, M., and A. Frank (1992) Object-Oriented Modeling for GIS. *Journal of the Urban and Regional Information Systems Association* 4(2): 3-19.

Frank, A., and S. Timpf (1994) Intelligent Zooming. *Computers and Graphics* 18(6).

McMaster, R., and S. Shea (1992) *Generalization in Digital Cartography*. American Association of Geographers, Washington, DC.

Timpf, S., G. Volta, D. Pollock, and M. Egenhofer (1992) A Conceptual Model of Wayfinding Using Multiple Levels of Abstraction. in: A. Frank, I. Campari, and U. Formentini (Eds.), *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Pisa, Italy. Lecture Notes in Computer Science* 639, pp. 348-367, Springer-Verlag, New York, NY.

David Fletcher, Craig Dean
Alliance for Transportation Research
Sandia National Laboratories

NAVIGATING IN A DIGITAL OBJECT HABITAT

Introduction

Throughout history every significant advance in information technology has been underestimated both in its application and its ultimate impact by pundit and developer alike. Although many products offered technical improvements, those that were revolutionary, such as the printing press, radio, television and the personal computer altered the ways that people perceived and interacted with the world. Books allowed readers access to distant authors' thoughts. Broadcasts added the experience of their personalities. The personal computer, linked in a network of other computers, made every reader a potential author, editor, publisher and performer. We believe that Object Oriented technology (OO) is another technical advance as poorly understood and as revolutionary as the preceding examples.

Object Oriented Thinking

Although OO was developed as a software engineering technique, its power extends beyond making better software. Our experiences with OO have highlighted four aspects that collectively suggest a new way of experiencing and understanding transportation and other large scale systems. These aspects of OO, when combined with OO abstraction and encapsulation techniques, point to a new way of thinking about objects and the next generation of information systems. We have discovered that objects have linguistic, convergent, weakly homomorphic, and symbiotic properties that can be exploited in complex systems analysis and architectural design.

The linguistic aspect of OO is derived from the semantic richness implied by the object paradigm itself. OO provides syntactical constructs that express concisely the complexity of human experience in human terms and provides the ability to transform these experiences into software analogs. The OO developer and the domain expert spend much time discovering, defining, and negotiating descriptive specifications of objects as they appear in the problem domain. OO methods provide the symbolic mechanisms to capture complex object knowledge and responsibility using this domain as its own best model.

These specifications translate the behavior and characteristics of physical objects into their digital counterparts. The specifications are maintained in semantic models (e.g., class hierarchies) that reflect both the problem domain (i.e., the physical world) and the solution domain (i.e., a digital world or habitat). The resultant OO system consists of physical objects immersed in the real world, software objects existing in one or more digital habitats and a single semantic object class model that reveals the salient behaviors of both (in human terms). The problem domain and the digital habitat converge and are congruent with the same model. The ability of the same semantic model to represent a physical object and its digital counterpart represents a profound capability that has yet to be fully exploited.

The third revolutionary aspect of OO thinking is to liberate objects in the digital habitat from the same natural laws as their physical counterparts. Digital objects may (and most likely should) have different behaviors than their physical counterparts. For example, physical objects are subject to gravitational effects; digital objects may be designed to float. Many physical objects are solid; their digital counterparts may appear ethereal.

Similarly, other physical effects such as time may appear differently to the digital object. Digital time can stand still, run forwards and backwards and elapse at rates faster and slower than world time. Although the video arcades provide abundant examples of alternative digital physics, no consensus on these behaviors has emerged.

Another way to think about object liberation is to give them responsibilities that their real world counterparts don’t or can’t have. For example, a physical book is an inert combination of pages, binding, covers, and so on with obvious properties (e.g., number of pages, title, Dewey Decimal Classification) and operations (e.g., open_cover, turn_page). A digital book can add additional responsibilities (e.g., determine_condition, schedule_binding) that unavailable to its physical counterpart.

Coupling digital objects to their physical counterparts is the fourth aspect of OO that offers even greater opportunities for illumination. MIT’s Media Lab labels this strategy “things that think” and has applied the concept to a number of consumer-related product ideas such as smart door knobs, smart clothes, and so on. In essence, the objects—physical and digital—exist in a symbiotic interaction. The digital object (Od) is aware of and can affect state changes in the physical object (Op). Objects may be tightly coupled (i.e., state changes in either Op or Od are immediately reflected in the other) or loosely coupled (i.e., state changes in either Op or Od are not immediately known to the other).

Liberating and coupling objects have significant consequences. In the book example, this strategy results in a different system architecture for a digital library. Without smart books, librarians must perform all intelligent operations. However, if books are allowed to assume some of these responsibilities, digital librarians no longer have to. Although a book object initially may need to depend on human agents to fulfill its responsibilities, it may ultimately substitute appropriate sensors and communication channels to its physical counterpart. Note that this substitution has implementation impacts and not an architectural ones.

We can apply the aforementioned ideas to geographic information science. Suppose that we create a set of digital geographic objects and integrate them into a digital habitat. The simplest case is a digital feature inventory anchored in digital two- or three-dimensional space with no attention to temporal or functional issues. This is a kind of object museum, useful (browse able) but not awesome and is exemplified in most existing GIS offerings.

Introduce the temporal aspect and we can now visit the museum with a time machine approach. Grab a temporal slider and inspect the habitat at a future or past (digital) time. The objects appear to age or rejuvenate. Better perhaps (we can predict the future and understand the past), but still not awesome.

But introduce a dynamism based on willful intent, behavior, and interaction (e.g., a bus driver traversing his route, a commuter facing morning rush hour, a family taking a vacation, an emergency vehicle responding to a call). We can watch the objects in the collective unfold, shrivel, or whatever and begin to understand cause and effect issues, watch flows, mitigate effects, plan to mitigate effects, anticipate impacts, choose to accept impacts, introduce change agents, etc. Now we can perhaps understand life cycle issues at the system, subsystem, component, and intra component levels. We can construct inference engines which infer what's possible, likely, etc.

Our museum is now more like a natural zoo (or THE Natural zoo) with an infinite number of ways to monitor the inhabitants. So we have migrated from:

- stuffed animals (static digital maps);
- to live caged animals (an inventory of digital models which cannot interact);
- to a natural habitat zoo (digital models are embedded in a realistic digital habitat and can interact with it);
- to the Savanna (digital models in a realistic digital habitat, highly interactive with one another and with their respective physical counterparts).

The "physics" we seek is demanded once we want to begin modeling INTERACTIONS, either between individual objects and their habitat (e.g., effects of natural forces on a bridge over time) or between digital objects (e.g., effects of individual or collective truck traffic on a bridge). It is the interaction of the separate models that begins to make things awesome. The ultimate digital habitats will have this "physics" built into (between?) them and will allow the expression of behavior, intent and interaction. The ability to model the emergent states of the objects interacting in the digital habitat is the promise of OO.

