Title
Object-Oriented Dynamic GIS for Transportation Planning

Permalink
https://escholarship.org/uc/item/2cx6k2db

Author
Golledge, Reginald G.

Publication Date
1996-09-01
Object-Oriented Dynamic GIS for Transportation Planning

Reginald G. Golledge

Working Paper
UCTC No 337
The University of California
Transportation Center

The University of California Transportation Center (UCTC) is one of ten regional units mandated by Congress and established in Fall 1988 to support research, education, and training in surface transportation. The UC Center serves federal Region IX and is supported by matching grants from the U.S. Department of Transportation, the California Department of Transportation (Caltrans), and the University.

Based on the Berkeley Campus, UCTC draws upon existing capabilities and resources of the Institutes of Transportation Studies at Berkeley, Davis, Irvine, and Los Angeles, the Institute of Urban and Regional Development at Berkeley, and several academic departments at the Berkeley, Davis, Irvine, and Los Angeles campuses. Faculty and students on other University of California campuses may participate in Center activities. Researchers at other universities within the region also have opportunities to collaborate with UC faculty on selected studies.

UCTC's educational and research programs are focused on strategic planning for improving metropolitan accessibility, with emphasis on the special conditions in Region IX. Particular attention is directed to strategies for using transportation as an instrument of economic development, while also accommodating to the region's persistent expansion and while maintaining and enhancing the quality of life there.

The Center distributes reports on its research in working papers, monographs, and in reprints of published articles. It also publishes Access, a magazine presenting summaries of selected studies. For a list of publications in print, write to the address below.

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

The contents of this report reflect the views of the author who is responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California or the U.S. Department of Transportation. This report does not constitute a standard, specification, or regulation.
Object-Oriented Dynamic GIS for Transportation Planning

Reginald G. Golledge

Department of Geography
and
Research Unit on Spatial Cognition and Choice
University of California, Santa Barbara
Santa Barbara, CA 93106-406C

Working Paper
September 1996

UCTC No 337

The University of California Transportation Center
University of California at Berkeley
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>Original Proposal</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reginald G. Golledge</td>
<td></td>
</tr>
<tr>
<td>II.</td>
<td>Working notes. Definitions and comments on Object-Oriented concepts</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>(A) Overview of map storage problems.</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Jon Speigle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(B) Object-Oriented definitions and concepts.</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Jon Speigle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(C) Object-Oriented systems and ATIS</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Jon Speigle</td>
<td></td>
</tr>
<tr>
<td>III.</td>
<td>Working notes Notes on an Object-Oriented database for use in an ATIS</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Jon Speigle</td>
<td></td>
</tr>
<tr>
<td>IV.</td>
<td>Contributions of GIS to Advanced Traveler Information Systems (ATIS)</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>Paper presented at the Western Regional Science Association Meeting,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mei-Po Kwan and Reginald G. Golledge</td>
<td></td>
</tr>
<tr>
<td>V.</td>
<td>Path Selection and Route Preference in Human Navigation</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>A Progress Report Paper prepared for COSIT Conference, Vienna,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>September 1995</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reginald G. Golledge</td>
<td></td>
</tr>
<tr>
<td>VI.</td>
<td>Computational Process Modelling of Disaggregate Travel Behaviour.</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>Paper presented at the IGU-Commission on Mathematical Models,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mei-Po Kwan and Reginald G. Golledge</td>
<td></td>
</tr>
<tr>
<td>VII.</td>
<td>Information Representation for Driver Decision Support Systems.</td>
<td>111</td>
</tr>
<tr>
<td></td>
<td>Mei-Po Kwan, Reginald G. Golledge, and Jon Speigle</td>
<td></td>
</tr>
<tr>
<td>VIII.</td>
<td>A Review of Object-Oriented Approaches in Geographic Information</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>Systems for Transportation Modeling. Manuscript submitted to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transportation Research, Part C.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mei-Po Kwan, Reginald G. Golledge, and Jon Speigle</td>
<td></td>
</tr>
<tr>
<td>IX.</td>
<td>Developing an Object-Oriented Testbed for Modeling Transportation</td>
<td>169</td>
</tr>
<tr>
<td></td>
<td>Networks Manuscript to be submitted to the International Journal of GIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mei-Po Kwan, Jon M Speigle, and Reginald G. Golledge</td>
<td></td>
</tr>
<tr>
<td>X.</td>
<td>Persons associated with the project</td>
<td>195</td>
</tr>
<tr>
<td>XI.</td>
<td>Papers and presentations</td>
<td>197</td>
</tr>
</tbody>
</table>
Section I

Original Proposal
UNIVERSITY OF CALIFORNIA TRANSPORTATION CENTER
COVER SHEET
RESEARCH PROPOSAL (August 1, 1994 - July 31, 1995)

Project Title: Object-Oriented Dynamic GIS for Transportation Planning

Project Director: Professor Reginald G. Golladge

Faculty Supervisor: (for dissertation grants)

Dept & Campus Address Department of Geography, University of California
Santa Barbara, CA 93106

Telephone Number: (805) 893-2731
Fax Number: (805) 893-3146
E-Mail Address: golladge@geog.ucsb.edu

OTHER KEY ACADEMIC PARTICIPANTS (Name, Dept, Campus Address, Phone)
Mei-Po Kwan, Graduate Student Researcher, Department of Geography, UCSB
(805) 893-2731, GSR V

One Graduate Student: 1 GSR II

PROJECT DESCRIPTION: (150 word abstract, with attention to pertinent key words.)

The purpose of this research is to develop an object-oriented GIS that can be used in conjunction with computational process model of travel behavior to develop a real world test bed for developing and testing products of policy measures such as IVES. Our GIS is designed to handle multi-dimensional network modelling, multi-level spatial modelling and multi-mode routing. It will have a dynamic component, and will use a variety of functionalities to help determine destinations, routes and activity pattern in the presence of different types of traveler information. Combined with recent development in mobile computer, our GIS is expected to handle fast changing values in location and temporal updates. In this phase, the project's focus will be on the conceptualization of the model, formulation of algorithms and programming work for implementation.

BUDGET REQUESTED: Total direct costs $ 82,448

Date of this proposal: 2/16/94
Object-Oriented Dynamic GIS for Transportation Planning

Description of the Proposed Research and Central Questions

Transportation planning has been concentrated on demand management as detailed in recent legislation such as the Intermodal Surface Transportation Efficiency Act (ISTEA). With the advent of Intelligent Vehicle Highway Systems (IVHS) and different management measures such as carpooling, vanpooling, and telecommuting, transportation modelling needs to incorporate analyses on these policy measures. Recent computer technology offers versatile functionality to model and evaluate impacts of these policies. Geographic information systems (GIS), as an integrating technology, has been increasingly used by DOT to handle transportation modeling and planning needs.

IVHS aims to utilize advanced information technology and processing to improve traffic efficiency. Both static and dynamic information are needed in an IVHS. First, a representation of the real world environment, with all the street network and information about location of related objects is required. Vehicle routing and navigation will be based on this network representation. Second, dynamic traffic information updating in a short time interval have to be included for accurate traffic forecast. With recent advances in technologies, fast location and temporal update are possible. Global positioning systems can accurately trace the location of a vehicle within several meters. Detectors of IVHS can provide traffic counts in a precise way. The central question is how to handle this fast temporal change and update of location within an efficient database system.

GIS have the potential to handle human movement in space and time. Existing GIS routines have been used to perform basic network algorithms such as shortest path, travelling salesperson problem and recently, location-allocation models. Applications commonly available in existing GIS software like TRANSCAD and ARC/INFO have largely been operated in data models that replicate the link-node network structure of the TIGER file. However, this structure does not satisfy the various requirements in today's traffic management, especially in an IVHS context.

First, the link-node data model is largely planar and it does not distinguish an intersection (e.g. when two surface streets intersect) from an intersection caused by an overpass or a subway line. Second, humans navigate not by link and node but by perceiving streets as continuous objects. Third, the network structure represented in these models is not detailed enough to represent lane information for navigation purposes without duplicating topology. However, duplicating topology in the GIS will increase storage for the system. As such, they do not handle routing in a multi-mode environment. For multiple-occupancy vehicles, routing may include HOV lanes in their route. It is not possible to reflect effects of HOV lanes if lane information is not contained in the GIS at all.

Kwan (1994), and Choy, Kwan and Leong (1994) have proposed an object-oriented approach and a method of indexing in handling the multi-level spatial modelling problem. Although object-oriented GIS has been discussed by Worboys, Heamshaw and Maguire (1990), and Gahegan and Roberts (1988), the multi-dimensional and multi-mode problems are still unsolved. Issues of fast location and temporal update in IVHS were unexplored. Naturally, no implementation of an object-oriented GIS for IVHS has been attempted in this manner.
Objective and Proposed Approach and Methods

The objective of this research is to develop and implement an object-oriented GIS data model for transportation planning that can handle multi-dimensional network, multi-level spatial modeling and multi-mode analysis in transportation planning. Also, the model needs to handle fast changing values in location and temporal updates for dynamic data in an IVHS. To extend the dissertation work by Kwan (1994), proposed tasks, approaches and methods are as follows:

1. To develop and implement an object-oriented GIS data model that represent the environment. The digital database on Santa Barbara available through the NCGIA at UCSB will be used as a starting point. The real world will be modelled as a collection of objects. Point, line and area objects are defined and their attributes and methods will be formulated to represent planar and non-planar elements in the transportation network and locations of activity. Lane information in each street will be included without duplicating topology. Research of how people perceive the environment will be used to formulate the design of the model.

In our conception, the street network will be modelled as a collection of line objects. Intersections would be further divided into planar and non-planar subclasses. Planar intersection will include junctions which cross at grade and non-planar intersection will include overpass and subway lines that cross surface streets. Each junction will be associated with point objects which indicate lane information. These definitions of objects and methods will be formulated as a foundation for routing in a real network for different modes of transportation. Navigation in this network will coincide with definitions of objects and will be interfaced to the users in a more natural way.

2. To tackle the location update problem. We propose to manage location (mobile) data in a GIS by a method of partitioning. The whole geographic area is partitioned into cells and each cell is designed to be served by a base station. In order to avoid massive frequent location update on the exact location of each user in the system, previous location or some "possible location" will be stored. However, this ignorance about the true location of users should be bounded. The bounded ignorance approach will guarantee the actual position of the object and the "possible location" will always be in the same partition (Imulinski and Badrinath, 1992).

Partition will be defined in terms of an individual's activity pattern. The pattern will be modelled by computational-process models. Based on work of Golledge, Kwan and Garling (1994), a feasible opportunity set will be defined for each individual. It would be further integrated with STARCHILD (Recker, McNally and Root, 1986a, 1986b) to identify a distinctive pattern for each user. Although maintaining information of activity pattern comes with a cost to the system, it is much more efficient and viable than frequently storing the exact location of each user. When paging to a particular user is needed, the bounded ignorance approach will reduce the search time significantly.

Microsimulation on the data set will be made to test the activity pattern of individual. Location error will be modelled by defining a corrector in the object-oriented data model. Data sets will be obtained from travel diaries for simulation of activity patterns.

3. To tackle the temporal update problem. We propose introducing temporal functions for aggregating fast changes in dynamic information. Time will be defined as a super class and will be associated with all the objects in the model. Pre-defined time aggregated functions in the temporal GIS, including average, maximum, minimum of a value on a time interval and functions for beginning time, end-time, user-defined time window, etc will be formulated for efficient operations.

4. To introduce and implement relational properties into the object-oriented model. Relational properties will function as object pointers to handle multi-level spatial modeling problems. Standard relational
operations such as selection, projection, join, union and difference will be incorporated into the object-oriented model. Induces will be built for major and minor streets, thus allowing efficient network abstraction and aggregation.

5 To experiment and establish spatial analyses to account for preferences and priority. Routing based on different criteria on turn penalties and preference for mode choice will be examined within the object-oriented data model. The model will capture individual preference in a natural and multi-dimensional environment. Mathematical formulation will be made to find the shortest path of covering activities using behavioral principles such as minimization of left-turns.

6 The object-oriented model with relational and temporal properties will be operationalized into a GIS system ARC/INFO Version 7 that contains object-oriented capabilities will be a starting point for implementation. Object-oriented database system such as O2 will be used to define the model. Programming for the model and the interface will be in C++, which allows for easy transportation to various platforms such as UNIX.

Relevance to Center's Theme

This project can be related to the center's theme relating to improving the quality of life for travelers. Our argument is that by providing a more comprehensive advising or routing system we can make driving less strenuous and less tension filled while at the same time influencing driver behavior under conditions of congestion, obstruction or hazard. This applies not only to automobile driving but also to multi-occupancy vehicles, thus allowing modeling and evaluation of impacts of demand management measures. In addition, our project is designed to improve the overall efficiency of operation of the existing transportation system by handling location and temporal update in an efficient manner.

Management Plan

We've anticipated that the principal investigators, Professor R. G. Golledge and Mei-Po Kwan, will focus on the design and development of the CPM and the object-oriented data model that lies at the heart of the GIS. The first part of this project, developing the object-oriented data model, will take place in summer of 1994. Ms Kwan will have the prime responsibilities for introducing temporal functions into the GIS and ensuring its dynamic function. A student will be used to clean up and prepare an enhanced digital data base of the local area which can be used as a test bed for all the functionalities developed in the object-oriented data base. In Winter and Spring quarters the ARC/INFO Version 7 GIS will be used as a starting point for implementing the object-oriented system, this again will be Ms Kwan's primary responsibility. Integration of models that identify individual travel pattern will then follow. Programming work will be carried out by a student who has background and interests in transportation, GIS and computer science.

Expected Product

It is anticipated that the object-oriented data base system will have been conceptualized and operationalized by the end of the project period. Algorithms for locating user or vehicle, and temporal functions will be incorporated into the model. We anticipate disseminating information about this by presentation of papers at a variety of transportation, regional science, and geography conferences and by publishing papers in appropriate journals in each of those academic areas.
References


Imelinska T and B R. Badrath (1992) Querying in highly mobile distributed environment *Proceedings of the 18th International Conference on Very Large Databases* pp 41-52.


BIOGRAPHICAL SKETCH

Reginald G. Golledge

Professor Reginald G. Golledge is a Professor of Geography at UCSB. He is a senior professor with interests in behavioral geography, including spatial cognition, cognitive mapping, individual decision-making, household activity patterns and the acquisition of and use of spatial knowledge across the life-span. His research has included work on adults, children, teenagers, mentally retarded persons, and adventitious and congenitally blind subjects. He has had more than two decades of experience in designing survey instruments and collecting data in both field and laboratory situations. He has published extensively in the literature of several fields including geography, regional science, and psychology. He has considerable experience supervision post-doctoral researchers, Ph.D and Master students, in managing large grants and contracts, and has been chair of his department.

Professor Golledge has written or edited thirteen books, fifty chapters in books, eighty-three papers published in academic journals, seventy-five miscellaneous publications including technical reports, book reviews, published research notes, and so on. He has presented one-hundred papers at local, regional, national, and international conferences in geography, regional science, planning, psychology, and statistics. He received an Association of American Geographers Honors Award in 1981. He was a Guggenheim Fellow in 1987-88. He is an Honorary Life-Time Member of the Institutes of Australian Geographers, and a Fellow of the American Association for the Advancement of Science. He has been Associate Editor and Editor of the journal Geographical Analysis and a Founding Editor of the journal Urban Geography. He has served on Editorial Boards of the Annals of the Association of American Geographers, Professional Geographer, Tijdschrift Voor Economische en Sociale Geografie, and has been a reviewer for many different journals, as well as the National Science Foundation, National Institute of Justice, National Institute of Health, Canada Council, The Australian Research Grants Council, and the European Science Foundation.

Professionally he has served on national research grants committees, on the AAG Honors Committee, and three times on the AAG Program Committee, acting as chair of that committee for the San Diego Meetings.

Selected publications include the following:


MEI-PO KWAN

EDUCATION

University of California, Santa Barbara (UCSB): Ph D in Geography (expected in June 1994)
Dissertation: "A GIS-based Model for Activity Scheduling in Intelligent Vehicle Highway Systems (IVHS)"

Specialization in transportation modeling and planning, geographic information systems, object-oriented data modeling and distributed systems, quantitative methods, and urban and regional planning.

University of California, Los Angeles (UCLA): Master of Arts in Urban Planning, 1989
Chinese University of Hong Kong: Bachelor of Social Sciences in Geography, 1985

RESEARCH AND WORK EXPERIENCE

Project Director
Department of Geography, UCSB, 8/93-7/94
Research project "A GIS Data Model for Transportation Modeling and Planning"

Co-Principal Investigator
Department of Geography, UCSB, 8/92-6/94.
Research project "A GIS-based Computational Process Model of Travel Destinations" with Reginald G. Golledge

Graduate Student Researcher
Department of Geography, UCSB, 12/90-7/92.
Assisted Reginald G. Golledge on research project "Activity-Based Models of Accessibility Planning Implications for Urban Subcenters"

Teaching Assistant
Department of Geography, UCSB, 1/92-4/92.

Computer Analyst/Programmer
Professional Services, School of Medicine, University of California, Los Angeles, 9/89-12/90

Redevelopment Consultant
GRC Redevelopment Consultants, Ltd. 9/88-9/89 Claremont, CA.

Research Assistant
Urban Planning Program, UCLA, 9/86-6/88
Advanced Materials Technology, Inc., Los Angeles, CA, 6/86-12/86.

Investment Opportunity Developer (Internship)
Southern California Ecumenical Council, Los Angeles, CA, 1/86-6/86

HONORS AND AWARDS

Dissertation grant, University of California Transportation Center, 1992-94
Graduate Student Fellowship, UCSB, 1992-94
Non-resident Tuition Fellowship, UCSB, 1990-94
Academic Award, UCLA, 1987
Non-resident Tuition Fellowship, Education Abroad Program, UCLA, 1985-88
First Class Degree Honors, Chinese University of Hong Kong, 1985

PUBLICATIONS


**PROPOSED BUDGET**  
Updated: 02/16/93

| **PI.** | Reginald Golledge, Professor of Geography |
| **Co-Investigator** | Mei-Po Kwan, Graduate Student Researcher |
| **AGENCY:** | UC Transportation Center, Berkeley |
| **PERIOD:** | August 1, 1994 to July 31, 1995 |
| **TITLE:** | Object-Oriented Dynamic GIS for Transportation Planning |

**SALARIES**

<table>
<thead>
<tr>
<th>R. G. Golledge, Professor A/S, PI.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 smr mos @ 50% @</td>
<td>$11,937 (Smr 94)</td>
</tr>
<tr>
<td>1 smr mos @ 50% @</td>
<td>$12,414 (Smr 95)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M. Kwan, Graduate Student Researcher V, Co-Investigator</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2 smr mos @ 100% @</td>
<td>$2,965 (8/94-9/94)</td>
</tr>
<tr>
<td>3 mos @ 49% @</td>
<td>$2,965 (10/94-12/94)</td>
</tr>
<tr>
<td>6 mos @ 49% @</td>
<td>$2,965 (1/95-6/95)</td>
</tr>
<tr>
<td>1 smr mos @ 100% @</td>
<td>$3,084 (7/95)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TBN, Graduate Student Researcher II</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 smr mos @ 100% @</td>
<td>$2,320 (9/94)</td>
</tr>
<tr>
<td>3 mos @ 49% @</td>
<td>$2,320 (10/94-12/94)</td>
</tr>
<tr>
<td>6 mos @ 49% @</td>
<td>$2,320 (1/95-6/95)</td>
</tr>
<tr>
<td>1 smr mos @ 100% @</td>
<td>$2,413 (7/95)</td>
</tr>
</tbody>
</table>

**SALARIES TOTAL**

<table>
<thead>
<tr>
<th>8/94-7/95</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,969</td>
<td>12,176</td>
</tr>
<tr>
<td>5,930</td>
<td>22,090</td>
</tr>
<tr>
<td>2,320</td>
<td>14,964</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>49,229</td>
</tr>
</tbody>
</table>

**BENEFITS**

<table>
<thead>
<tr>
<th>R. G. Golledge, Professor A/S, PI.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5,669 @ 3.40% (smr)</td>
<td>203</td>
</tr>
<tr>
<td>6,207 @ 3.40% (smr)</td>
<td>211</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M. Kwan, Graduate Student Researcher V, Co-Investigator</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5,930 @ 1.80% (smr actual)</td>
<td>107</td>
</tr>
<tr>
<td>4,359 @ 1.80% (acad yr)</td>
<td>78</td>
</tr>
<tr>
<td>8,717 @ 1.80% (acad yr)</td>
<td>157</td>
</tr>
<tr>
<td>3,084 @ 1.80% (smr actual)</td>
<td>56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TBN, Graduate Student Researcher II</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2,320 @ 1.80% (smr actual)</td>
<td>42</td>
</tr>
<tr>
<td>3,410 @ 1.80% (acad yr)</td>
<td>61</td>
</tr>
<tr>
<td>6,821 @ 1.80% (acad yr)</td>
<td>123</td>
</tr>
<tr>
<td>2,413 @ 1.80% (smr actual)</td>
<td>43</td>
</tr>
</tbody>
</table>

**Health insurance for 2 Grad Student Researchers:**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2 GSRs for Fall 94 @</td>
<td>$259 /qtr, 1 qtr</td>
</tr>
<tr>
<td>2 GSRs for Winter &amp; Spring</td>
<td>$259 /qtr, 2 qtrs</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>518</td>
<td>1,036</td>
</tr>
<tr>
<td>1,554</td>
<td>269</td>
</tr>
</tbody>
</table>
(Benefits Continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>8/94-7/95</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuition/Fee Remission for 2 GSRs: (Out-of-State Students)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 GSRs for Fall 94 @ $4,128 /qtr, 1 qtr</td>
<td>8,256</td>
<td>24,768</td>
</tr>
<tr>
<td>2 GSRs for Winter &amp; Spring @ $4,128 /qtr, 2 qtrs</td>
<td>16,512</td>
<td></td>
</tr>
<tr>
<td><strong>BENEFITS TOTAL.</strong></td>
<td></td>
<td>27,403</td>
</tr>
</tbody>
</table>

**TRAVEL.**

**DOMESTIC TRAVEL:**
3 RTs for Annual Transportation, Regional Science or Geography National Meetings x 1 person for each Conference
- Air coach @ $800/person x 3 trips: 2,400
- 4 days per diem including lodging @ $120/day x 1 person x 3 trips: 1,440
- 3 RT S B - U.C.Irvine to consult with UCI researchers
  - Private Car Mileage @ $.24/mile x 300 miles x 3 trips: 216
  - 1 day per diem including lodging @ $120/day x 3 trips: 360

**TRAVEL TOTAL.**

**OTHER DIRECT COSTS**
- Office Supplies: 200
- Telephone and Fax: 200
- Misc. Research Supplies including computer disks, color tape for printers, data sets and software: 1,000

**OTHER DIRECT COSTS TOTAL:**

<table>
<thead>
<tr>
<th>Cost Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Direct Costs:</td>
<td>$22,448</td>
</tr>
<tr>
<td>Indirect Costs @ 45% of MTDC:</td>
<td>$56,126</td>
</tr>
<tr>
<td><strong>Total Project Costs:</strong></td>
<td><strong>$107,986</strong></td>
</tr>
</tbody>
</table>

* Academic salaries are projected to increase by 4% on July 1 each year.

** Graduate Student Health Insurance Fee is provided to all Graduate Student Researchers and Teaching Assistants at 25% time or more.

*** Office supplies, telephone and fax costs are included in the budget for project research related activities.

**** This is the DHHS negotiated, predetermined, on-campus rate for Research Projects covering the period 7/1/94 - 6/30/97.
Section II

Working notes: Definitions and comments on Object-Oriented concepts

(A) Overview of map storage problems

(B) Object-Oriented definitions and concepts

(C) Object-Oriented systems and ATIS
II. Working Draft: Definitions and Comments on Object-Oriented Concepts

by

Jon Speigle

(A) Overviews of Map Storage Problem

The first section overviews the rationale behind map storage by the software. The second section discusses the format of the map when stored on disk. The third section discusses the interface. The fourth, describes how far along the various tasks are.

Map data structures:

Review of approach used on Blind Mobility Project:

My thinking as to how to process the map is based to a certain degree on the approach taken for the Blind Mobility Project. The maps of the Santa Barbara campus used on that project were stored on disk in Autocad’s drawing exchange file format (DXF). In this format, drawing "layers" are defined into which are then placed the drawing entities (e.g., points, lines, polylines, circles). The entity definitions include both the spatial information and layer to which the entity belongs. No topological information is defined in the format. The structures used to contain the loaded map conceptually resembled the DXF format --- lists of layers were maintained, with each layer including separate lists for the different types of entities contained within that layer. This approach allowed easy access to the entities on a layer-by-layer basis and by entity type. Attributes were stored in a relational database with the attributes linked to entities by their unique identifiers. The attribute information was stored in tabular form in text files because the dxf format does not support attribute storage.

For performance reasons, the map was partitioned into a spatial quadtree. The major limit on the Blind Mobility Project was in terms of memory availability running in DOS mode. The partitioning process began with the bounding box enclosing the two dimensional positions of all entities. This box was subdivided into four smaller boxes at the midpoints of the two sides. Dxf files were created into which were placed the entities contained by each of these smaller bounding boxes. Containment by a bounding box was based on the entity centroid. For the file associated with each bounding box, a running count was kept of the memory necessary to store the partition's entities. If a partition exceeded a criterion amount, its bounding box was subdivided and its entities were re-partitioned. The process thus created a quadtree with variable size leaves. The final tree defined a set of regions requiring less than a certain amount of memory when loaded. A description of the final partitions was stored for future, map access.

The Blind Mobility Project required landmark information and more local features to be available for immediate access. The more local information was used for collision avoidance, route planning and for purely informative purposes. The decision as to which leaves were loaded was based on the proximity of the user to the centroids of the bounding boxes. A buffer region, smaller than the leaf bounding box, was compared to the user position to determine when to re-configure the loaded partitions.
Review of approach for Object-Oriented Transportation Project:

Developing the project under Windows NT removes some of the restrictions faced by the Blind Mobility Project. Under NT, the operating system employs a virtual memory scheme allowing access of a 5 gigabyte address space. Relatively large maps could be loaded into this space unstructured. There is, however, a cost incurred each time the operating system swaps in a new segment of virtual memory from disk. Storing elements which will be accessed sequentially, near one another will reduce this overhead. For this reason, and the fact that local analyses are likely to be performed, map partitioning will still be necessary.

The approach outlined above is only one of several approaches for partitioning a map. Under NT, partitioning the map shall be easier due to the alleviation of memory restrictions. The large address space of NT allows a memory-based rather than a disk based version of the partitioning process. The basic approach would be to store the map information common to all partitions at the base of the partition tree. A spatial quadtree could then be constructed in memory into which the entities could be placed. The granularity of the spatial quadtree might be based on performance criteria such as the amount of time to access all entities in a given leaf.

Versant:

Overview:

Before considering exactly how the gis might be stored in Versant, it makes sense to briefly overview Versant's approach to object-oriented database design. In the grossest sense, Versant makes object-oriented data persistent across executions of a given program. It thus takes the place of a storage medium for the data used by a program. Versant specializes in storing hierarchically defined data which may include pointers to other data.

Versant is setup as a client-server database. The server manages the database and passes requested data back to the client. The server at present is limited in the ways that data may be specified by the client. Versant only allows items to be selected based on the values of class fields. No 'server-side methods' may be executed to select data. Selecting the elements which are contained by a bounding box would require the server to send the entire database to the client and the client would make the selection. This constraint will have an impact on the feasibility of different data designs for the gis.

Gis and Versant:

This is the part which needs to happen. A database will be created and the classes defining the basic gis entities will be registered with it. Instances of those registered classes may be added after the database is opened. The tricky part lies in how the classes are defined for indexing into the database. The ideas which we have bounced around look quite a bit like the partitioning schemes discussed above --- a way is needed to spatially organize the data such that accessing the entire dataset is avoided when selecting entities within a given region.

I will only consider one solution. The solution involves a tree structure where each node of the tree has associated with it a bounding box. The entities are assumed as belonging to a single layer and each leaf is assumed to have a list (for each entity type) for each layer of all entities contained by that leaf. The tree structure would be used commonly across all layers. An entity would be added to the tree by finding the leaf whose bounding box
The leaf would then add the pointer to the entity to the appropriate layer's entity list. This implementation would require:

- a QuadTreeRoot class which would store the layer information common to all leaves as well as a pointer to a QuadTree element (see below)
- a QuadTree class whose member variables include a bounding box, a pointer to the parent leaf, four pointers to leaves lower in the tree, and a list of 'LayerContents', one for each of layer of the gis (see below)
- a LayerContents class where each instance contains lists of pointers to entities, where N equals the number of different gis entities

The question becomes how the data is to be grouped. In the blind project, entities were layered based on item class and the same spatial quadtree was used for all layers. A leaf possessed a list layers and each layer contained a set of entity lists for all entities contained by that leaf’s bounding box. This approach to partitioning makes sense if all layers are to be accessed for processing according to the same local restrictions. If items are to be accessed largely independently, then it may be beneficial to partition different layers according to different spatial quadtrees. The first question is how the data will be classified, specifically, is the conventional layering approach appropriate? The benefit of taking an object-oriented approach to database storage might be in the ability to hierarchically define a classification scheme. Such a scheme would turn what in a conventional layering approach would be “join” operations into simple accesses. The process of accessing all entities of a given type higher up in the classification tree would be no more complicated than the access of the leaves of the tree.

**MapBase file format:**

The etak map is in the "MapBase" format. The manual indicates that Arc/Info, Autocad and several other gis packages are able to input MapBase formatted maps. The manual details the format sufficiently for routines to be written for processing the format.