The Transportation Habitat

The transportation habitat (as described by the transportation object model developed in the GIS-T Pooled Fund Study) is a multidimensional domain (i.e., the domain extends into physical, spatial, temporal and functional dimensions) where transportation and other geographic objects live and interact. Consequently, all transportation related objects (either physical or digital) have a functionally defined role that determines certain behavioral characteristics that affect other objects proximate to them during some part of their mutual existence.

The fundamental object class described by the Study is the Transport System. Transport Systems provide basic transport access, connectivity, and capacity services through an interconnected network of transport links and nodes. Systems are distinguished by their ability to provide specific transport services in response to explicit policy objectives. Individual transport system instances may be based on major modal distinctions (e.g., highway systems, rail systems), on minor modal distinctions (e.g., truck route systems, public transportation systems) or on functional distinctions (e.g., farm to market systems, just-in-time manufacturing systems, recreational travel systems). Functionally integrated transportation systems may include a number of different modal components.

Transportation links are distinguished by mode (e.g., highways, railways, pedestrianways) and are existing, historical or proposed travelways between customer sites (i.e., travel origins and destinations). Specific travel choices are made at junctions (i.e., allowing change of routes) or terminals (i.e., allowing change of mode).

Origins and destinations can be spatially (e.g., major employer sites, households, vehicle locations), temporally (e.g., events), and/or functionally distinct (e.g., this bus stop is also a bus transfer point) sites. Each destination is in a service state. That is, destinations are willful, desired physical, spatial, temporal, and functional states of being. Similarly, origins can be defined as demand states.

Navigating through the Transportation Habitat

Since the Transport System is defined as the set of all access, connectivity, and capacity services, navigation issues revolve around finding those services (or the best path to get to them). However, in this transportation habitat, navigating becomes a far more complex (i.e., realistic) activity than merely determining vehicle locations and determining map directions. Imagine the habitat as the union of all possible service and demand states (i.e., all possible origins and destinations). Motion in this habitat is defined as the act of changing state. Navigation is defined as the process of planning, recording, or controlling the motion between an origin state and a destination state.

Navigating is not limited to traveling from one spatial state (i.e., position or place) to another, but encompasses any purposeful motion through the habitat. This allows for the polymorphic navigation operation to apply to temporal and functional space, as well as to geographic dimensions. Measures of proximity can also be applied in these other spaces.

The responsibility for navigation can be assigned to a Navigator object. The Navigator determines one or more paths between demand and service states. As mentioned earlier, the navigator may be coupled with other objects in the habitat, such as vehicles or travelers which may be tightly coupled to their respective physical counterparts.

A series of example scenarios may help visualize n-dimensional navigation:

1. Given a known origin object (defined as a function of class, location, role and time) and a known destination (also expressed as a n-dimensional function), determine a traversal (e.g., shortest, least cost) between them in a specified transport system. The dimensionality of the traversal is determined by the dimensionality of the origin and destination objects.

For example, ignoring role and temporal state changes in the origin and the destination objects produces the conventional navigation problem. That is, the origin and destination are in different geographic locations; the traversal connecting them needs only to navigate through space. However, if we select a specific object as the origin and the same object at some other time as the destination, we have defined object oriented time travel. In a sense, we are navigating through the set of state altering events (e.g., actual or simulated experiences) between the origin and destination states of the same object.

Allowing two or more dimensions to vary creates more complex navigation problems. For example, the intercept problem requires a traversal between two points whose locations are a function of time.

2. Given a known n-dimensional origin in a transport system, find a destination object satisfying an n-dimensional search criterion.

3. Given known origin and destination objects plus a traversal between them expressed in one traversal reference method, describe the same traversal in another specified traversal reference method (or system). For example, provide a description of a bus route A using street names and address ranges.

Issues Surrounding the Transportation Habitat

The following issues present significant challenges for the transportation habitat object developer. This list is intended to be illustrative, not exhaustive.

1. Since objects are semantic abstractions of the essence of things, a common domain language is essential. However, intelligent navigation requires cooperation among representatives from several domains that have not historically had much interaction.

2. Some of the objects in the habitat need to persist for many decades (e.g., highways); others need only exist for a few seconds (e.g., real time vehicle positions).

3. The number of objects in the habitat is extremely large (e.g., number of vehicles, roadway sections). In addition, the physical objects are owned and operated by a very large number of agencies, organizations and individuals.

4. Most of the objects have very complex responsibilities, some of which may not yet be known. Emerging technologies such as in-vehicle GPS navigational devices present a moving target habitat.

5. The very large number of objects existing at any point in world time and in simulation time (i.e., route planning time) implies an enormous number of object interactions.

6. Because the state of any object is interdependent on the states of objects proximate to it, both object and system level behavior is nonlinear.

7. Transport system components exist in a multidimensional universe. Navigating means more than moving in three-space. What does proximity mean in this context?

8 As of early 1996, there is no agreed upon architecture to guide developers. Most work to date has been proprietary and of limited application.

9. Although it is likely that digital objects will be more tightly coupled with their physical counterparts (e.g., smart cars), there is no consensus on which objects get smarter.

Conclusions

Much of the opportunity in this area results from attempting to cultivate, then manage this digital habitat. Creating static collections of inert things is relatively trivial; evolving the diversity and functionality necessary for a robust, useful digital universe is well beyond the state of the practice.

We are seeking some sort of a digital framework into which we can place the digital objects. This framework forms the habitat and allows and supports the interactions between the inhabitants. It is likely that these interactions will be unknown in advance. That is, proximate objects affect each other—without caring or advance warning. Defining the principles (i.e., protocols) of this ecosystem presents many challenges, most as of yet unanswered. For example, digital objects must reveal themselves coherently in time, space, and functional role. This requires them to manage not only their internal states but also respond to the influence messages sent from nearby objects.

Although our existing development tools seem quite primitive artifacts with which to fashion complex systems, those that are Object Oriented are the best that are available. We have found four tools that enhance the basic OO tools of class hierarchies, encapsulation and polymorphism.

Including semantic abstraction, convergence, weak homomorphism and symbiotic coupling has allowed us to more confidently navigate through these problems.

References:

Phase B Preliminary System Design Report, GIS-T Pooled Fund Study, Albuquerque, NM.

Taylor, David A., Business Reengineering with Object Technology, John Wiley & Sons, New York, 1995.

Definitions

automatic vehicle location (AVL) - A mechanism to determine the position of vehicles and report the positions back to a central control.

destination - 1. The point at which a trip terminates.

dynamic routing - In demand-responsive transportation systems, the process of constantly modifying vehicle routes to accommodate service requests received after the vehicle began operations, as distinguished from predetermined routes assigned to a vehicle.

highway, street or road - General terms denoting a public way for purposes of vehicular travel, including the entire area within the rights-of-way. The recommended usages are as follows: in urban areas, highway or street; in rural areas, street or road.

link - In planning, a section of a transportation system network defined by intersection points (nodes) at each end; that is, a link connects two nodes.

link travel time - the time required to traverse a link, including waiting time.

navigation - The capability to determine ones position (IVHS America).

network - 1. In planning, a system of links and nodes that describes a transportation system. 2. In highway Engineering, the configuration of highways that constitutes the total system. 3. In transit operations, a system of transit lines or routes, usually designated for coordinated operation.

origin - The point at which a trip begins.

path - In planning, any series of links where each succeeding link has the ending node of a previous link as its beginning node.

roadway - The portion of a highway built, designed, or ordinarily used for vehicular travel, except the berm or shoulder.

route - 1. The geographical path followed by a vehicle or traveler from start to finish of a given trip. 2. A designated, specified path to which a transit unit is assigned. Several routes may traverse a single portion of roadway.

route guidance - the provision of assistance to a driver in getting from his current location to his destination in a detailed step-by-step manner (IVHS America).

route planning - The ability to select an optimal route from ones location to a final destination (IVHS America).

transportation system - A system that provides for the movement of people, goods or both.

trip - 1. A one-way movement of a person or vehicle between two points for a specific purpose.

~~**vehicle** - Any device or contrivance for carrying or conveying people or objects, including land~~ conveyances, vessels, aircraft and spacecraft.

Andrew U. Frank
Geoinformation
Technical University Vienna
frank@geoinfo.tuwien.ac.at

1. Object-Oriented Modeling

Object-oriented modeling is a strategy developed in software engineering to help with software projects, but it is applicable generally as a tool to describe complex situations. Traditional methods of programming attempt to describe processes and procedures as successions of steops to execute. Such descriptions were regularly found to be incomplete or erroneous. It appears that the human cognitive system is not well prepared to organize and describe complex processes. As a simple example: can you describe—in writing—how to tie a shoelace, such that somebody else can follow the instruction to learn how to tie his shoes? On the other hand, humans can comprehend and describe complex static situations: a car consists of many parts, a town consists of buildings and roads and cars circulate in it, etc. The functional integration between these parts is easily understood. Even our language seems to provide more nouns and constructs to explain relations between things than to qualify the description of actions.

An object-oriented method for modeling describes objects and the operations applicable to them. Objects can be physical objects, like cars, roads, etc., but they can also be events, like road accidents, or purely abstract, like vehicle count per building. The operations applicable are, for example, car driving, road construction or maintenance, accident occurrence, or aggregated counts.

An object can be anything which we perceive as having an existence, real or imagined. Everything we describe with a noun can be an object. Operations can be divided into those which create or change the object and those which allow us to observe some current property of an object, e.g., the current position of a car, the length of a road, etc. The level of detail for a model and the focus of the modeling effort depend on the task to be carried out.

Descriptions of objects are easier to give than descriptions of complex sequences of processes. It is assumed that an object-oriented description helps the analyst and the programmer to produce software which responds better to the user's needs. The earliest application of object-oriented methods was in simulation, and today's object-oriented software engineering methods are necessary to construct the graphical user interfaces which have become customary. But the object-oriented technique is not limited to this and has much wider applicability. It is a general approach to the description and modeling of objects.

2. Formalization of Object-Oriented Description

Capturing a complex situation and describing it, such that others can reconstruct it, is difficult. It requires a defined terminology independent of the individual, i.e., it is a formal method. Algebra, in particular universal algebra, provides the mathematical foundation for object-oriented programming.

An algebra is a triple of

- a set of individual objects
- operations applicable to those individual objects, and
- rules which describe the result of the operations.

The object-oriented method has been invented in several branches of computer science and different terminology has resulted. We have found it necessary to separate the following issues and use the following terms:

Class for an abstract description of a set of objects, with the operations applicable and their effects (often called abstract data types, but this implies also a data structure, which may appear hidden);

Data or data structure for the computerized description of a single individual (sometimes called a data type), and

Instance for the explanation of how operations are carried out for a specific data structure (could be called implementation).

Several fields have grouped these concepts differently; programming languages often use the word *class* for both the abstract description and the (sometimes hidden) implementation. The class here compares to the template in C++, whereas the C++ class includes data and instance. In the terms of the OMT terminology, class here describes the object with its operation, whereas the instance gives the OMT methods. Relational database modeling is mostly concerned with data and data structure.

All objects in a class have the behavior described by the operations and the rules; different sets of objects with particular representation can have the behavior of the same class. Chevrolets, Fords, and Vws all have the behavior of cars. A GM light truch has the behavior of car (driving etc.) and truck (load capacity etc.). This is often described as "multiple inheritance": an object "inherits" the behavior of multiple parent classes. In the terminology used here, this is described by giving instance definitions which link the data objects with the classes; for one class multiple data structures (implementations) may be provided and for one data structure, operations from different classes may be defined.

In an object-oriented design, relations between objects follow certain patterns:

classification: objects of the same class have the same behavior;

generalization: the general class describes operations which all of the objects in the more specific classes have in common (trucks, racing cars, and sedans are all cars);

aggregation: an object consists of parts (a car consists of wheel, engine, etc.).

Objects can play **roles** (e.g., a person is the owner of a car; person and car being objects, ownership being a role).

3. Model for a Road Network

An object-oriented model of a graph can be used as a base for modeling a road network. The objects are then nodes (for road junctions, etc.) and edges (for road segments). The translation of standard algorithms, e.g., Dijkstra's method to find a shortest path, into this environment is not difficult (two hours for straight translation of shortest path).

The road network can be modeled at different levels depending on the task to be carried out. Timpf et al. (1992) described three levels of detail for a road network model, oriented towards:

trip planning
driving instructions, and
actual driving.

At each level, different objects and operations are in focus. Objects of the more general levels are disaggregated into objects of the lower level. Equally important is a subdivision of a road network in a hierarchy of interstate, highway, city road, etc. Car and Frank (1995) show how to determine the fastest path in a structured network (without inspecting trips along roads of lower hierarchical levels).