The MapBase format consists of records with lengths of 256 bytes. There are five types of records, each given a single letter designator: C, D, A, S, and L. The C-records occur first in the file and store information about the map (such as the copyright notice, provider company, map version number, etc.). The D-records store the basic map data, referred to as "features". Features come in three flavors: points, lines and polygons. As far as I can tell, the D-records are all lines. Each D-record, however, defines two point features, the TO and FROM points. The records include the positions of the TO and FROM points. The records contain the positions of the TO and FROM points, topological information such as the "left" and "right" polygons, a name, a feature class, and additional attribute information. I will describe the attribute information contained in the D-Record below. In summary, it consists of information associated with the left and right polygons such as address information and census classifications. An A-record stores an additional name for a given element. A-records are used to assign multiple names to a given feature, most commonly used for roads. Each feature is assigned a unique identifier code (ID). Attributes are assigned to features via their IDs.

I have listed the attribute fields of the D-record in Table 2 and of the L-records in Table 3. The D-record is described on pages 29 to 48 of the manual. The L-record is described on pages 54 to 56. (I wanted to include a description of this sort in the source files anyway.) The C, A and S-records will only be described as above.
Feature classes:

The feature class-code included in the D-record serves the purpose of identifying features of the same type. From the types of feature codes, it appears that the MapBase format is primarily used for storing road information. The feature codes all have to do with the type of line element. Most of the codes have to do with a classification scheme for roadways (see pp. 61-65 of manual or Table 1, this document). The feature classes may be useful in terms of separating items into different lists within memory, effectively defining different entity "layers".

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>high speed ramp</td>
</tr>
<tr>
<td>1</td>
<td>interstate highway</td>
</tr>
<tr>
<td>2</td>
<td>primary state highway, subsidiary interstate highway, or subsidiary US highway</td>
</tr>
<tr>
<td>3</td>
<td>arterial roadway connects high-volume traffic generators and highways</td>
</tr>
<tr>
<td>4</td>
<td>collector roadway: connects light duty roads to arterials or highways</td>
</tr>
<tr>
<td>5</td>
<td>light duty roadway</td>
</tr>
<tr>
<td>6</td>
<td>alley or unpaved road</td>
</tr>
<tr>
<td>7</td>
<td>not used</td>
</tr>
<tr>
<td>8</td>
<td>railroad</td>
</tr>
<tr>
<td>9</td>
<td>low speed ramp</td>
</tr>
<tr>
<td>H</td>
<td>not used</td>
</tr>
<tr>
<td>M</td>
<td>restricted access road</td>
</tr>
<tr>
<td>N</td>
<td>positional accuracy unknown. used to identify features which are not stationary</td>
</tr>
<tr>
<td>P</td>
<td>political boundary</td>
</tr>
<tr>
<td>R</td>
<td>not used</td>
</tr>
<tr>
<td>S</td>
<td>shoreline</td>
</tr>
<tr>
<td>U</td>
<td>unknown feature usually a feature observable from an aerial photograph which was not classified</td>
</tr>
<tr>
<td>V</td>
<td>driveway or private road</td>
</tr>
<tr>
<td>W</td>
<td>walkway</td>
</tr>
<tr>
<td>X</td>
<td>shipping route</td>
</tr>
<tr>
<td>Y</td>
<td>Y-line: artificial boundary used to partition a large polygon into smaller, more manageable regions</td>
</tr>
<tr>
<td>Z</td>
<td>Z-line: artificial boundary at the boundary of the database</td>
</tr>
</tbody>
</table>

Table 1. D-record feature codes.

Attributes:

Attributes are stored either as L-records or within the D-records. A decision needs to be made as to how we want to structure our data. We will certainly have data structures for storing points, lines and polylines. These will include the spatial information, but one question is whether we want to preserve all of the attribute information included in the D records as well. It be easy enough to place the unused information into a relational table linked to the features by identification code.
The fields of the L-record are listed in Table 3. The L-records duplicate some of the information in the associated D-record. This allows the L-records (to a certain degree) to be processed independently of the D-records. The L-records may be treated as a table and standard spreadsheet techniques could be used to summarize or sort the L-records.

L-records are used in several different ways, depending on the value of the 'l_class' field. They may be attributes of line or polygon features, point features, or indicate virtual roads. When the L-record describes an attribute associated with a line or polygon feature, the value applies to the entire feature. When the L-record defines a point feature, the point

---

**Table 2. D-record contents.**

- **Basic information:**
  - id: identification code for line feature
  - name: name to be associated with this line feature
  - from_id: id of FROM point feature
  - from_x: longitude of FROM point
  - from_y: latitude of FROM point
  - to_id: id of TO point feature
  - to_x: longitude of TO point
  - to_y: latitude of TO point

- **General attributes:**
  - class: 1-byte feature classification code
  - oneway: direction of street
    - blank for two-way
    - 'F' for one-way flowing forward
    - 'R' for one-way flowing reverse
  - interpol: flag indicating whether the addresses were interpolated
  - pos_unknown: flag indicating whether positional accuracy is assured (BLANK) or unknown (1)

- **Left side attributes**
  - left_id: left polyline id
  - left_fradd: left side FROM address
  - left_st: left side state abbreviation
  - left_ua: left side census urbanized code
  - left_cnty: left side FIPS county or census area
  - left_mcd: left side census minor civil division
  - left_city: left side census city code
  - left_zip: left side postal code
  - left_tract: left side census tract code
  - left_block_group: left side census block group code
  - left_urban_flag: left side census urban place flag: U for urban, R for rural, blank for unspecified

- **Right side attributes**
  - same as above
feature is associated with either a line or polygon as is the line or polygon attribute, but it is assigned a specific location.

The classes for attributes associated with line elements are listed on page 63. They include such things as shipping routes, bridges, tunnels, pipelines and topographical features. None of the line feature attributes look particularly useful as far as our purposes.

The classes for attributes associated with polygon elements are listed on page 70. They largely classify regions as types of parks or bodies of water. Again, these attributes will not be incredibly useful to us.

The types of point features are the most extensive types of L-records, described on pages 71-74. Many of them are types of buildings or types of destinations: airports, railroad stations, bus stations, courthouse, hospital, schools, shopping centers, etc. Another range of codes serve to classify cities based on population (i.e., population > 5000).

<table>
<thead>
<tr>
<th>l_class</th>
<th>attribute class</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_id</td>
<td>identification code for this attribute</td>
</tr>
<tr>
<td>name</td>
<td>attribute value or point feature name</td>
</tr>
<tr>
<td>l_x</td>
<td>longitude</td>
</tr>
<tr>
<td>l_y</td>
<td>latitude</td>
</tr>
<tr>
<td>l_nicell</td>
<td>count of features within this attribute, always 1</td>
</tr>
<tr>
<td>l_dim</td>
<td>dimension associated feature: 0, 1 or 2</td>
</tr>
<tr>
<td>l_cell</td>
<td>id of associated feature</td>
</tr>
<tr>
<td>low_address</td>
<td>lowest number in address range</td>
</tr>
<tr>
<td>high_address</td>
<td>highest number in address range</td>
</tr>
<tr>
<td>address parity</td>
<td>parity of addresses in address range E for even, O for odd</td>
</tr>
<tr>
<td>state</td>
<td>state abbreviation</td>
</tr>
<tr>
<td>county</td>
<td>county code</td>
</tr>
<tr>
<td>mcd</td>
<td>minor civil division code, always 000</td>
</tr>
<tr>
<td>city</td>
<td>census place (city) code</td>
</tr>
<tr>
<td>zip_code</td>
<td>postal zip code</td>
</tr>
<tr>
<td>delta_x</td>
<td>change relative to l_x of vicinity</td>
</tr>
<tr>
<td>delta_y</td>
<td>change relative to l_y of vicinity</td>
</tr>
</tbody>
</table>

Table 3. L-record contents

**Determination of map contents:**

Etak did not provide any description of the contents of its database. For our research purposes this sort of information is essential. A procedure for summarizing a gis in this way potentially has commercial value, possibly as a consulting operation. The process is not necessarily complicated once the database may be processed. Even with only data structures defined which store the basic types of MapBase records it would be possible to summarize the contents. How complicated the summary process is would depend on the desired goals. With Versant, it might be relatively straight-forward to provide object counts where the types of items are hierarchically defined. A first pass would, of course, be to define vectors of the same length as the number of feature classes and to simply enumerate the occurrence of the instances.
Interface

Mathematical operations:

The basic mathematical operations of addition, subtraction, and multiplication are defined on matrix variables. Scalar multiplication and element-by-element multiplication and division are also permitted. Functions for operating on matrices element-by-element have been defined for most of the trigonometric functions.

The logical operations of equal-to, greater-than, less-than, greater-than-or-equal-to, less-than-or-equal-to, and not-equal-to have been defined. These operators may be included in if-statements to control program flow.

Commands:

Routines may be written and made accessible from the command-line by registering them with the interface. Each of these routines accepts as input a list of matrices and returns a separate list of matrices. The return list is empty unless the routine allocates a matrix in the course of its operations. In these cases, the return matrices may be assigned to variables or used as inputs to another routine.

Several commands have been defined which concern the GIS. These routines provide the highest level of access to the database. Some involve the addition of layers, circles, points, lines or polygons. These simply set up the argument list for the routines which are called when a map is read. Others involve the reading, partitioning, or merging of map files. In most cases the code which defines the command consists of only a thin layer between the user and the lower level routine which actually performs the operation.

The capability to manipulate and query the GIS from the command line will be one of the features undergoing the most development. As the database and operations on it are expanded, access to those operations from the command line will make their use more flexible. One of the most necessary features will be that of selecting entities based on some criterion. Some of the most useful criteria will be proximity and type. The capability to select elements of the database based on a query, assign the list of pointers to entities to a variable, and pass this variable to a subsequent routine would make the database entirely accessible to scripted operations.

One eventual feature of the command line will be its use in assigning attributes to entities. In an earlier version which included relational attributes, the entity could be specified either by its unique identifier or by selection with the mouse. The attribute type was then specified and the user was prompted for the attribute fields. The same approach may be used to add attributes which are stored in a hierarchical, object-oriented database. The difference would lie in the routines which actually made the attribute persistent. The lower level routines performing these operations will be created first to operate on attributes stored as text files. The command line only allows the "file" to be created interactively.

More general routines exist for saving, clearing, and listing the contents of the environment. These have not been extended to treat the GIS as easily as the matrices are treated, although such treatment is feasible. Separate commands handle clearing the GIS and clearing the environment variables.

Presently, the GIS exists as an integral part of the environment. It and its contents are not "types" of user-accessible variables. Converting to having layers, points, lines or polylines all as types of variables might be powerful, but it will have to wait...
Help facility:

Help may be displayed for any function by typing 'help name ', where name is the command name. A text file must exist which has the same name as the command name with the '.m' extension. This file is opened and any comment lines at the beginning of the file are displayed to the command window.

A fancier version, using Windows-style help would not be tremendously difficult given the numerous examples on how to do this in just about every book on Windows programming.

Summary:

I refer collectively to the parser, graphical user interface, registered C-functions, and user defined text functions as the 'interface'. The 'Interface' is defined as a C++ class. This base-class includes the basic features of the environment, command, edit, and graph windows. The 'Gis' is defined as a subclass derived from the base class, Interface. The Gis adds member variables and functions for reading, writing, displaying and maintaining lists of graphical features.

The final goal for the interface is for three types of windows to be used: a command window, text editing windows, and graph windows. The command window provides a prompt at which the user may type commands. The text editing windows will allow the user to modify text functions or scripts. The graph windows will be for graphical output. The command window is largely functional but still requires the cursor to be positioned at the end of the file prior to the user entering text. Most of the code for the editor and graph windows has been written, although requires further testing.

Project's current state:

Interface:

I have been running only under the Visual C++ debugger because of bugs having to do with quitting the application. The most important components are in place:

- Matrix variables may be defined at the command line
- Basic mathematical operations on matrices have been tested
- Built-in C-functions may be called and passed variables as arguments

Several features have not been tested since porting to Windows NT, although they were functional under the interface's previous incarnation:

- 'if-elseif-else' control flow statements
- 'for' loops
- User-defined text-file functions

The code for these features is roughly intact so I do not foresee particular difficulties in testing it. I just have to get to it.

GIS:

I have been working on the code for storing and displaying the map. The map is stored in roughly the format described above under Review of approach used on Blind Mobility Project. The map storage and code for manipulating it is object-oriented.
I have also rewritten the code for reading Autocad's dxf format to reflect the shift to object-oriented data structures. The dxf reading code itself has also been rewritten in an object-oriented fashion. This will allow as much code to be common across file-formats. My reasoning is that all of the formats will have the same basic structure: point/line/polygon entities and attributes attached via unique object identifiers. How the data is stored will be the same no matter what file-format is used for disk storage. An object-oriented approach to the file-reading process will allow for alteration of only a subset of the file-processing routines. At a relatively low level, the data from the file will be converted to the classes actually used to store the point/lines/polylines. All of the routines having to do with manipulation once the data is stored in those classes will be common across file-formats (e.g., element partitioning). In the previous version, those stages were not separated.

MapBase:

I have begun writing the code to process the MapBase map. The rewriting of the dxf processing code described above makes processing the graphical data of the MapBase map relatively straightforward — only the lowest level routines need to be written. The place where decisions need to be made is in how to treat the attribute information. The MapBase map contains details which may not be useful to us. The solution which I would favor is to choose which attribute fields to preserve and to store the others in a relational table linked to the entities by their object identifier. We really need to sit down and declare the classes which will be used for the network algorithms. In the interim, I am creating structures which basically mimic the fields of the MapBase records I will transfer the spatial data into the existing point/line/polyline classes and discard the rest of the attributes.
(B) Object-Oriented Concepts and Definitions

Object-oriented approaches have been reviewed and discussed in several books (Khoshafian, 1993; Khoshafian & Abnous, 1990; Booch, 1994). The major concepts are well-documented in these books and reports. Major constructs about object identity, encapsulation, inheritance, generalization, specialization, aggregation, and polymorphism are discussed also in this literature and in an increasing number of GIS related journal articles.

OBJECT IDENTITY

Object-oriented models look at the real world in terms of objects and how they behave. Objects are tangible elements in the real world. For example, the transportation network can be represented as different objects such as highways, intersections, etc. Each object has its own methods which specify different manipulation that are possible. By sending a request to the object, it invokes the methods of the object. For example, by sending a request to the methods of "streets", we can calculate the shortest travel time between two locations.

Note that existing GIS usually represents the transportation network in terms of link segments with nodes (the link-node structure). Each line segment would have unique identification. Usually a relational database would be used to link this spatial object with other attributes. In a relational database, the ID would change due to the addition or deletion of certain segments. Identity is not preserved in the object-oriented system, each object can have a unique ID and even if the object migrates several stages (i.e., change as the transportation network changes), it will still keep the unique identification through versioning. As such, we say that the OID is persistent. We can use versioning to represent the changes of, for example, census tract through different years (Worboys). Mainguenaud (1995) handles multi-level networks by arranging the OID in a hierarchical manner.

CLASSIFICATION

Objects can be classified into different object types. An object class is a software implementation of an object type. Classification in object models is an abstraction method and models the real world in terms of a hierarchical taxonomic structure. In the object model, a class can be further divided into subclasses. In this paper, we have experimented a system that comprises of spatial object hierarchy (Figure 1) which examines the network as a hierarchy of point, polyline, polygon and polyline list objects. The "node" is a subclass of the class "point" and "junction" is a subclass of the object class "node". We can further model the real world junction as "no stop", "signalled" and "stop sign" junction subclasses as in the real world transportation network.