4. Road Movement

The continuous movement of vehicles on a road network is modeled based on current speed and previous position ($s_{i+1} = vt + s_i$). Such a model understands the links between the nodes as a continuous line. Kuipers, in the TOUR model, described a discrete model of navigation in a road network considering only the nodes, the decisions to be made at a node, and the node reached after each decision:

```
type Pos = Int    -- meters
type Time = Int    -- seconds
type Speed = Int       -- meters per second

class Vehicles v where
  make :: Pos → Time → v
  at :: Time → v → Pos
  speedup :: Int → Time → v → v
  slowdown :: Int → Time → v → v
  slowdown a t v = speedup (-a) t v

Data Vehicle = V Pos Time Speed

instance Vehicles Vehicle where
  make p t = V p t 0
  at tt (V p t v) = v * (tt - t)
  speedup a tt (V p t v) = V (at tt (V p t v)) tt (v+a)

v1 :: Vehicle
v1 = make 0 0
```

To calculate where a vehicle is after speeding up and slowing down:

```
at 4 (slowdown 2 3 (speedup 3 1 v1))
```

The challenge for proper temporal modeling of individual agents moving in a road network is to combine these two models. In an object-oriented style, individual algebras can be merged with minimal inference.

5. Tools

The object-oriented concept is used in several programming languages (Smalltalk, Simula, C++) to varying degress and with notable conceptual differences—typically stressing the sharing of code and not the commonality of behavior of objects. Object-oriented databases are appearing on the market. Object-oriented methods are also applied in software engineering tools (CASE tools).

We have used a functional programming language (Gofer or Haskell) which structures an object-oriented model in a formally very clean way. It is close to an ideal specification language with a comprehensive set of tools for formal checking of the specification, but it has the additional advantage that the code can be executed (at least for prototyping) and the behavior observed. This helps to detect errors early.

6. Conclusions

Object-oriented modeling is a general approach to describe informally or formally some part of reality which is of interest. It concentrates on objects and operations applicable to them. The task one tries to solve indicates which objects and which operations are in focus. Different models of the same situation are often necessary and they are later combined.

The object-oriented approach is very general. It is based on the human cognitive ability to comprehend complex sttic structure of objects and how they can be used and avoids the difficulty of organizing complex tasks in subtasks, etc., which is notoriously difficult for humans.

Anything which is thought of as an object with corresponding operations can be an object. The physical objects in the world are good examples of objects: road segments, junctions, towns, etc. Geographic features are other examples of objects. For a simulation of road traffic, the vehicles are objects, but also statistical counts, events like accidents, etc., can all be modeled as objects with corresponding operations.

References

Car, A., and A.U. Frank (1995) Formalization of conceptual models for GIS using Gofer. *Computers, Environment, and Urban Systems* **19**(2): 89-98.

Frank, A.U., and W. Kuhn (1995) Specifying open GIS with functional languages. *Fourth International Symposium on Large Spatial Databases (SSD 95), Portland, ME.* Springer-Verlag, Heidelberg.

Timpf, S., G.S. Volta, et al. (1992) A conceptual model of wayfinding using multiple levels of abstractions. In A.U. Frank, I. Campari, and U. Formentini, editors,*Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*. Springer-Verlag, Heidelberg and Berlin **639**: 348-367.

Michael J. Franklin
University of Maryland
College Park
franklin@cs.umd.edu
http://www.cs.umn.edu/users/franklin

The Object Relational Manager

Many classes of applications are not adequately supported by traditional relational or object-oriented database systems. Relational database systems excel at providing high-level associative (i.e., query-based) access to large sets of flat records. In contrast, object-oriented database systems provide more powerful data modeling capabilities and have been designed to support efficient integration with host programming languages and navigation-based access to data. Because of these different, and in some ways complementary strengths, it has become apparent that database systems combining the best aspects of the relational and object-oriented approaches are likely to gain acceptance across a large range of applications.

Database system builders have been approaching this perceived need in several ways. Relational vendors are moving towards integrating object-oriented features into their systems (e.g., the emerging SQL3 standard) while vendors of object-oriented systems are adding more powerful query facilities (e.g., OQL). Furthermore, a new class of hybrid "Object-Relational" systems has recently started to appear. These efforts have resulted in significant progress towards integrating object and relational concepts, but this progress has been primarily at the language and data model levels. In contrast, relatively little attention has been paid to the development of underlying *system architectures* that can efficiently support these hybrid models—particularly in a distributed environment.

Architectural Implications

While most modern database systems are designed to execute in a client-server environment, relational and object-oriented systems tend to exploit the resources of such environments in significantly different ways. Relational systems and their descendants are typically based on *query shipping*, in which the majority of query processing is performed at servers. The benefits of query shipping include: the reduction of communication costs for high selectivity queries; the ability to exploit server resources when they are plentiful; and the ability to tolerate resource-poor client machines (e.g., mobile clients, set-top boxes).

Object-oriented database systems, on the other hand, are typically based on *data shipping,* in which required data is faulted in to the client to be processed (and possibly cached) there. In a data-shipping system, the servers are relegated to the role of "intelligent" storage managers and transaction managers. Data-shipping has the advantages of exploiting the resources (CPU, memory, and disk) of powerful client machines and reducing communication in the presence of locality or large query results. Both of these factors allow data-shipping to scale as users (and hence, their associated client machine resources) are added to the system. Data-shipping also allows for light-weight interaction between host language-based applications and the database system, as is needed to support navigational data access.

Caching for Query Processing

Query shipping (as used by RDBMSs) and data shipping (as used by OODBMSs) are static policies, and as such, they each fail to fully exploit the distributed resources of a client-server system in many situations. A more flexible approach is to extend the techniques for client caching that have been developed and used in data shipping OODBMSs to allow for efficient query-based access as well.

The DIMSUM (Distributed Information Management Systems at the University of Maryland) project is developing such a flexible query processing architecture. This caching-based query processing approach can exploit the resources of distributed environments for reducing communication, for load balancing, and for parallelism in a way that can adapt to the requirements of a wide range of applications and system properties. For example, query operators can be placed at server machines when server resources are plentiful, but can be placed at clients to exploit cached data, thus reducing communications when bandwidth is limited, or offloading shared servers [Fran96]. Furthermore, the operator placement policy can take recent history into account, in order to determine where the caching of data is likely to pay dividends in the long run [Koss96].

Exploiting the Semantics of Cached Data

The integration of queries and caching raises additional opportunities for performance enhancement, beyond the merging of data shipping and query shipping approaches. One promising approach is *semantic caching* [Dar96]. Using semantic caching, the client maintains a semantic description of the data in its cache, instead of maintaining a list of physical pages or tuple identifiers. This allows for a compact specification, as a *remainder query*, of the items in the server database, not available in the client cache, needed to answer a query. Remainder queries provide reduced communication requirements and additional parallelism compared to faulting-based approaches. Semantic caching is particularly effective in situations where client cache space is limited, such as in a mobile environment, because it avoids the high overheads associated with caching on a per-object basis, while avoiding the sensitivity to object clustering that arises with page-based caching. Finally, semantic caching allows the use of value functions that incorporate *semantic* notions of locality. Such value functions can provide dramatic performance improvements over traditional approaches for applications such as mobile query environments.