GENERALIZATION

Several classes of objects, which have some properties and operations in common, can be grouped to a more general superclass. The subclass describes a specialization of the superclass. It is especially important for the abstraction of a real transportation network. For example, the object type "highway", "interstate" and "local" can be grouped to a superclass "road" in our transportation network example (Figure 2).
ASSOCIATION

A relationship between similar objects can be grouped together to form a higher level object (Brodie, 1984). The term "set" is used to describe the association, and the associated objects are called "members". This is important to GIS because it can model spatial relationships. For example, a highway is inside a city. Sometimes it may be important to take into account of the ordering of the collection of objects. An ordered collection of intersections can be grouped to form a road.

AGGREGATION

Several objects can be grouped together to form a composite object, which describes the higher level object. The model of this process is called aggregation. For example, the object type "highway" is an aggregation of lanes, road sign, exits and ramps.

Details of objects are suppressed in the constituent objects. Every instance of an aggregate object can be decomposed into instances of the component objects while keeping its own functionality. However, the operations on the aggregate may not be compatible with operations on its components (Egenhofer and Frank, 1989).

INHERITANCE

Inheritance is a major class-subclass relationship. A class inherits properties (data structure and methods) of its superclass. If it inherits properties from more than one superclass, it is called multiple inheritance.

Give an example.

POLYMORPHISM

Polymorphism generally represents the quality or state of being of an object, which is able to assume different forms. In a programming context, it means the same construct can be used to manipulate objects of different types. One type of polymorphism is called overloading. With overloading, the same operator (+) or "print" can perform different functions depending on the recipient's object class. It is a very important feature for transportation modeling. For example, the link-node structure assumes that the transportation network is represented as links and nodes. And there is no differentiation between these links and nodes. So the delay function, for example, cannot be easily modeled differently for signalized intersections versus non-signalized intersections, and highway versus local streets. With polymorphism in an object-oriented context, these different elements in the network can be modeled using different functions, but with the operator (such as "COST") used to calculate the delay function. Khoshafian and Abnous (1990) listed the advantages of overloading in terms of programming: (1) extensibility - the same operator applies to instances of many classes without modifying its code, (2) development of more compact code because there is no need to list out all the situation for different classes; (3) clarity - the code is more readable and comprehensible. (A penalty is paid for the run-time binding and/or type checking that must be done to guarantee correctness. Nevertheless, has tables or indexes could be used to lessen the performance penalty of dynamic binding.)

In some systems, parametric types can be used to create classes. Parametric polymorphism allows the construction of abstract classes, with type parameters. For example, a delay function can be formulated as COST[D] with D as a parameter of different types of distance measure. In transportation research, distance can be measured as either travel time or actual
distance. So the parameter D can be implemented as a type of integer (in terms of minutes of travel time) or as a type of real numbers (in terms of meters of physical distance). Both sets share the code as it is implemented in \text{COST}[D]. The procedure can be used to implement the same function for different types. Thus, one of the strongest advantages of parametric polymorphism is code sharing for generic types with the power of strong typing.

**ENCAPSULATION**

OO models enforce encapsulation and information hiding. The state of the objects can be manipulated and read only by invoking operations that are specified within the type definition. (ALSO, can incorporated R-tree or slay tree, we - quadtree).

**TYPES OF OO**

When we talk about object-orientation, we accept that the concept may involve different extensions. OO includes programming, database management, and modeling.

Mary Loomis (of Versant) espouses the view that the object-oriented approach to database management allows a more natural integration of the database with the object-oriented programming language used to implement the database.

**a. oo programming**

Instead of thinking in terms of loops, program steps and procedural code, a programmer will find the necessary object types (a category of objects) and work on how the objects can be accessed and modified. Analysts will create diagrams of events, triggers and operations that cause objects to behave in certain ways, or extend their behavior as appropriate. As such, the gap between system requirements and system implemention is reduced (Need definition and elaboration of terms).

Also. in conventional languages an operation such as "+" could be used for objects of different data types. In OO, this sort of polymorphism, is called overloading, and is available to every user-defined abstract data type.

**c. OO modeling**

OO models the real world as closely as possible (in the code). It is a conceptual tool that can be used in a problem domain. Unlike other modeling concepts which focus on functions, OO focus on the essential constructs within a problem domain. The OO model views the world as objects, starts with objects with distinct attributes and behaviors.

**Conceptual, logical and external- level**

Mangueaud (1995) modeled the logical graph topology to allow the handling of multi-level networks. The existing digital network databases do not distinguish a local versus a regional transportation network. The introduction of a Master-node (with respect to a Master-edges) in this object-oriented data model allows definition of a node as an abstraction of a sub network. The classes of the data model are organized in a tree. This tree represents the relation between classes and super-classes.

**c. OODBMS**

Database management systems are very important in handling transportation data, especially in view of the increasing size of digital databases. OO DBMS supports the object-oriented
paradigm and stores data and operations, rather than just data. It is efficient in storing complex objects. However, it still has some difficulties in establishing a standard query language. The limitations of existing DBMS, especially relational DBMS, have been discussed by Wiegand & Adams (1994). They began with the argument that relational models do not easily handle the complexity of geographic data and a better GIS can be built by (1) using a DBMS as a base for the system (2) using object-oriented modeling and (3) having an extensible system. They used an actual OODBMS (Object-oriented Database Management System) and transposed current relational GIS into an OO modeling. Comparisons are made between the two (Weigand and Adams, 1994).

Selected OODBMS currently available on the market are listed in Kemper and Moerkotte (1994). Some of them include Gemstone, 02, Ontos, Itasca, Versant, Matisse, Objectivity/DB, ObjectStore, etc. The data model, control concepts, architecture, related literature, and performance enhancements are discussed. Weigand and Adams (1994) examined the commercial OODBMS such as POSTGRES, Starburst, 02, Objectstore, and Gemstone. Advantages of OO according to their research includes multiple representation (e.g., different scales), multiple geometries and an increased amount of non-spatial information that can be stored within the feature object. Relationships between feature objects can be directly stored as part of the model rather than via an indexing key.


Gollu (1995) has reviewed some examples of software frameworks including.

a. Ptolemy. This is an object-oriented data flow based simulation tool (Ptolemy Manual, 1995). Ptolemy produces star maps in which objects called stars are represented by inputs, outputs, and pattern maps. The input output connections propagate messages that allow object interaction. Object evolution can be driven by some combination of time, events, or data tokens. It is most useful for specification and simulation of data flow among a static configuration of objects. However, it does not appear as suitable when large numbers of object relationships are concerned. It does not interface well with other systems and is still under development at the UC Berkeley College of Engineering.

b. COSPAN. This is a general purpose software tool that provides an automatic description syntax on a set of operations to combine. Its logical analysis consists of symbolic testing of a system for user defined behavior and in essence its analysis constitutes a mathematical proof of the stated system behavior (Har-El & Kurshan, 1987).

c. CSIM. This is again a general purpose C-based process oriented environment. It is specifically designed to simulate a discrete event system and its most frequent use relates to the behavior of data communication networks (Schwetnam, 1989).

EXAMPLE OF CREATING A ROAD HIERARCHY IN A RELATIONAL SYSTEM VS. AN OBJECT-ORIENTED SYSTEM:

We want to describe how it is possible to create a road hierarchy within a relational system such as Arc/Info. The purpose will be to illustrate the difference between 'Layer' and 'Class' within each system. A hierarchical type system may be created within a relational
system by assigning an attribute to each road which describes its type. Doing things such as displaying all 'Roads' when there is a road-type hierarchy is difficult. In an O0 system such an operation would be natural and simple.

ADVANTAGES AND DISADVANTAGES OF OBJECT-ORIENTED SYSTEMS

(c) It overcomes the difficulties of the planar data structure and will allow for a finer differentiation of various geographic objects in the system. Running routing and spatial search algorithms will become less problematic. (d) It will facilitate the representation and processing of a multi-level transportation network through introducing new classes across different levels and new functions for these classes: spatial search and queries will be handled more efficiently. (e) It will facilitate the integration of a wide variety of geographic objects and relations within a comprehensive geographic database.

Gollu (1995) argues that the validity of an object-oriented system model is likely to depend on:

• Our ability to validate the internal logic of the models
• The deployability of model component specifications

Further he goes on to argue that the principal requirements for valid and usable software systems include:

• An ability to associate physical and logical representations (modularity)
• The ability to add new components to the system with minimal code rewrite (openness, modularity, robustness)
• The ability to collect arbitrary statistics during simulation (openness, modularity)
• The ability to run simulations with acceptable performance (performance)
• The ability to adjust simulation granularity (modularity, openness)
• The ability to simulate up to 100,000 vehicles (performance)
• The ability to specify system behavior in a straightforward language (ease of use)

(Gollu, 1995, pp 31,32)

F. Versant medium

We used a particular O0 DBMS software - VERSANT - as our primary medium (see also the recommendation in Gollu, 1995). Part of using Versant software as the medium for an object oriented GIS is the development of the partitioning scheme. This is possible because of Versant's implementation as a client-server database. Our application is the "client" and would request data from the database "server." Of the several ways in which a client's request may be couched, one is to use simple equality/inequality statements on a given class's attribute values. The server will return class instances for which the conditional is met. We believe that this is the highest complexity of queries supported by Versant. More complicated, user-defined selection procedures are not supported yet because such procedures would need to reside on the server side. This is what is meant as "storing class methods on the server." Versant only provides the means to store the data, not the methods. The only way around this inability to conduct complicated queries (such as a bounding box comparison against an entity's position) seems to be to "recompute" these necessary computations. This is why it seems necessary to partition the map within Versant. The need is probably greater than it was for displaying the map.

The methods described above for creating hierarchically defined attributes and a hierarchically structured set of layers were mapped to Versant. Versant's role was to allow
for storage of class instances between sessions. In a relational database, this amounts to the storage on disk of an application's tables and their loading when next required. In much the same way that Versant scans the application's C++ header files to determine the class hierarchy, the code that has been written for creating class hierarchies does the same thing. The class definitions are used to create the "meta-class" information used to describe the memory layout of the attribute fields. The procedures to store class instances to disk were then written based on the meta-class information (i.e., the class fields are stored as a stream of bytes which is exactly what is needed to store the instance to disk). This approach is an interim solution while the partitioning, network and entity-attribute relations are worked out. All of these components seem required prior to storing class instances using Versant. First, the data to be stored in Versant must be instances of C++ classes. Second, using Versant as the storage media appears to require storing the map within the database. Storing the map requires partitioning. It is still possible even with a partitioning approach that map storage in Versant will prove prohibitively expensive in terms of space or access time. We will investigate possible solutions to this problem, particularly one that would be to store only the partitioning structure within Versant, store the map separately in some other file format, and then load the map into the partitions at run time.
(C) Object-Oriented Systems and ATIS

State of affairs:

I've managed to display the MapBase map and most of the interface that I was working on is functional. I describe the partitioning, layering and network hopefully not too cryptically or too confusedly to be useful. Partitioning the map seems required given its size. Layering seems to be one place at which object-oriented data structures are required. The network also needs to be partitioned and to take advantage of the hierarchical layering scheme. At the end I say a few things about Versant (basically, that I haven't gotten to it yet).

Layering:

Overview:

The map is divided into layers. For now, most of the layers are defined according to the attribute types defined within the MapBase format. Each L-record attaches information to a point, line or region and has a code which determines the type of attribute. A layer is created for each type of attribute; and each layer contains lists of the following types of entities: circles, lines, points, polylines, nodes, links and line attributes. The first four are the graphical entities common to most: CAD and GIS systems. The nodes and links are the entities used to define the street network and will be described below. The line attributes are records used to describe the line and link entities. They contain a subset of the information attached to each line element within the MapBase format: the left and right street addresses, census block numbers, topological information, etc.

Because the MapBase format defines many more attribute types than actually occur within the Santa Barbara County map, most of the layers end up empty. Space is conserved by allocating only the entity lists which actually contain items.

Partitioning:

It became apparent that some means of partitioning the map was required when displaying the zoomed-in map took nearly as long as displaying the entire map. The reason was that storing the map as a "flat" file required comparing each entity to the viewing box. The extent of the viewing box is determined by a translatable origin and axes scale factors. This problem has been alleviated by partitioning the contents of each layer. Partitioning allows the rendering of only those sections of the map which overlap with the viewing box without "touching" all of the entities. The increase in performance resulting from accessing only the necessary entities will be observed whenever a spatially localized subset of the map is to be referenced. This will occur not only during the tasks of displaying the map but during many other analyses, including route computation. Although I'm pretty green at this, partitioning must be an accepted, textbook procedure.

The description given above of a "layer" stated that each layer contained a set of entity lists, one for each type of entity. The partitioned entity lists replace that description. Each layer consists of either the lists of entities or a set of partitioned entity lists. A hierarchical spatial tree was used to implement the partitioning. (I'm not sure if it is a "quadtree" because I made the partitions at each level N-by-N rather than 2-by-2). The tree consists of a set of hierarchically arranged bounding boxes. The bounding box for the entire map is used as the boundary of the "root". In a sense, the unpartitioned entities are stored at the root of the tree because all of the entities are contained within its associated bounding box (by definition). The root bounding box is divided into N-by-N quadrants to define the 1st level. The
original box is the "parent" and the smaller, N-by-N boxes are the "children". Each child is itself subdivided into N-by-N boxes. This recursive subdivision is repeated until a specified "depth" is achieved. The bounding boxes at the last level define the tree's "leaves". The contents of each layer is partitioned by deciding which leaf's bounding box contains the entry. This decision is made using the entry centroid. The entry is copied to the appropriate entity list in the leaf and is subsequently removed from the unpartitioned list.

Although it is possible to use a different spatial partitioning for each layer, the same spatial quadtree is used for all layers.

At the moment, only the leaves of the tree are used to determine whether to draw each partition's entities. The more sophisticated and efficient approach would be to compare the bounding boxes at each level of the tree to the viewing box. A saving in processing time would arise whenever the remaining depth of the tree could be ignored because the bounding box at the given level did not overlap the viewing box.

**Hierarchical layering**

We require the capability to hierarchically relate different road types for use in route selection. I have written code to create these relations at the "layer" level. My aim was to define a "road-layer" class and then to "derive" more specific types of roads from this class. "Derive" is object-oriented lingo meaning to establish a "kind-of" or "parent-child" relation between objects. The derived or "child" class is said to "inherit" the properties of its "parent" class, including the parent class's attributes and its associated functions. (A relation might be established between a class "car" and a class "vehicle". It could then be said that "car is derived from vehicle" and that "a car is a type of vehicle").

The following text defines a road class which is a type of layer. The syntax of the definition is similar to that of C. The keyword, "class", indicates the beginning of the definition. The next word is the class name. If the class is derived, then the class name is followed by a colon and the colon by the parent class's name. The entries within the curly brackets define the attributes of the class. In this case, the "road" class consists of a name, speed limit and number of lanes.

```c
class road : layer {
    string name;
    int speed_limit;
    int n_lanes,
};
```

A layer for a more specific type of road can be declared by deriving from "road". An instance of class "highway" derived from "road" will have the attributes of both the "road" and the "highway" class. One additional attribute that the "highway" class might possess indicates whether the highway has a carpool lane.

```c
class highway : road {
    bool carpool_lane;
};
```

The class definitions are stored in text files which, when processed, define the layer hierarchy. The class definitions describe both the hierarchical relation as well as the attributes attached to "instances" of each class. In our case, the "instances" correspond to
items placed into the given layer. Defining the layer hierarchy is taken care of by the first line, "class road. layer". It is useful at the same time to define the potential attributes of entities contained by the layer. The attributes are only "potential" because the entities are not required to have defined attributes.