Data Dissemination

Finally, in many environments, such as ITS, *data dissemination* may be a more appropriate means of data delivery than the request/response approach of current client-server systems. Using data dissemination, the transfer of data is initiated by servers, resulting in a reversal of the traditional relationship between clients and servers. We have developed a model called *Broadcast Disks* for structuring the repetitive transmission of data in asymmetric communication capacity between clients and servers. The Broadcast Disks approach integrates two main components: 1) a novel "multi-disk" structuring model that allows bandwidth to be allocated to data items in proportion to their importance; and 2) client-side caching and prefetching algorithms that are tailored to the multi-disk broadcast.

The Broadcast Disks approach can improve response time for non-uniform access to data. The inversion of the traditional relationship between clients and servers, however, has significant implications for the management of client storage resources. Examples of these implications include the need for cost-based cache replacement strategies and the unique opportunity for data prefetching that arises in this approach. The integration of data dissemination with more traditional query processing is a major focus of our current work in this area.

References

[**Acha95a**] Acharya, S., Alonso, R., Franklin, M., Zdonik, S., "Broadcast Disks: Data Management for Asymmetric Communications Environments", *ACM SIGMOD International Conference on Management of Data, San Jose, May 1995*.

[**Acha95b**] Acharya, S., Franklin, M., Zdonik, S., "Dissemination-based Data Delivery Using Broadcast Disks", *IEEE Personal Communications*, Vol. 2, No. 6, December, 1995.

[**Dar96**] Dar, S., Franklin, M., Jonsson, B, Srivastava, D., Tan, M., "Semantic Query Caching and Replacement", *Submitted for Publication*, February, 1996.

[**Fran96**] Franklin, M., Jonsson, B., and Kossmann, D., "Performance Tradeoffs for Client-Server Query Execution", *ACM SIGMOD International Conference on the Management of Data, Montreal, June 1996.*

[**Koss96**] Kossmann, D., and Franklin, M., "Cache Investment Strategies", *Submitted for Publication*, February 1996.

John R. Herring

Oracle Corporation
196 Van Buren Street
Herndon, Virginia 22070

OBJECT ORIENTATION IN INTELLIGENT TRANSPORTATION SYSTEMS

The question is what are "the capabilities and benefits of object orientation in navigable databases for Intelligent Transportation Systems (ITS)?" It is hard to explain how and why without writing a treatise on how to use object orientation. It is also a particularly hard time to do it. The geographic data and processing world is in a state of flux. There are and will continue to be changes in the way we put systems and data (objects) together. While we cannot be certain of what the outcome will be, we can make some educated guesses based on the technical capabilities, tendencies and limitations of the systems that will make up the new geodata world.

I am not going to give yet another primer on objects. By this time we all should have an idea what encapsulation, polymorphism, and inheritance mean. What I ask you to remember is that the deeper we get into the world of objects, the more we discover that the basics are not what we thought. Be ready to jettison any ideas on objects as they become obsolete.

To answer this conundrum, we have to realize some simple truths about the realities of objects. Some of the following were learned only after long and painful experimentation, failure, thought, and success. These hard-won realities are:

1. Objects make good programming easier. Within the context of an application, or of a related suite of applications, the use of well-designed object class libraries can lead to better code, fewer bugs, and a better system overall. Unfortunately, objects do not make bad programming any more difficult.
2. The use of strict encapsulation and interface polymorphism (behavioral modeling, not structural) is a reasonable basis for defining interoperability between running processes without requiring common application object libraries.
3. Object programming languages that are too pure tend to be performance hogs. Commercially viable programming needs to balance theory and pragmatism.
4. Object structures, interfaces, and implementations are designed with specific applications, system requirements, and implementations in mind. If these change, then the optimal object design can change even though the information content of those objects remains essentially the same.
5. Complex systems consist of many applications, with different data representation, process performance, and functionality requirements. Thus, complex systems by their very nature require the same data in different objects structures because of 4 above. The more complex the system, the more this effect dominates.
6. Persistence and process do not necessarily have the same requirements. The more complex the situation the more likely they are to diverge.
7. Databases systems that are used as repositories for common data in support of multiple applications within a complex system must be able to serve that common data in a variety of

object formats (object views) to support the different object class libraries that are implied by 6.

8. Object oriented database systems (OODB) that are based on adding persistence to an object oriented programming language (OOPL) are designed to extend applications environments. As such they work well with a well-defined set of class libraries and related applications that utilize these libraries. They are not designed to do object views and, therefore, perform poorly in complex, multiple application systems.
9. Object access comes in two varieties which are access by identify (pointer navigation from known objects) and access by property values (query). Further, query access must be able to support navigation to implicit properties.
10. The keys to performance for value-based query are indexes.
11. Most object query languages emphasize navigation and ignore property based query. In such systems property-based query becomes navigation of collection objects.

Next we have to see how these truths apply to geographic data in general and ITS in specific. The observations are:

1. Geodata systems are complex. ITS systems just make it more so.
2. Geodata objects have location (geographical and temporally referenced geometry) as a property. The is just about the only thing that ties all geodata together.
3. The geodata world operates on value added.
4. The biggest databases that will ever exist will contain spatial data. The only possible exception is the human genome project.
5. In geodata, accuracy and relative accuracy are both important. In geodata processing, relative accuracy is more important than accuracy because the real world processes that they model are local in exception, even if they are global in extent.
6. ITS systems have to live within a larger geodata environment. Even if we say that they use outside sources of data and processing minimally, we know that the value added guys are going to become consumers of the data.
7. Databases for geographic objects have to optimize both spatial selection and spatial joins. Spatial joins are multiple class queries that reference a spatial relationship between the individuals for the classes.
8. Join is orders of magnitude harder than selection. Spatial indexes that optimize joins will probably be multiple entry indexes, based on regular decompositions of space. No one has gotten it right yet—meaning that there are reasonable data sizes for which each of the known implementations gives unreasonably bad performance.

Now we can answer the question:

1. Object programming languages, especially C++ for its balance of performance and functionality, make good application development environments for all types of geodata
2. OODBs based on OOPL make good persistent stores for closely related applications that use a relatively small, consistent and well-defined set of class libraries for the application objects. This is especially true if “pointer navigation” plays a large role in the applications processing.
3. RDBMS with object extensions (especially if they are in line with the SQL3 proto-standard) make good data stores for complex, multiple application environments. This is especially

true if large data volumes, value added applications, data security, concurrency, long-term persistence, distributed databases, or non-geodata applications play a role in the system.