**Associations between entities and attributes**

There seem to be several approaches to associating entities and attributes, each having different drawbacks. One is for each entity to contain a pointer to its attribute. When the entity possesses an attribute, the pointer would be set to the attribute's memory address; when the entity does not, the pointer would be set to NULL. A second method, similar to the first, would be to place a pointer to an entity within each attribute. A third method would be to assign a unique identification code (i.e., "handle" or "object identifier") to each entity. Attributes would then refer to their entity using this code. A sin-le-bit flag might be set within an entity to indicate that an entity possesses an attribute.

I have not yet fully evaluated the costs of each method. The first two are pointer-based. They have the drawback that they allow only a single attribute to be associated with each entity (unless a more complicated scheme is implemented to allow lists of attribute pointers for each entity). One or the other approach would be preferred depending on from which side most of the computation would be done. If most computation involved only the attributes with only infrequent reference to the entities, then the pointers might be better placed within the attributes. An additional consideration is whether the entity attribute relations will be sparse or dense. The space consumed by NULL pointers under sparse conditions might be unacceptable. For the third method, actually associating an entity with its attribute requires searching the attribute list for the attribute having the entity's identification code. This required computation is what is made implicit within the pointer-based approaches.

**Generic attribute hierarchies**

The capability to define class hierarchies is not specific to layers, but may also be used to define hierarchies of any information (e.g., types of activities, types of destinations). The use of the word "layer" in the road-class definition causes some special handling that creates internal data structures related to the map. Another way to say this is that the keyword "layer" acts as a built-in "base" class. The trick will be to decide what we want hierarchies of, what the attributes should be, and how we will fill in the attributes.

**Network:**

I began by reading the MapBase map into the circle, line, point and polyline entities. The model that I am working toward will allow reading the street entities into the network rather than into the line and polylines. Sections of roads will be represented as "links", with the endpoints of the links defining the "nodes". The nodes correspond to either intersections or dead ends. A curvy but unbroken stretch of road will not be defined by nodes. (This approach is actually taken in the MapBase format by using the D-records to define the starting and ending points of a line, while intervening points are specified in the following S-records.)

Each "link" will have pointers to the two "nodes" at its endpoints as well as to the previous and next "links" (when the link is part of a continuous chain). Each node will have a list of links of which the node is an endpoint. Intersections will be nodes where at least two links are connected that are not continuations of the same chain.
Member variables for links (i.e., attributes)

Because the entire computational focus of the project is the network, the majority of the attributes will involve the network. I have been working under the notion that the network will be stored across the layers which define the road hierarchy. In this case, the attributes which would be attached to links and nodes would depend on the attributes given in the layer definitions.

Methods for links

One important question which I am not entirely clear on concerns the algorithm structure which would take advantage of the road hierarchy. I believe this issue might be resolved by considering the obvious functions of the road hierarchy. One function will be to allow selecting road types over each other (e.g., choosing a highway over an alleyway). This could be accomplished by assigning weights to the network links commensurate to the link's place in the road hierarchy. Effectively, this is what is done by placing the different road types into different layers --- layering the network is nothing more than "pre computed" sorting. Another function of the road hierarchy seems to be the definition of different attributes for the different levels of the hierarchy. Object oriented programming is designed to handle the case where the same functions are implemented for a number of vastly different objects. For example, two classes might both possess a member function for printing an instance. In one case an XYZ location would be printed while in the other a textual string would be printed. The routine which asked the instance to "print itself" would not care how or what the member variables were because it is up to the class to provide the means of executing the operation. What we are proposing to do is to design the functions for utilizing the attributes of different types of links for route-finding. These functions would be the class "methods" of each road class. The interface to each function would be the same (from the point of view of the abstract route-finding algorithm), but the actual implementation of the method would depend on the class. But of course, this is exactly what an object-oriented programming approach is designed to do.

One approach might be to define a base class (that is, a class from which all road classes are derived) which defines methods for such operations as computing a cost function for traversing a given link. If the derived class desired to improve or enhance the cost function by using its specialized attributes, then it would be up to that road class to provide its own unique method.

The first step which I have been working toward is to develop the basic network structure such that a route may be computed using purely positional information. Within the previous context, this traditional route-finding algorithm could be considered as the case where only the base class defined a cost functional and where the functional depended only on the link length and possibly the speed limit. A functioning traditional network will provide the framework on which to develop the algorithms which combine positional and non-positional information.

Versant:

You are probably wondering, "Where is Versant?" Part of using Versant as the medium for an object-oriented GIS will be the development of the partitioning scheme. This is due to Versant's implementation as a client-server database. Our application is the "client" and would request data from the database "server". Of the several ways in which a client's request may be couched, one is to use simple equality/inequality statements on a given
class's attribute values. The server will return class instances for which the conditional is met. I believe that this is the highest complexity of queries supported by Versant. More complicated, user-defined selection procedures are not supported yet because such procedures would need to reside on the server side. This is what is meant as "storing class methods on the server". Versant only provides the means to store the data, not the methods. The only way around this inability to conduct complicated queries (such as a bounding box comparison against an entity's position) seems to be to "precompute" these necessary computations. This is why it seems necessary to partition the map within Versant. The need is probably greater than it was for displaying the map.

The methods described above for creating hierarchically defined attributes and a hierarchically structured set of layers will eventually be mapped to Versant. I currently do not use Versant to implement it, but then again I haven't exactly added in the capability to store data. Versant's role will be to allow for storage of class instances between sessions. In a relational database, this amounts to the storage on disk of an application's tables and their loading when next required. In much the same way that Versant scans the application's C++ header files to determine the class hierarchy, the code that I have written for creating class hierarchies does the same thing. The class definitions are used to create the "meta-class" information used to describe the memory layout of the attribute fields. Although I have not yet written the procedures to store class instances to disk, doing so will not prove difficult given the meta-class information (i.e., the class fields are stored as a stream of bytes which is exactly what is needed to store the instance to disk) I am mostly thinking of this approach as an interim solution while the partitioning, network and entity-attribute relations are worked out. All of these components seem required prior to storing class instances using Versant. First, the data to be stored in Versant must be instances of C++ classes. Second, using Versant as the storage mediator appears to require storing the map within the database. I have already argued above that storing the map requires partitioning. It is still possible even with a partitioning approach that map storage in Versant will prove prohibitively expensive in terms of space or access time. One solution would be to store only the partitioning structure within Versant, store the map separately in some other file format, and then load the map into the partition at run time.
Section III

Working Notes: Notes on an Object-Oriented database for use in an ATIS
III. Working Notes: Notes on an OO-Database for Use in an ATIS Context

by

Jon Speigle

Introduction

This is a summary of what I have done and plan to do toward creating an object-oriented spatial database accessible from a command line. An advanced traffic information system is being developed within this framework.

The introduction continues with a very brief discussion of the concepts underlying object-oriented database systems. The following section discusses the system: early design decisions, the major classes, features of the command-line, and system evaluation. The final section summarizes what I am currently working on and the next steps. An appendix illustrates the command sequence of partitioning a map.

Object-oriented concepts

For now, I will take for granted that the following object-oriented concepts/buzz-words are understood: methods, inheritance, encapsulation, polymorphism, and aggregation. The concepts have been detailed in a number of texts as well as within the GIS community. The goal of the project was to study the application of these concepts to GISs, specifically transportation networks. The system therefore represents a blending of object-oriented concepts with a spatial indexing system.

Object-oriented database concepts

In addition to the above concepts, the texts on OODBs discuss the following: persistence, object identity, and object-oriented query languages. The features proposed for the query language include allowing

1) path queries
2) use of both class methods and class attributes
3) the results of one query to be used as input to a subsequent query
4) queries to search both the base and derived classes (with, of course, polymorphism on the derived class's methods).

Conceptual design of the system

Overall design decisions

In this overview I am more than anything organizing my thoughts on the relation between the programming language, the database language, and the interface. Things are at the point where I actually desire to attach aspatial information to the spatial data. On one hand I have C++ classes for holding the spatial data while on the other, I have a command-line interface for defining a layer hierarchy where entities placed into each layer are defined to have a
specific set of attributes. The issue which I hope to clarify is how the data will be made persistent. Persistence is an issue which I had delayed while developing the C++ framework for manipulating the map. It is time to resolve it.

Two substantially different approaches may be taken toward developing an OODB. The first is to use an object-oriented programming language (e.g., C++) as the data-definition and data manipulation languages (DDL and DML) with an additional data query language (DQL). The second approach is to create a DDL, DML, and DQL distinct from the underlying programming language.

A command-line may be viewed as merely one of several interfaces between the user and the database. Both of the above approaches might utilize a command-line for the purpose of querying the database or for controlling any non-metadata aspects of the system. Such an interface would allow accessing the database without an expertise in the application's programming language yet would still allow scripting of repetitive command sequences. A command-line interface allows more interactive development than the design compile-test cycle used for compiled languages.

On the other hand, there is a certain appeal to integrating the programming and database languages. Separating the two means that a great deal of effort must be expended on implementing the object-oriented concepts already provided by the programming language. A number of database vendors have taken this approach, supplying their customers with libraries for making a given object oriented programming language persistent (e.g., Versant). The goal of melding the programming and database languages could in fact be a criterion for selecting a programming language as some languages require compilation (e.g., C, Pascal, Fortran) while others operate from a command-line (e.g., Lisp, Smalltalk, Basic). The downside of this approach is that the database development must be performed by a programmer skilled in the object-oriented programming language.

What I am about to say probably applies only to the compiled programming languages and not to the interpreted-programming languages. I have in mind a database system programmed in a compiled environment where the end-user application has a command-line query interface. Without further description of the command-line language, the application could be of either type. The mechanisms used to create persistence could be incorporated into the class methods in the programming language. Extending the command-line language to include the capabilities of the DDL and DML, however, requires a major change in how the underlying system creates persistence because the database classes would now be definable at a different level than the programming language.

Versant provides both a C++ and a C-level interface. The C++ interface provides the means of making classes defined in the programming language persistent. The C-level interface provides lower-level access to the Versant persistence mechanisms. This interface could be used to add persistence to database classes definable at the command-line.

The graph class

The graph class is the programming construct whose methods control the visual presentation of the database contents. More will be added on this class. This class controls not only the visual display of information, but also stores all data associated with the map. The spatial entities stored in the graph are separated into 'Layers' based on object type. The layers may be hierarchical. In addition to the hierarchical component, each layer definition describes the attributes allowable for entities stored in that layer. The attributes which I have attached to each layer are accessible from the command-line only. This is problematic.
for creating C++ network algorithms but not insurmountable --- the attributes' accessibility from the command-line is after all written in C++.

**The layer class**

Each layer contains lists of the following types of entities: circles, lines, points, polylines, nodes, and links. The first four are the graphical entities common to most CAD and GIS systems. The nodes and links are the entities used to define the street network and will be described below.

**Hierarchical layering**

One goal of the project is to develop a hierarchy of different road types for use in route selection algorithms. These relations are made at the "layer" level. The specific goal was to define a "road-layer" class and then to "derive" more specific types of roads from this class (where "derive" is object-oriented lingo meaning to establish a "kind of" or "parent-child" relation between objects). The derived or "child" class is said to "inherit" the properties of its "parent" class, including the parent class's attributes and its associated functions (A relation might be established between a class "car" and a class "vehicle". It could then be said that "car is derived from vehicle" and that "a car is a type of vehicle".)

The following example defines a road class which is a type of layer. The syntax of the definition is similar to that of C++ although all of the capabilities of C++ have not been incorporated. The "class" keyword indicates the beginning of a definition. The next word is the class name. If the class is derived, then the class name is followed by a colon and the colon by the parent class's name. The entries within the curly brackets define the attributes of the class.

In this case, the "road" class consists of a name, speed limit and number of lanes.

```c++
class road : layer {
  string name;
  int speed_limit,
  int n_lanes;
};
```

A layer for a more specific type of road can be declared by deriving from "road". An instance of class "highway" derived from "road" will have the attributes of both the "road" and the "highway" class. One additional attribute that the "highway" class might possess indicates whether the highway has a carpool lane.

```c++
class highway : road {
  bool carpool_lane;
};
```

The class definitions are stored in text files which, when processed, define the layer hierarchy. The class definitions describe both the hierarchical relation as well as the attributes attached to "instances" of each class. In our case, the "instances" correspond to items placed into the given layer. Defining the layer hierarchy is taken care of by the first line, "class road: layer". It is useful at the same time to define the potential attributes of entities contained by the layer. The attributes are only "potential" because the entities are not required to have have defined attributes. The layer hierarchies actually utilized by the system will be described in the Specific Implementation section.
**Partitioning**

Spatial indexing is necessary for efficient access of large maps. One immediate benefit of spatial partitioning is in map display. At a given "zoom" level only a small portion of the map may intersect the view volume. Storing the map in a spatially unordered or "flat" representation requires comparing each entity to the view volume to determine whether to display it. This performance problem is alleviated by partitioning the contents of each layer. Partitioning allows the rendering of only those sections of the map which overlap with the viewing box without accessing all of the entities. The improvement in performance will be a function of the size of the view volume and the size of the partitions. The improvement will be observed not only during the task of displaying the map but during any other analysis which incorporates spatial constraints on the region of interest (e.g., route computation). This is a textbook point.

The description given above of a "layer" stated that each layer contained a set of entity lists, one for each type of entity. The partitioned entity lists replace that description. Each layer consists of either the lists of entities or a set of partitioned entity lists. A hierarchical spatial tree was used to implement the partitioning. (I'm not sure if it is a "quadtree" because I made the partitions at each level N-by-N rather than 2-by-2.) The tree consists of a set of hierarchically arranged bounding boxes. The bounding box for the entire map is used as the boundary of the "root." In a sense, the unpartitioned entities are stored at the root of the tree because all of the entities are contained within its associated bounding box (by definition). The root bounding box is divided into N-by-N quadrants to define the 1st level. The original box is the "parent" and the smaller, N-by-N boxes are the "children." Each child is itself subdivided into N-by-N boxes. This recursive subdivision is repeated until a specified "depth" is achieved. The bounding boxes at the last level define the tree's "leaves." The contents of each layer are partitioned by deciding which leaf's bounding box contains the entity. This decision is made using the entity centroid. The entity is copied to the appropriate entity list in the leaf and is subsequently removed from the unpartitioned list.

Although it is possible to use a different spatial partitioning for each layer, the same spatial quadtree is used for all layers.

At the moment, only the leaves of the tree are used to determine whether to draw each partition's entities. The more sophisticated and efficient approach would be to compare the bounding boxes at each level of the tree to the viewing box. A saving in processing time would arise whenever the remaining depth of the tree could be ignored because the bounding box at the given level did not overlap the viewing box.

**Recent additions**

The map may now be repartitioned; and items outside the partition tree are added to the layer's entity lists. The second will be elaborated upon. When an entity is outside the bounding box of the tree, it is added to the layer's separate lists. After partitioning, the tree contains all entities whose centroids fell within the tree's bounding box. The layer's separate lists contain the entities outside the tree's bounding box.
Future additions

Layer encapsulation

Proper 'encapsulation' of the layer class would be for the internal representation of the class to be unobservable. Whether a layer is partitioned or not should be transparent to the user, from either the C++ or command-line interfaces.

Iterating on the contents of a layer

I have been thinking about how to 'iterate' over the contents of a layer from the command line. This would allow user-defined functions to acquire pointers to entries which could then be individually queried. The full battery of features might include the capability to remove or add entities from layers. This of course hinges on the capability to assess the entity's spatial parameters from the command line.

The network

The network classes

The link class might be declared as follows. Links will consist of a pair of nodes and a set of intervening 'shape' points. (Allowing shape points within Links is in keeping with the MapInfo format and is probably a common means of reducing the number of Nodes.) The methods will include 'Cost' and 'IsInBBox' functions. The Cost member function will return the cost of traversing the link. The default method might simply compute the path length of the link.

The default cost function could be overridden by derived classes to produce more complicated behavior. The IsInBBox function will be used during partitioning and queries which incorporate spatial constraints.

```cpp
class Link {
    Node * n1,
    Node * n2;
    VertexList shape_points,
    float Cost();
    bool IsInBBox(BBox *bbox);
};
```

The node class is described below. Nodes will be used as the junction points between Links. A dead-end link will create a Node with a single Link. The crossing of two roads would create a node with four Links. Nodes will also have a cost function which describes the cost of traversing the node. This cost will in most situations be a function of the pair of links which traverse the intersection. For instance, if there were turn restrictions at the node, then these would be coded for within the Node. Maybe we will want to create a 'Turn restriction matrix' which describes the restrictions on travel between any pair of links. Different types of intersections might be defined, each having different cost matrices.
class Node {
    float x;
    float y;
    LinkList links;

    int Connect(Link *link),
    float Cost(Link *link1, Link *link2);
    bool IsInBBox(BBox *bbox),
};

The network formed from the Link and Node classes is continuous even though the links might be classified differently. Each Node has a list of the connected Links. The intersection of an off-ramp and a surface street would be represented as a Node. The links defining the surface street and the link defining the off-ramp would each be listed in the intersection's link-list. Polymorphism allows the link-lists to contain pointers to any class derived from 'link'. The network algorithm may crawl along the network with no knowledge of how the different links are classified.

One of the things that I want to look at is a route finding algorithm (probably just minimal distance to start with) to get a feel for what the algorithm requires of the Links and Nodes. That is what I have in mind by including a 'Cost' function in the Node and Link classes. Other equally necessary member functions might become apparent by considering a class of network algorithms.

The part of the problem that is still fuzzy concerns how the class methods will be declared. The issue is whether the class structure needs to be written in C or whether it should be definable from the command-line.

Integrating the network classes with the Gls, Graph, Layer and Partition classes

Two possible representations were considered for integrating the Node and Link classes with the Graph. The first was to maintain a list of nodes and links within each layer. The second was to maintain only a single node layer with the links scattered throughout the layer hierarchy.

The second approach was chosen because it acknowledges a basic difference between the nodes and links. We use the nodes to represent street intersections and the links to represent different types of roads. Our goal is to develop a hierarchy of street types, each of which possesses different attribute information. A two-road intersection does not belong to either street layer. We may eventually consider a hierarchy of different types of intersections. Placing the nodes into their own unique layer recognizes the problem of classifying intersections using the road-type hierarchy. This choice also allows iterating over all nodes in the database.

Links and nodes are stored as part of Layer::contents (along with the line, polyline and point entities). Layer::contents is a list of lists.

Each type of entity list is only allocated when an entity of that type is added to the layer.

The entities of each list are allowed to have an associated attribute set which is defined by the derivation of the class from the base Layer class.
Part of the process of creating the network will be to add nodes to the node layer only if the node does not already exist.

**Creating the network topology**

Storage as lines and polylines destroys the topology. Reconstructing a network topology from a set of lines and polylines will be a painstaking process.

For the moment, consider an isolated line or polyline as the input to a routine converting the entity to a link and pair of nodes.

There are assuredly a host of approaches to creating a network topology. The following algorithm is very simple, probably horribly inefficient and scarcely worthy of the space used to present it. The following code incorporates a polyline into an existing network.

```cpp
nodeLayer = FindLayer("node")
v1 = v.get(v.count-1)
v2 = v.get(0)
n1 = nodeLayer->AddNode(v1->loc,v1->handle)
n2 = nodeLayer->AddNode(v2->loc,v2->handle)
link = AddLink(n1,n2)
// More link initialization code
```

The starting and ending vertices of the polyline will be taken as representing the nodes of the link. The intervening vertices will be used only as 'shape points'.

The first step in creating a link is to check whether the starting and ending vertices already exist as nodes. If they do, then their pointers are acquired. If they do not, then nodes are created at the vertices' locations. A link is created using the node pointers. The link shape points are set to the intervening vertices.

Because creating the network requires checking whether each node exists, the time required to add a link will increase as the number of nodes increases. Partitioning the map prior to creating the network will greatly decrease the conversion time. It will be important to identify the partition to search given the node's position.

**A partitioned network**

When a layer is unpartitioned, an unpartitioned network is created. When a layer is partitioned, the same partition tree is used for the network as is used for the other entities. A node is assigned to a partition based on its location. A link is assigned to a partition based on its centroid. This might result in a link and its nodes being stored in different partitions.

One goal of the system should be the encapsulation of the layer contents. This means that the partitioned layer should have the same functionality as the unpartitioned layer. Adding/removing entities, display. The interface should be the independent of the partitioned/unpartitioned state of the layer. The following code, excised from Layer::CreateNetwork routine, leads to this functionality.
// Create node layer if needed
if (nodeLayer does not exist) {
    create nodeLayer
}

// Convert partitioned layer.
if (partitioned) {
    for (all partitions) {
        // Convert lines
        for (all lines) {
            CreateLinks(line)
            Remove(line)
        }
        // Convert polylines
        for (all polylines) {
            CreateLinks(poly)
            Remove(poly)
        }
    }
}

// Convert unpartitioned layer.
else {
    // Basically do only the inner loops over lines and polylines
}

Note that the Layer class has not been modified to add elements to the partitions once the map is partitioned. This means (at the moment) that converting the map to a network must precede partitioning. It is time to fix this. Functions should be added to the Layer class for testing whether the layer is partitioned (i.e., Layer::IsPartitioned()). The routines for adding entities should be modified appropriately to add the item to Layer::partitionTree or Layer::contents.

Path algorithms

A discussion of network algorithms is needed. The discussion should focus on how an algorithm might take advantage of the object oriented properties of the network, layer and partition classes.

An analysis of a network algorithm will assist in deciding upon member functions for the network classes. As discussed in the two approaches section, the member functions will be made accessible from the command-line.

Part of creating command-line network algorithms is the capability to iterate on the layer contents. I should look at the SparsePQT algorithm and determine what functionality is missing in the interface to implement it from the command-line. As part of this, I will want to discuss how partitioning affects the algorithm. I should photocopy the pages of the algorithm discussion from the text and step through it and indicate where the changes are.

Persistence

The persistent store is the mechanism which provides storage of class instances across sessions. One of the crucial components of the persistent store is the partitioning scheme as it will be important to optimize the requests for data from the store.
We will use Versant's system as a client-server database. Our application is the "client" and requests data from the database "server". Of the several ways in which a client's request may be couched, one is to use simple equality/inequality statements on a given class's attribute values. The server returns class instances for which the conditional is met. I believe that this is the highest complexity of queries supported by Versant although more sophisticated object-oriented query languages are certainly under development (i.e., ODMG specification of an OQL). More complicated, user-defined selection procedures are not supported yet because such procedures would need to reside on the server side. This is what is meant as "storing class methods on the server". Versant provides the means to store the data, not the methods. The way around this inability to conduct complicated queries (such as a bounding box comparison against an entity's position) is to "precompute" the necessary indices. This is why partitioning the map is so crucial when it comes to the persistent store.

The methods described above for creating hierarchically defined attributes and a hierarchically structured set of layers will eventually be mapped to Versant via its "C" interface. Versant's C++ interface provides the means of scanning an application's C++ header files to determine the class hierarchy. The class definitions are used to create the "meta-class" information used to describe the memory layout of the attribute fields. This metadata is available within the system after the command-line parser scans a class definition. Calls through the Versant's "C" interface will allow registration of the command-line's metadata.

The command line

General features

The general features which could be discussed in this section include: variables, user-defined text files, built-in functions, class syntax, mathematical and conditional operators, and flow-control statements (e.g., if-elseif-else and 'for' loops).

It should be noted that the capability to define class hierarchies is not specific to layers. Classes may be defined for any information (e.g., types of activities, types of destinations). The use of the word "layer" in the road-class definition leads to "special handling" that creates layers within the current graph. Within a class definition, the "layer" keyword acts as a built-in "base" class. If other situations arise where it becomes desirable to derive from other built-in classes at the command-line, this functionality may be added to the system. It may become necessary to allow multiple inheritance for instance.

Queries

Many of the object-oriented query languages presented in the literature presume the availability of a list of all instances of a class. A query is then formulated over this set. The 'layer' class within our system performs a similar function. Each layer describes a 'class'. Entities in that class are placed into the layer's entity lists. The spatial partitioning scheme is a method of clustering entities to enhance performance for queries having spatial constraints. This issue is what distinguishes geographic information systems from other database applications. Does what I am laying out have a history within the RDBMS or non-object-oriented GISs?

Partitioning the map contents might be considered a case of using the results of previous queries as the input to further queries. The first set of queries would be to select all entities within a set of bounding boxes. These results would be stored and associated with the
bounding boxes of the query. Further spatially constrained queries could be made more efficiently by breaking them down into a first step of finding which previous result is applicable to the present spatial constraints and a second step of executing the query on that previous result.

Queries return pointers to the items satisfying the query constraints. I am still working on allowing access from the command-line to 1) the instance's attributes, 2) built-in 3) user-defined functions associated with the instance. The full battery of features might include the capability to remove or add entities from layers. This of course hinges on the capability to access the entity's spatial parameters from the command-line.

**Specific Implementation**

**Data sources**

Data sources:

<table>
<thead>
<tr>
<th>Source</th>
<th>Coverage</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB city</td>
<td>city of SB</td>
<td>ACAD</td>
</tr>
<tr>
<td>SB county</td>
<td>SB county</td>
<td>ACAD</td>
</tr>
<tr>
<td>SB county</td>
<td>cities of SB and Goleta</td>
<td>TMODEL2</td>
</tr>
<tr>
<td>ETAK</td>
<td>SB county</td>
<td>Mapinfo and ARC</td>
</tr>
<tr>
<td>Navtech</td>
<td>?</td>
<td>ARC</td>
</tr>
<tr>
<td>Wessex</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

**Layer hierarchy**

The map is divided into layers. For now, most of the layers are defined according to the attribute types defined within the MapBase format. In the MapBase format, L-records attach information to a point, line, or region. Each L-record has a code which determines the type of attribute. A layer is created for each type of attribute. But because the MapBase format defines many more L-record types than actually occur within the Santa Barbara County map, most of the layers end up empty. Space is conserved by allocating only the entity lists which actually contain items.

The initial set of attributes which we would like to incorporate into the database are the following:

1. road widths
2. road direction
3. median strips
4. number of lanes
5. speed limits
6. turn options
7. over/underpass locations
8. road gradient
9. intersection type: stop signs, signals

10. bicycle paths on side of streets

11. signal phases at roads of unequal status

System evaluation

Quite a bit of effort could be expended testing the various parameters of the OOGIS (e.g., partition depth). A suite of tests should be developed. Performance problems should be identified and dealt with, either by parametric tuning or by major revision. The test suite should be relatively automated so that the relation between system parameters and performance could be determined and optimal points chosen.

Of course, once the system is optimally configured, different network algorithms should be compared. The time taken by the network algorithm might even be used as the criterion for system tuning. The different algorithms might then have different sets of corresponding optimal parameters.

Game plan

I am currently working on

1. Converting the roads in the MapInfo map into the network structures rather than reading them into line and polyline structures.

2. Saving attribute information stored in the MapInfo map and as command-line accessible instances (e.g., address ranges, street direction).

The order of the next steps seems flexible. Some of them involve completing partially developed features of the OOGIS. Some of them concern the ATIS

1. Actually pin down the layer hierarchy and the layers’ associated attributes. This stage should precede the addition of data from other sources because the slots for the other sources can be built into the classes.

2. Convert the map to persistent storage. Versant’s C interface will be used.

3. Develop ‘query’ methods which incorporate both spatial and value constraints. This shouldn’t be too difficult as I have already written the procedures to access instance’s attributes. The query procedure will simply apply some conditional to some subset of the partition tree’s entity lists.

4. Incorporate additional data sources into the persistent store. This might need to follow the improvement of the query capabilities. Part of placing the data into the store will be finding the object to which the data should be assigned. Some form of geocoding will be needed.

5. Implement a simple route selection algorithm which does not take advantage of the object-oriented capabilities.

7. Define tests to evaluate Versant's performance as a persistent store.

8. Add mouse support to allow selecting map regions individually or by bounding box.

**Appendices**

**Partitioning the Etak map:**

1. Create a 'graph window' by typing 'graph' in the command window. (The prompt is the '=>' symbol. Type commands at the prompt. I haven't protected the prompt from the user hitting backspaces and deleting it, so be careful.) Hit the return key after typing 'graph'. A second window should appear labeled 'Graph 1'.

2. Move the graph window and enlarge it so that it isn't overlapping with the command window. You might want to enlarge the frame first.

3. Click back into the command window with the cursor at the fresh prompt.

4. Load the map by typing 'readfile("sb.mbs")'. This should take a little while. When it is finished, there will be some summary information printed in the command window and the map should begin drawing itself in the graph window.

5. In order to partition the map, the bounding box for the entire map is required. Type 'b=computebbox' to create a variable 'b' which contains the bounding box.

6. Now type in the command to partition the graph. 'partition(b,4)'. This will take a little bit and a new prompt should reappear when it is finished. (The graph won't redraw automatically.)

7. Zoom in on the map 'zoom(10)'. You should see some lines which are the partition boxes. I think it is zoomed in on the ocean so there might not be much else.

8. Scroll the graph upwards by clicking the mouse on the 'UP' arrow at the top of the graph window's vertical scrollbar. After a few clicks, goleta should come into view.

9. You may zoom in and out by typing 'zoom' with parameters greater than or less than one, respectively.

10. Oh yeah. The map is flipped because of the sign convention that longitude increases from right to left. To display the map correctly, type 'flipx'.
Section IV

Contributions of GIS to Advanced Traveler Information Systems (ATIS)

Contributions of GIS to Advanced Traveler Information Systems (ATIS)

By

Mei-Po Kwan
and
Reginald G. Golledge
Department of Geography
and
Research Unit in Spatial Cognition & Choice
University of California Santa Barbara
Santa Barbara, California 93106

Paper presented at the
Western Regional Science Association Meeting
Tucson, Arizona
February 23-27, 1994
**Introduction**

Transportation planning has been turning away from the solutions of building highways and transit routes to changing people's travel choices and making more efficient use of existing facilities. With the recent research focus on Advanced Traveler Information Systems (ATIS), it is imperative to understand the travel behavior of people and the information required to change their travel choices.