4. Some combination of OODB for 'extended, persistent virtual memory' cache management on long running, complex applications with ORDBMS acting as long-term backing store probably makes the best scenario for complex systems. Neither database paradigm will eliminate the other.
5. Distributed computing environments, such as those based on OMG's CORBA (common object request broker architecture) will become the primary execution environment for most applications, especially for complex point products that derive their functionality from a variety of object components.

Shashi Shekhar
Computer Science Department
University of Minnesota
Minneapolis, MN 55455.
shekhar@cs.umn.edu
http://www.cs.umn.edu/research/shashi-group

GIS-T NAVIGABLE DATABASES : A CONCEPTUAL MODEL AND CHALLENGES IN USING OBJECT MODELS

Background

We have used Object data models for various aspects of Geographic Information Systems in Transportation (GIS-T), including routing/navigation with road-maps[1], driving simulation[2], tracking public transportation (Travlink project) and communicating incidents [3] with the Minnesota Guidestar (MNDOT), and Federal Highway Authority Inst. for Intelligent Transportation Systems at University of Minnesota. We are currently designing a database archive the traffic measurement from the 3000 sensors and cameras on the Twincities highway network.

This extended abstract defines our view of the data modeling needs of GIS-T (including navigable databases), the object data model, and the challenges in using object data models for GIS-T.

Data Modeling Needs of GIS-T

GIS-T is used to collect, analyze and output information describing the physical and logical properties of the surface of the earth occupied by the transportation networks. In a mathematical sense, the GIS-T manages a set of values for attributes of the points in multi-dimensional continuous space representing the time and transportation network geography. GIS-T are also known as navigable databases since they facilitate travelers, traffic managers, and commercial/public vehicles to navigate effectively.

Features represent individual characteristics of space. Each feature represents a mapping from space to a domain of values. Examples include traffic, elevation, soil type, and water level. Some features are *Proper Features*. A proper feature represents a collection of uniquely named geographic places such as roads, rivers, cities, and countries. Each instance of these entities has a unique name, and the entity's geographic location can sometimes change. For example, rivers like the Mississippi flood and change their course. Features which are not proper features are *Common Features*. These are often identified via a location in multi-dimensional space. Examples include accident, congestion, land parcel, etc.

Features as such cannot be stored in a GIS database due to the infinite size of the continuous space occupied by the transportation network. Features are discretized to create *coverages*, which are then stored in GIS. The discretization relationship between features and coverages often has the attributes of an *interpolation model* and *continuity constraints.* A coverage contains spatial (and temporal) extents with corresponding values of the feature. The geometry and topology of the spatial extents is often characterized to facilitate data manipulation and analysis. Geometry describes the shape (e.g., polygon, line, point) and dimensionality (e.g., 0, 1, 2, 3) of a spatial extent. Topology is a property of a spatial object that is unaltered by elastic deformations. An example is the connection between two road intersections in a road network.

For ATIS and other applications, an efficient and effective spatial graph database is needed to support spatial graph analysis which includes path computation, route evaluation, location/allocation, dynamic segmentation, and spatial graph overlay.

One important phase in GIS processing is providing a visual representation (e.g., maps) of the stored data. An effective visual representation does not convey any information which is not present in the GIS data, and the visual representation should convey all the information requested by the user at a detailed or an abstract level. *Visualization constraints* try to ensure this property in the presence of map generalizations and distortion to highlight major entities.

Object Model and Object Data Model

Object oriented databases (OODBs) were advanced in the last decade to address the "impedance-mismatch" between the relational model and non-standard applications such as geographic information systems (GIS). Each real-world entity is represented by an object with a *state* (e.g., attributes) and a *behavior* (e.g., methods). Each object is identified by a single OID (Object Identifier), which is independent of the values of object attributes, object location, and object structure. Objects exhibit encapsulation and information hiding, i.e., the private attributes of an instance are not visible from outside the object boundary. A set of objects which have the identical internal structure, i.e., the same attributes and the same methods, are represented by a class. Inheritance allows a class called a subclass to be defined based on the definition of another class as a superclass. The subclass inherits all the attributes and methods of the superclass. The subclass can define additional attributes and methods or redefine inherited ones. In many situations, it is convenient to overload a method name, i.e., to use a common name for different operations defined inside a class or a set of classes. An example of overloading occurs when a subclass redefines the implementation of an inherited operation. OODBs also support late binding, i.e., they identify the implementation to be used for an invocation of the operation at run time.

Object data models are different from the object models used in programming languages (e.g. Smalltalk) and in analysis and design of software systems. We list some of the major differences here.

OODB objects are persistent, i.e., they continue to live after the program terminates. Classes may be persistent or non-persistent as well. Some OODBs support meta-classes and allow objects to change their classes. Primitive objects need not have unique identifiers.

For example, many OODBs identify primitive entities, such as integers and characters, by their values. OODBs support constructors to create complex values and types from primitive ones. The minimal set of constructors provided by most systems includes set, lists, and tuples. Strict encapsulation is not considered desirable. Query languages allow predicates on the value of attributes, and most OODBs allow direct access to attributes to avoid, among other things, implementation of many methods for simply reading and writing attributes. Finally, the issues of interaction between inheritance and encapsulation have not fully resolved within OODBs.

Finally, several semantic extensions to the object models have been proposed for OODBs. These include associations, composite object semantics, and integrity constraints. An association is a link between entities in an application. Associations are similar to the notion of relationships in the Entity-Relationship model used by relational databases, and are characterized by degree and by cardinality constraints. Most OODBs do not explicitly represent associations.

Challenges in Using the Object Data Model in GIS-T

We proposed use of an object data model to represent several aspects of the GIS-T domain in 1991. Since then the object data model has been used for many applications within GIS-T.

We reflect on our experience in this paper to list the challenges in using object data models for GIS-T:

1. Scalability to GIS-T sized networks (million nodes/edges) :

Carrying out shortest path computation over a map stored in a database takes hours and hours [7]! While algorithms for spatial graph analysis are well understood in small networks in a main memory environment, there is a great need for research in developing and evaluating algorithms and database structures for processing of large road-maps with millions of road-intersections and road-segments.