IVHS (Intelligent Vehicle Highway Systems) have aimed at the utilization of advanced information processing and communications technologies to achieve improvements in travel efficiency and safety. As one of the major components of IVHS planning, ATIS essentially is targeted to assist drivers in trip planning and decision making on destination selection, departure time, route choices, congestion avoidance and navigation. ATIS is to provide travel information for two types of traveler, namely, the pretrip traveler and the en-route traveler. It deals with spatial decision-making at an individual level.

Under a IVHS context, we need to know not just what type of information is needed and how people process information, but also how the individuals interact with the environment in real time, especially how people make choices in activity scheduling, and routes and destination choices.

Micro-simulation techniques have been widely used to model the driver's response to information in reducing recurrent and non-recurrent traffic congestion. For example, computer-interactive simulators have been developed to study commuter behavior through laboratory experiments as an alternative to real world situations. Bonall et al. (1991) developed IGOR (Interactive Guidance on Routes), which investigates factors affecting drivers' compliance with route guidance advice, such as quality of advice and familiarity with the network. Allen et al. studied the impacts of different information systems on drivers' route diversion and alternative route selection. Adler et al. (1992) suggested a framework to model individual en-route behavior in response to real time traffic information based on conflict assessment and resolution theories. Nevertheless, these simulators are deterministic, with all traffic conditions and consequences of driver actions predetermined. They interact with one subject at a time, without considering the
interaction among drivers. Also, Allen et al. and Adler et al. assume perfect and static information, which is not realistic in real traffic situation (Chen and Mahmassani, 1993).

Route choice in the presence of information is studied by Ben-Akiva and Kaysi (1991), who propose a framework for modeling the process of drivers' information acquisition and behavior. Khattak et al. (1992) investigated commuters' diversion propensities and evaluated how drivers use real-time information. Lotan and Koutsopoulos (1992) developed route choice models in the presence of limited information, using concepts from fuzzy set theory, fuzzy control and approximate reasoning.

The simulation-assignment model by Chen and Mahmassani (1993) consists of a traffic performance simulator, a network path processing component, and the user decision-making component. Driver behavior and response to real-time traffic information systems is regarded as a complex process involving human judgment, learning and decision-making in a dynamic environment. The decision of one driver in the system is affected by other drivers, and this interaction is highly nonlinear. However, the context of the paper is restricted to the morning peak-period of commuters in congested traffic corridors.

Kaysi et al. (1993) developed an integrated approach to vehicle routing and congestion prediction for real-time driver guidance. The system design consists of a surveillance system, a congestion prediction element (COP) and a control and routing element (CAR). Congestion prediction is based on infrastructure data, historical origin-destination data and the updated OD data. In the system, COP should provide CAR with projected traffic condition as the bases for prediction. A dynamic traffic assignment model (DTA) should be used for COP. Lastly, CAR should maintain guidance/prediction consistency.

However, these researchers have been concerned with traffic condition as the mere factors for travel decisions. They either take traffic condition as given, or try to model the traffic condition and the drivers' response. The fact that travel is a derived demand and travel is generated by the decision of individual to participate in activity is largely absent. In the routing model for planning a trip, however, it is usual to assume a fixed (known) origin and destination. The problem is then how to find the best route for this link, largely by the shortest path algorithm. Nevertheless, route choice in the presence of information depends on how activities are scheduled in time and space. The origin and destination may be different if there is a change in the trip chain due to adaptation.
to real traffic conditions. In the case of diverting to another route in face of congestion or accident, the scheduling of activities may be adaptively changed, resulting in changes on routing in space and time. Prediction of traffic condition would be affected as such. Activity scheduling is thus an important component missing in current ATIS research.

Most research also ignored the interdependencies of travel decisions. Interrelated decisions for pretrip traveler include the decisions required by each household members. For the en-route traveler, decisions made by other drivers in the system are important. The consistency issue is then important (Kaysi et al., 1993). However, they propose a concurrency check system but this problem is not dealt with directly in the database design. In addition to quality of traffic information provided to a traveler, the assurance of privacy is also important. Value-added information like the yellow pages and tourist guide, including locations of restaurants, tourist spots, etc. would greatly improve the usability of the system. This also can hardly be achieved if the system just concentrated on the network system and traffic conditions.

Proposed model of integrating CPMs into GIS

Most ATIS models assume that a traveler would take the rational approach to minimize travel distance and/or time in traveling, without paying attention to individuals' preference in route choices. Also, the facts that individuals are opportunistic, and that activity scheduling is highly dynamic and adaptive to real world traffic conditions, are not considered. In order to build a system that take into account the preference of individuals and their adaptive behavior, a computational-process modeling (CPM) approach is adopted. CPM examines how an individual interacts with the environment and adapts to changes. It allows us to investigate the relationship between travel behavior and the cognitive representation of the environment and describes explicitly the steps in the process of individual decision-making in the form of a computer program. It is a flexible and recursive process for modeling individual decision-making in a dynamic environment. Also, it allows us to take into account the preference and priority of individuals that interact with a cognitive environment.

It is argued in Gärling et al (1993) that activity/travel decisions need to be treated in a single coherent conceptual framework operationalized as a CPM. Various efforts have been made to implement a conceptualization of travel decisions in a computer program mimicking people's
decision-making processes. Kuipers (1978) developed TOUR to model individual's memory
representation of the environment and its acquisition and use in wayfinding. The NAVIGATOR
model (Gopal et al. 1984) is based on empirical results reported in Golledge et al. (1985). Route
planning is modeled by TRAVELLER (Leiser and Zilberschatz, 1989) and ELMER (McCalls
et al., 1982). However, none of the models examine the dependencies between different travel
decisions and between travel decisions and activity choices (Gärling et al., 1993).

STARCHILD (Recker et al. 1986a, 1986b) proposed a psychologically more plausible
noncompensatory decision rule for selecting among the generated alternatives. However, the
notion that all feasible activity schedules are generated is unrealistic. Gärling et al. (1989) offered
a conceptual framework to perform activity scheduling. The selection of destination in the model
is based on the nearest neighbor heuristic. The application of the heuristic assumes people travel
via Euclidean distances. When applied in a IVHS context, in which route information can be
dynamically retrieved from the system, the interplay of the activity scheduling and route choice
would be an important area to look at. In order to implement this in reality and in the presence of
information, a time-dependent network and the GIS operations would be used.

Need for a GIS

Salient problems of the CPMs are the requirement of detailed data of the environment and the
interaction between the individual and this environment. Usually detailed travel surveys are
carried out to record individual travel patterns towards route, mode, and destination choice. It is
hoped that geographic information systems (GIS) can provide a comprehensive database and the
necessary analytical methods to handle these refined and disaggregate data (Replogle, 1989). GIS
operations can help to define individuals' spatial and temporal constraints of accessibility
(Golledge et al., 1993). However, how to form a GIS as a base for both the static and dynamic
environment needed for activity scheduling is still an unexplored area.

Most IVHS research focuses on the network system to provide congestion prediction and real-
time traffic information. Activity scheduling literature suggests that the decision of people to
participate in an activity affects network performance. In order to address this interrelationship, a
coherent working system is needed. GIS provides such a comprehensive database system to work
Most IVHS is based on simplified network elements (Vaughn et al., 1993). GIS, on the other hand, provides a realistic representation of the environment for modeling. Different information can be integrated through geo-referencing. GIS also provides a comprehensive database for both aggregate and disaggregate information. Also, a GIS is highly flexible in terms of manipulating spatial objects and distance according to rules, it also facilitates the representation of environment according to individual's behavioral characteristics. For network modeling, these rules include putting high impedance on a link to represent the lower probabilities of traveling on that link.

How to represent people's movement in time and space has been explored in Miller (1991). GIS operations can also help to model destination choice by defining a feasible opportunity set (Golledge et al. 1993). Research on complex travel behavior usually generates all the possibilities for activity scheduling by a combinatorial algorithm. It is computational demanding and inefficient. By defining such a set we can eliminate this problem.

Similarly, path selection by an individual can be benefited by defining a feasible route set. With the detailed network with spatial topology, a routing algorithm can be implemented to determine this feasible route for a route choice model.

When faced with changes in traffic condition under ATIS, a traveler would consider an alternative route. A CPM could model how an individual adapts to changes in the environment. In a GIS, information in the environment can be modified and spatial relationships recalculated. Different scenarios can also be simulated to test the “what-if” cases.

Proposed tasks

In this research, work will be done on GIS data model and database to provide a computer-simulated environment for micro-simulation in the IVHS context. The focus would be to examine how GIS provides a basis for dynamically integrating travel decision models. GIS is, however, not without problems. The proposed tasks to explore the GIS data model issues for integrating travel decision models are as follows

1) Activity scheduling and destination choice:

   Feasible opportunity set. since individuals will only consider certain places for an performing activity, a feasible opportunity set should be defined for searching. It will not just define a static set for an individual, but dynamically check the opportunity set when in
a particular location. It may result in changes in schedules. Rules for defining such a set would be from experiment to relating to different criteria. For example, buffer zones by travel time or distance can be manipulated and checked with real data or secondary sources. The interrelationship between the travel decisions of a household member for pretrip planning and other drivers on the road for en-route traveling, as well as the effects on defining the feasible opportunity set will be examined.

2) Routing elements:

Feasible route set: store different route and possible sets based on different criteria. Research to find out how people attempt to minimize total distance when they choose a path between several destinations (TSP) can be used as a guide for routing in a CPM (Hirtle and Gärling, 1992). Using the rules, which can then be written as heuristic algorithms, the actual and alternative path selection may then be modeled.

Develop a time-dependent network for routing. Research from Ziliaskopoulos et al., (1993) on a time-dependent shortest path algorithm, Ran et al. (1992) on travel time function network model, and Chen and Mahmassi (1993) on a time-dependent network to model real-time information processing will be explored. Travel time function will be formulated. Turn penalty on routing has not been widely incorporated into the routing literature in IVHS. However, in reality people may prefer not to make left-turns. This research explores the opportunity of incorporating turn penalty into the routing elements.

Behavioral assumptions Recognizing that it's people that travel. The behavioral characteristics are important for providing useful information for the traveler. Some may prefer freeway to local street while the others may have a reversed preference. The routing alternatives have to be provided accordingly.

For en-route traveler, travel information will be updated quite frequently. To take into account the effects of other travelers on overloading a particular route with diversion information from the ATIS, concurrency control has to maintain in the GIS database.

Data set

Travel diaries data will be used to validate the formation of feasible opportunity set and feasible route set. These data should consist of detailed information on the traveler's time of day, origin and
destination of each link, mode, duration, purpose, usage of freeway, and route information of travel. Socioeconomic data from this data set is also available. The California Telecommuting Pilot Project provide such information. Also, the Peugeot Sound Panel Data set record such data. Travel diaries from the Peugeot Sound Telecommuting Demonstration Project will also be useful. Network data will be from simulated data and the TIGER file from the Census Bureau. Other aggregate socioeconomic data would be from the Census to the most disaggregate level.

Conclusions:

The range of physical, environmental, behavioral, and decision making components required in the ATIS part of an IVHS lends itself to representation in a GIS. Although there are difficulties to be overcome in making such a system operational in real time, we feel the long run advantages of a GIS based ATIS will justify the needed basic research.

The principal author of this paper is pursuing this problem as a dissertation and will undoubtedly develop a menu of additional problems to be resolved by future research.

References


Section V

Path Selection and Route Preference in Human Navigation: A Progress Report

Paper prepared for COSIT Conference. Vienna, September 1995
Path Selection and Route Preference in Human Navigation: A Progress Report

by

Reginald G. Golledge
Department of Geography
and
Research Unit in Spatial Cognition and Choice
University of California Santa Barbara

Prepared for
COSIT Conference
Vienna
September 1995

Acknowledgments: Erika Ferguson, Graduate Student in Psychology; Amy Ruggles, Joanna Schulman, and John Dutton, Graduate Students in Geography, University of California Santa Barbara, for help in running experiments and preparing data for analysis.

This project was supported by NSF Grant #SES-9207836 and UCTC Grant DTRS 92-G-0009
INTRODUCTION

Not only do we select and follow a limited set of paths through the complex environments in which we live, but we have developed many models capable of finding solutions to these path selection problems (e.g., linear programming; traveling salesmen; shortest path). The question is, however, are these the criteria used by humans to solve their own movement problems - or are they methods best suited to mathematical or computer determination of optimal paths through complex multi-node networks to ensure economic efficiency of commercial or fleet traffic, but yet using criteria of which people in general are unaware, or are incapable of using? To explore this question, we examine the process of human navigation and report on pilot experiments that provide insights into the variety of Path Selection criteria used in different contexts.

BACKGROUND

Navigation seems to be one of the primary functions of vision in virtually all biological systems. The processes involved include cue or landmark recognition, turn angle estimation and reproduction, route link sequencing, network comprehension, frame of reference identification, route plotting strategies (e.g., dead reckoning, path integration, environmental simplification and en-route choice, shortcutting). These processes are used in encoding environmental information for internal processing and use in wayfinding situations. Because of human inaccuracies and errors in recognizing places and coding geometrical components of landscapes, history has seen the development of a variety of technical aids designed to substitute for these human frailties. For example, the prismatic compass was developed to provide greater accuracy than was possible by visually estimating direction. Distances were not measured accurately until the development of distance units and devices such as surveyors' chains, theodolites, range finders, and now ultrasonic laser beams. To find one's way efficiently through complex network structures, computer programs focusing on criteria such as shortest path, minimizing total distance or time traveled, or
maximal covering (Church & ReVelle, 1976) now replace the human interrogation of the network for destination choice and for optimal or feasible path selection in most transportation planning interactions where aggregate flows are allocated to routes. But what of the navigation and wayfinding activities of individuals? Do they conform to such principles?

Human navigation usually involves vision which in turn implies the use of inexact measurements and error prone or distorted cognitive maps. This is in contrast to the computerized algorithms for solving navigational problems that rely on explicit quantitative models and exact solution procedures. Some critical features of human navigation and wayfinding that have recently been highlighted are:

(1) The human navigation system interacts with and adapts to the environment in which it is navigating (Golledge, 1995).

(2) Navigation proceeds by initiating body motion and receiving and translating sensory feedback received from self perception of motion over time (Loomis, et al. 1992)

(3) The imagery developed by sensing the environment constrains the nature, type, speed and direction of motion (Golledge, 1992; Kitchin, 1994; Gärling, et al. 1984).

(4) Potential routes are imaged as larger or shorter depending on whether they proceed towards or away from a primary node or reference point (Sadalla, Burroughs, & Staplin, 1980).

(5) Many route-distances are imaged as being non-symmetric (Montello, 1992).

Thus, human navigation is often conceived of as a suboptimal system, as compared to vehicle navigation which is often considered as optimized movement in a precisely specified networked environment.

**RESEARCH QUESTIONS**

We wished to examine questions about: (a) how characteristics of the global stimulus environment affected route choices overall, (b) how the differences between pairs of points affected route choice
within a given environment; and (c) how varying network properties influenced path selection criteria.