2. Querying and data model for networks:

One of the strengths of data models and databases is to allow sophisticated querying. If the road maps used a proper data model of the network, it will reduce many obstacles in their use. Consider the problem of detecting road map quality problems such as ensuring the names of all segments of a road are consistent. This is a central yet conceptually simple problem in cleaning up TIGER files for creating road maps for in-vehicle navigation systems. Yet, the rudimentary structure of TIGER files makes it very hard to do. Navigation quality maps are available for only a handful of states in USA. Present OODBs provide a limited set of constructors and operators for modeling and analyzing graphs and networks, which are very critical for GIS-T. There is also a great need for further research and for developing consensus data models and query languages for networks.

3. System support for discretization and interpolation:

Features are often discretized in different ways in different coverages. Analysis of datasets about such features often requires tedious work to overlay one dataset over the other. System support of discretization (e.g., dynamic segmentation) will go a long way to facilitate such analysis.

4. Representing relationships during analysis:

An object model does not directly support user-defined relationships between classes, even though methods can be used to implement user-defined relationships between classes. In essence the E-R model may be more useful for conceptual data modeling, while the object model may be better suited for logical/detailed data models. We invite the reader to construct an object data model of a simple GIS-T partial to make this point concrete.

5. Common features and OIDs:

An object model requires that objects have an OID which is independent of the values of object attributes, object location, and object structure. It is not convenient to create such OIDs for many common features including land parcels, traffic measurements, accidents, congestions etc., which are usually identified by their location in space and time.

6. Features are continuous sets:

Objects inherently represent discrete countable sets of things, such as coverages. Physicists and engineers use very different data models (numerical analysis, differential equations, Markov fields ...) to model continuous phenomenon. Are object data models really the right way to model features ?

References

Bertino, E., and L. Martino (1993) *Object Oriented Database Systems: Concepts and Architectures*. Addison-Wesley.

Shekhar, S., A. Yang, B. Hamidzadeh (1991) Path planning and evaluation in IVHS databases. *Proc. Int'l Conf. on Vehicle Navigation Information System, IEEE.*

Shekhar, S., S. Ravada, G. Turner, D. Chubb, and V. Kumar (1995) Load balancing in high performance GIS: partitioning polygonal maps. *Proc. Intl Symp. on Large Spatial Databases, Springer Verlag (Lecture Notes in Computer Science).*

Shekhar, S., and D.R. Liu (1994) Genesis and advanced traveler information systems (ATIS): killer applications for mobile computing? *Proc. Workshop on Mobile and Wireless Information Systems National Science Foundation* .

Shekhar, S., A. Yang et al. (1991) A geographic database for IVHS management. *ASCE Proc. Int'l Conf. on Appl. of Adv. Tech. in Transportation Engineering*.

Shekhar, S., T.A. Yang, and P. Hancock (1993) An intelligent vehicle highway information management system. *Intl. Jr. on Microcomputers in Civil Engineering* **8**(3): 175-198.

Shekhar, S., A. Kohli, M. Coyle, and T. Single (1993) Pair path computation algorithms for advanced traveler information systems. In *Proc. Intl. Conf. on Data Engineering IEEE.*

Shekhar, S., and D.R. Liu (1995) Connectivity-clustered access methods for networks and network computations: a summary of results. In *Proc. Intl. Conf. on Data Engineering, IEEE*.

Michael F. Worboys
Department of Computer Science
Keele University
Staffs ST5 5BG UK
phone: +44782583078
fax: +44782713082
email: michael@cs.keele.ac.uk

AN APPROACH TO OBJECT MODELLING OF A 'NAVIGABLE' DATABASE

Introduction

The aim of this position paper is to identify some of the functionality of a navigable database and begin to consider an object-oriented model of such a system. We assume a navigable database to be database system containing information about a transportation network. The following is a list of some of its properties:

1. The underlying spatial framework is a dynamic, labelled, directed graph embedded in a coordinatized surface. Nodes and edges of the graph will have geometrical, topological and other attributes. Examples of attributes of nodes and edges include:

- node geometrical attributes (surficial embedding), topological attributes (connectivity to other nodes, etc.), other attributes (traffic lights, etc.);
- edge geometrical attributes (surficial embedding), topological attributes (connectivity to other edges at nodes, direction, turn restrictions), other attributes (classification, travel time).

2. The spatial framework will be populated by a dynamic collection of objects (vehicles, sensors, traffic lights, etc.). Objects will be referenced to the underlying spatial framework. The generic data model, hinted at below, classifies this population into observers, informers and resources. Data on the state of the transportation system are captured by means of a heterogeneous collection of observers, collated within the DBMS, and coordination and control is relayed to objects in the system by means of informers.

3. The dynamic nature of the data model is crucial for the application, and a primary requirement is for access to a dynamic and up-to-date view of the network, its inhabitants, and their properties

4. Allied to the previous item, there is a requirement for responsiveness to changes in the network configuration and properties, so as to support operational decision making.

5. On a longer time-scale, there is a need for the capability to analyze histories and patterns of network use, so as to plan for the future.

In summary, the information space will be populated with data having spatial and temporal references, and collected from multiple, heterogeneous data sources. Complex data types will be dynamic and spatio-temporally referenced. A key technological objective is to extend the core functionality of DBMS to provide information system support for navigable database requirements.

Responsive GIS

It may be useful to draw a parallel between navigable databases and responsive GIS (Williams, 1995). There is a substantial class of information systems that has the following characteristics:

- large amounts of spatially referenced data are required to be readily available;
- a sizable proportion of the data is regularly updated from external data sources; some of the data are noisy, conflicting and incomplete;
- rapid decision-making is to be supported.

Examples of applications that require information system support that fall into the above class include transportation networks and environmental monitoring. Observations on the locations of entities in the domain, for example provided by sensors, act as data sources for the information system, which may then use the overview of activity for coordination and control.

Database support for dynamic spatial systems

There is a requirement that the DBMS can respond to a dynamic scenario, retain performance, maintain integrity and keep up-to-date information. The temporal dimension may be treated at a range of granularities. Temporal support will be required to handle queries on application domain object histories. Also, data sources will be updating the database with new information about locations and other properties of objects, raising many of the issues important for the performance of a real-time database, such as satisfaction of timing constraints applied to the transactions accessing the database. Current spatial database technology provides the capability for a system to hold and process a single, static snapshot of the locational configuration of an application domain, for example, the plan of a building or the spatial relationships of an urban road network. However, for dynamic domains where locations and other attributes are being regularly updated, the support available for proprietary DBMS can no longer be assumed.

As well as the entities that the system is tracking, it is important that the data sources (observers) are modelled correctly. Typically, each type of observer will have its own characteristics in terms of sampling rate (regular or irregular, interval), locational information (granularity, topology, etc.). A characteristic of much of the data for this class of applications is that they are noisy, conflicting, and incomplete. For example, sensors may observe that a vehicle was firstly on road A, not sensed for ten minutes, then sensed on road B and an intersection at roughly the same time. Deductive capability must be called on to establish a coherent pattern of

movement of the entity which does not conflict with observations of other entities and known domain rules. In order to build up such coherent patterns, history of past movements may be important and will need to be held in the system. Having established locational behaviour of entities in time, there will then come the ability to coordinate future activities on the network, controlling its behaviour so as to optimize desirable outcomes for the domain (e.g. an even spread of entities throughout the domain).

Some key classes in the object model

With regard to the general model of the data objects in the system, the following are some of the principal object classes for the application domains under consideration:

domain

The framework in which a given set of observers, informers, and resources are located.

observer

A source of data for the system associated with a single domain

informer

Each informer is associated with a single domain. An informer provides information from the system to certain of the resources in the domain in which it is situated

resource

Resources move within and between domains, and are tracked by observers in the domains within which they have a presence.

event

Aggregations of movements of resources. Events are usually the primary objects of interest in the application domain.

Reference

Williams, G.J. (1995) Templates for spatial reasoning in responsive geographical information systems. *International Journal of Geographical Information Systems* **9**(2): 117-131.

APPENDIX III: CONFERENCE DESCRIPTION AND PROGRAM

Conference on Object Orientation in Navigable Databases

Program

The National Center for Geographic Information and Analysis (NCGIA) and the California Department of Transportation (Caltrans) are convening a two-day conference/workshop to carry out a systematic and comprehensive review of the capabilities and benefits of object orientation in navigable databases for Intelligent Transportation Systems (ITS), March 15-16, 1996 in Santa Barbara, CA.

Navigable databases are defined here as databases capable of supporting such ITS applications as in-vehicle map display, synthesis of navigation directions, and direction of vehicles to a given street address.

The goals of the conference include:

---evaluating the range of products currently available

---assessing the total experience to date of Object Oriented (OO) databases

---comparing OO and more conventional approaches with specific emphasis on distributed storage, processing, and maintenance

The conference will provide a setting for identifying the associated issues and potential solutions regarding OO database design and implementation for transportation applications.

Workshop attendees will consist of:

1) Ten experts in the fields of OO theory, navigable databases, and OO GIS

2) Caltrans personnel

3) NCGIA researchers

4) Other invited participants from industry and academia with knowledge and interest in

the areas mentioned in (1) with a total of 40-60 individuals attending

Each of the ten experts will provide a two-page position paper reflecting views on the theory and implementation of object orientation for navigable databases. These position papers, along with a synthesis of the meeting discussions, will be published as a monograph following the conference. A listing of the position paper authors follows:

- Francois Bancilhon, O2 Technology
- Kurt Buehler, Open GIS Consortium

- Max Egenhofer, University of Maine, Orono
- David Fletcher, University of New Mexico
- Andrew Frank, Technical University of Vienna
- Michael Franklin, University of Maryland, College Park
- John Herring, Oracle Corporation
- Shashi Shekhar, University of Minnesota
- Alan Vonderohe, University of Wisconsin, Madison
- Michael Worboys, Keele University, UK

The following is an excerpt of the research proposal from which the conference is generated. It is provided here as background information:

Object orientation is a relatively new paradigm that is considered by many to be the basis for the next generation of software, both programs and databases. Although some Object Oriented Database systems exist (e.g. Versant), this field is in its infancy. This technology can have a significant impact on the GIS industry. Further such developments, especially OO databases, can have a significant impact on the long term approach for Intelligent Transportation Systems (ITS) and a navigable map database for the California Department of Transportation (Caltrans). Through a joint research effort between Caltrans and the National Center for Geographic Information and Analysis (NCGIA), it is proposed to carry out a systematic and comprehensive review of the capabilities and benefits of object orientation in navigable databases for ITS. To do this, the NCGIA will convene an expert meeting in the form of a two-day workshop. A core group of experts in object oriented databases, map databases, object orientation, and transportation applications will be invited from universities, industry and research groups across the nation. Invitations will also be issued to selected Caltrans and ITS personnel. A total of 40 and 60 participants is anticipated. The workshop will include position papers from the experts to be published as a monograph, along with a synthesis of the discussion and findings. Included in this monograph will be the synthesis of the impacts and potential future for Oodb for Caltrans. It is proposed to develop a system-distributed prototype which will involve an application at the Caltrans CAPTS lab in Sacramento, NCGIA and the UCSB-Caltrans Center for Interoperability. A report of this will be developed as well.

NCGIA/Caltrans Conference Program

Object Orientation in Navigable Databases

Friday, March 15, 1996

Radisson Hotel, La Cantina Room

7:45-8:30 Breakfast
8:30-9:15 Brief Introduction of All Participants
9:15-10:00 Keynote Address

- David Fletcher, Alliance for Transportation Research, "Navigating in a Digital Object Habitat"

10:00-10:30 Break
10:30-11:15 Keynote Address

- Andrew Frank, Technical University of Vienna, "Object Oriented Modeling for Geographic Information Systems in Transportation"

11:15-12:00 Overview of Meeting Objectives

- Ramez Gerges, Caltrans
- Professor Richard Church, UCSB

12:00-1:00 Poolside Lunch
1:00-3:00 Discussion by Panel Members of Position Papers and the Morning

Presentations:

- Francois Bancilhon, O2 Technology
- Kurt Buehler, Open GIS Consortium
- Max Egenhofer, University of Maine, Orono
- Michael Franklin, University of Maryland, College Park
- John Herring, Oracle Corporation
- Shashi Shekhar, University of Minnesota
- Alan Vonderohe, University of Wisconsin, Madison
- Michael Worboys, Keele University, UK

3:00-3:30 Break
3:30-4:30 Plenary to Discuss Tasks for the Breakout Groups for Saturday, 3/16

- Professor Michael Goodchild, Facilitator

5:00-6:30 Welcome Reception, Radisson Hotel El Cabrillo Room

<u>Saturday, March 16, 1996</u>

Radisson Hotel, El Cabrillo Room

7:45-8:30 Breakfast
8:30-8:45 Plenary to Organize Breakout Sessions
8:45-10:00 Breakout Sessions (3 Groups/Rooms: El Cabrillo, Gazebo, El Monte)
10:00-10:30 Break, El Cabrillo
10:30-12:00 Breakout Sessions Continued (3 Groups)
12:00-1:00 Poolside Lunch
1:00-3:00 Reports from the Breakout Sessions with Reactions from the Panel
3:00-3:30 Break
3:30-4:30 Closing Remarks--David Fletcher, Andrew Frank

Image not loaded **Go Back to NCGIA Introduction**
Image not loaded **Go to UC Santa Barbara menu.**
Image not loaded **Go to Conferences menu.**