Questions investigated included the following:

- Do people try to retrace routes when the task involves using more than a single origin or destination?
- How consistent are people in terms of their criteria for route selection as the environment changes (e.g., from simple grid to grid with curves or grid with diagonals)?
- How often do people retrace the same route when traversing between origins and destinations?
- How often is the same criteria chosen when traveling routes of different complexity?
- What criteria do people usually think they use when they are performing route selection tasks in the laboratory and in the field?
- What criteria do people feel they use most frequently when choosing routes in their normal everyday movements through real world environments?

HYPOTHESES

Specific hypotheses to be examined were:

(a) The dominant route selection criteria will change as the environment changes.

(b) The dominant route selection criteria will change as trip complexity changes from a single origin-destination pairing to a multiple stop trip.

(c) As the number of potential “stops” increase in a trip chain, the probability of retracing a route will decrease.

(d) Traditionally accepted criteria such as shortest path or least time will dominate as route selection criteria.

(e) Route selection criteria will not change as orientational perspectives change.
Route selection criteria will not differ between map base or laboratory conditions and real world route following conditions.

Experiment #1.

The Laboratory Tasks: Route Selection from Maps

In this project we study the kinds of routes that people select when navigating through a given environment. Experiments were undertaken in both laboratory and field situations to observe routes taken and then inferences are made about the criteria that was used. Initially, subjects were given a series of maps on which two locations were marked. These maps consisted of simple rectangular grids. Three different routes were laid out from a common origin to a common destination. Subjects were asked to imagine that they lived in a town built around the grid network shown on each map, and to imagine that moving from the origin to the destination represented a daily home-work or work-home activity. They were asked to decide which of three routes they would take. The routes allowed them the choice of taking the longest leg first (C), the shortest leg first (A), or a stepwise route that approximated a diagonal join between origin and destination (simulating most direct, least effort or least time (B)). Given the regularity of the grid, however, each route was exactly the same distance and varied only in its configurational properties. Maps and routes were configured so that trips were undertaken either as one travels from South to North in conventional coordinate terms or from North to South (Figure 1). Different configurations of O-D paths were provided while actual distances were kept constant. When choosing a route, subjects were required to place or hold the maps horizontally with the northern edge being furthest from the body. No rotation or translation of a given map (or subject) was permitted. However, by rotating a map 90° in either a clockwise or counter-clockwise direction and labelling the furthest edge as north, the same geometric configuration can be maintained while orientation and perceptive changes. This procedure was followed for all map types.

A second task involved route selection after the number of nodes to be visited en-route was increased (i.e. trip chaining). Again, routes were configured so that travel took place either from South to North or North to South. In this task the environment was changed from a regular grid to
Figure 1

Map Grid - A in $N^\text{th}$ - 2 Locations Step Diagonal

A = Shortest leg first
B = Diagonal
C = Longest leg first
A third task involved changing the regular grid to include curved roads and nonorthogonal and intermittent intersection blockages. Polygons representing either negative or positive externalities (e.g., waste dumps or parks) were interspersed throughout the maps. Blockages were described on different trials as parks (a positive attractor) or waste dumps (a negative attractor) (Figure 3). The same route choice task was repeated controlling directional components and total length of trip. This task again increased the number of places to be visited to see if criteria were used that differed from simple barrier-free origin-destination selection. After each map trial was completed, individual suggestions were solicited regarding what route choice criteria were perceived as being used on these tasks, and what criteria the subject “usually” used in daily real world interactions. Such variables were examined to isolate the type of reasoning or inference that underlies path selection.

**SUBJECTS:**

Subjects consisted of 32 adults, 16 women and 16 men. Most were students. Ages ranged from 20-35 years of age. Approximately 50% were geographically trained.

**DATA COLLECTION**

The individual data was compiled on packets of maps in the following manner:

(a) Six stimulus groups were formed by crossing the three environments, grid, diagonal, and curved, with the two orientations “A in the Nth” and “A in the Sth.” A in the Sth was a 90° rotation clockwise or counter-clockwise from A in the Nth. A separate map packet was used for each rotation.

(b) Within each stimulus group for each unique route drawn by subjects between each pair of points (i.e. on each page of the packet) a line of a different color was drawn on
Figure 2
Map Grid - A in $S_\text{th}$ - with Diagonals
Figure 3

Map Grid - A in S\textsuperscript{th} Aesthetics
the compiled map to represent the route taken. The number of subjects in the group who had drawn each of the consequently defined unique routes was tallied at the bottom of each page.

(c) Possible routes were classified into cadres such as shortest distance, fewest turns, longest leg first, shortest leg first, most aesthetic, many curved roads, least time, first route noticed, most turns, and “different from a way I had already gone.”

(d) Matches were made between lines drawn on each map and one of the selection criteria mentioned above. Examples of other possible selection criteria (e.g., gradients) were not possible to determine on these grids.

(e) Data from all six compiled packets were entered into an EXCEL spreadsheet to examine each of the following alternative route selection criteria.

Results of matching these route types with routes actually chosen by subjects (i.e., percentage time each route was chosen) were tabulated (Table 1).

(i) Fewest Turns

For each environment, the total number of people who chose a route with the fewest possible turns between each pair of points was recorded. If there was more than one unique route on the compiled map that had the fewest turns possible, then all such numbers were aggregated and the number of people using all such routes was recorded. The actual number of turns that defines “the fewest” for each pair of points was also recorded. The proportion of people in the particular stimulus group who chose a route with the fewest turns was calculated.

(ii) Longest Leg First

This spreadsheet was prepared in a manner similar to Fewest Turns. Here the total number of people who chose a route in which the longest leg of their chosen route was the first segment of the route was first recorded. “Longest” was defined in terms of total distance (not number of blocks). If no one chose a route in which the longest leg was first, then the number of people entered was zero. The number of legs of each route was also recorded.
Table 1

Ranking of Criteria Most Often Used in Route Selection

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest Distance</td>
<td>1</td>
</tr>
<tr>
<td>Least Time</td>
<td>2</td>
</tr>
<tr>
<td>Fewest Turns</td>
<td>3</td>
</tr>
<tr>
<td>Most Scenic/Aesthetic</td>
<td>4</td>
</tr>
<tr>
<td>First Noticed</td>
<td>5</td>
</tr>
<tr>
<td>Longest Leg First</td>
<td>6</td>
</tr>
<tr>
<td>Many Curves</td>
<td>7</td>
</tr>
<tr>
<td>Many Turns</td>
<td>8</td>
</tr>
<tr>
<td>Different from Previous</td>
<td>9</td>
</tr>
<tr>
<td>Shortest Leg First</td>
<td>10</td>
</tr>
</tbody>
</table>
(iii) **Preference for Curves**

The question here is whether people have a preference for routes involving curves. For each pair of points, the number of people who indicated routes including at least one curved portion were averaged. Each unique route was recorded. The overall preference for curves was quite high. There was quite a bit of variation between routes. However, this measure does not take into account how many curved routes were possible between each pair of points.

(iv) **Preference for Diagonals**

This was similar to the Preference for curves spreadsheet. Again, the overall preference for taking a diagonal was quite high.

(v) **Shortest Route**

For the diagonal and curve maps, actual distance was measured to determine the true shortest routes. For the regular Grid maps, since all routes that remain within the boundaries of the two points are necessarily of equal length, the question was whether subjects chose a route that would seem to minimize Euclidean distance by traveling "through the middle".

(vi) **Most Aesthetic**

This criteria could only be used with the final set of maps in which polygons representing parks and waste dumps were included. Routes heading away from waste dumps and/or following an edge of a park were labeled most aesthetic.

(vii) **Other Criteria**

Other criteria were defined in similar ways by observing characteristics of the chosen route and inferring what might have prompted its selection.

Detailed results of this study are published elsewhere (Golledge, 1995) but some of the more interesting results are reviewed here as being pertinent to several hypotheses.

Let us now turn to a detailed discussion of selected criteria and examine consistency of selection in different environments and from different perspectives.

**Route Selection Criteria**

Fewest Turns. It is apparent that as the environment changes, so does the popularity of this
criteria, dropping from a high of 67% in a simple regular grid environment to 25% in a curvilinear environment (Table 2). Data is reported for each of three environments (Grid, Diagonal, Curves).

A second table illustrates changes in criteria selection when perspective changes, i.e. when travel is from a distant origin or to a distant destination (Table 3). In the case where perspectives differ, there is a remarkable difference in choice of this strategy when the path to be traveled heads from Sth to Nth (65%) as opposed to heading from Nth to Sth (7%). A significant difference occurs in the diagonal environment also, but not in the curvilinear one.

O-D with Intervening Points (Trip Chaining)

Turning now to a slightly more complicated situation in which an intervening point was included on the trip (e.g., from homebase A to intermediate point E to destination point C) we find substantial differences in path selection criteria in each type of environments. Focusing still on the fewest turns criteria, for the simple orthogonal grid map where the origin was in the Nth, 46% used the fewest turns as a strategy but only 38% used it on the inward trip (Table 4). For the map with diagonals, 9% and 4.5% used fewest turns when A was in the Nth and Sth respectively; for the map with curves, 12% used it when A was in the Nth, while 21% did so when A was in the Sth. Similarly, variable results were obtained for all the different criteria selected.

Shortest Path Because of the way the simple regular grid was configured, all routes were of equal distance. Shortest path criteria thus could only be examined in the grid with diagonals, and grid with curves cases. This criterion is the one generally accepted as dominant in most network flow or routing models. It makes sense that it should be so if one is trying to maximize economic utility or minimize costs or time expended in travel. In these experiments however, we again found inconsistencies in criterion use. For example, in the diagonals case, with a single O-D path, 58% used the strategy, while 84.5% used it in the trip chaining cases. Sixty-eight percent used the strategy when the origin was in the Nth, while 80% used it when the origin was in the Sth. For the environment with curves, 74% used it when A was in the Nth, while 90% used it when the origin was in the Sth. Eighty percent adopted it in the trip chaining case, but 54% used it for single O-D pairings (Table 5).
Table 2
Fewest Turns

<table>
<thead>
<tr>
<th>Route</th>
<th>Grid Maps</th>
<th>Grid with Diagonals</th>
<th>Route with Curves</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% Using Criteria When A in Nth</td>
<td>% Using Criteria When A in Sth</td>
<td>% Using Criteria When A in Nth</td>
</tr>
<tr>
<td>A-B</td>
<td>0.83</td>
<td>0.71</td>
<td>0.47</td>
</tr>
<tr>
<td>B-A</td>
<td>0.78</td>
<td>0.71</td>
<td>0.47</td>
</tr>
<tr>
<td>A-C</td>
<td>0.89</td>
<td>0.66</td>
<td>0.41</td>
</tr>
<tr>
<td>C-A</td>
<td>0.94</td>
<td>0.66</td>
<td>0.35</td>
</tr>
<tr>
<td>A-D</td>
<td>0.67</td>
<td>0.79</td>
<td>0.94</td>
</tr>
<tr>
<td>D-A</td>
<td>0.83</td>
<td>0.71</td>
<td>1.00</td>
</tr>
<tr>
<td>A-E</td>
<td>0.72</td>
<td>0.79</td>
<td>0.12</td>
</tr>
<tr>
<td>E-A</td>
<td>0.78</td>
<td>0.64</td>
<td>0.12</td>
</tr>
<tr>
<td>A-F</td>
<td>0.72</td>
<td>0.57</td>
<td>1.00</td>
</tr>
<tr>
<td>F-A</td>
<td>0.89</td>
<td>0.57</td>
<td>0.88</td>
</tr>
<tr>
<td>B-D</td>
<td>0.94</td>
<td>0.71</td>
<td>0.47</td>
</tr>
<tr>
<td>D-B</td>
<td>0.83</td>
<td>0.79</td>
<td>0.12</td>
</tr>
<tr>
<td>B-F</td>
<td>0.78</td>
<td>0.71</td>
<td>0.18</td>
</tr>
<tr>
<td>F-B</td>
<td>0.78</td>
<td>0.43</td>
<td>0.06</td>
</tr>
<tr>
<td>C-D</td>
<td>0.78</td>
<td>0.71</td>
<td>0.24</td>
</tr>
<tr>
<td>D-C</td>
<td>0.83</td>
<td>0.79</td>
<td>0.41</td>
</tr>
<tr>
<td>C-F</td>
<td>0.67</td>
<td>0.79</td>
<td>0.06</td>
</tr>
<tr>
<td>F-C</td>
<td>0.56</td>
<td>0.64</td>
<td>0.00</td>
</tr>
<tr>
<td>D-E</td>
<td>0.94</td>
<td>0.93</td>
<td>0.41</td>
</tr>
<tr>
<td>E-D</td>
<td>1.00</td>
<td>0.86</td>
<td>0.53</td>
</tr>
<tr>
<td>A-E-C</td>
<td>0.28</td>
<td>0.36</td>
<td>0.06</td>
</tr>
<tr>
<td>C-E-A</td>
<td>0.61</td>
<td>0.64</td>
<td>0.24</td>
</tr>
<tr>
<td>A-F-B</td>
<td>0.50</td>
<td>0.21</td>
<td>0.00</td>
</tr>
<tr>
<td>B-F-A</td>
<td>0.67</td>
<td>0.29</td>
<td>0.12</td>
</tr>
<tr>
<td>A-F-C</td>
<td>0.17</td>
<td>0.36</td>
<td>0.00</td>
</tr>
<tr>
<td>C-F-A</td>
<td>0.56</td>
<td>0.43</td>
<td>0.12</td>
</tr>
<tr>
<td>ALL ROUTES</td>
<td>0.73</td>
<td>0.65</td>
<td>0.32</td>
</tr>
</tbody>
</table>
Table 3

Fewest Turns: Criterion Selection in Each Environment

<table>
<thead>
<tr>
<th>Environment</th>
<th>% Subjects Choosing This Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grd</td>
<td>67%</td>
</tr>
<tr>
<td>Curves</td>
<td>25%</td>
</tr>
<tr>
<td>Diagonal</td>
<td>57%</td>
</tr>
</tbody>
</table>

Source: Golledge, Experimental Data
Table 4

Fewest Turns: Perspective Change

<table>
<thead>
<tr>
<th>Environment</th>
<th>A in N&lt;sup&gt;th&lt;/sup&gt;</th>
<th>A in S&lt;sup&gt;th&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gnd</td>
<td>7%</td>
<td>65%</td>
</tr>
<tr>
<td>Curves</td>
<td>56%</td>
<td>58%</td>
</tr>
<tr>
<td>Diagonal</td>
<td>32%</td>
<td>18%</td>
</tr>
</tbody>
</table>

Source: Golledge, Experimental Data
## Table 5

Shortest Path

<table>
<thead>
<tr>
<th>Route</th>
<th>Diagonals</th>
<th>Curves</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% Using Criteria When A in Nth</td>
<td>% Using Criteria When A in Sth</td>
<td>% Using Criteria When A in Nth</td>
<td>% Using Criteria When A in Sth</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-B</td>
<td>0.53</td>
<td>0.60</td>
<td>0.65</td>
<td>0.80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-A</td>
<td>0.71</td>
<td>0.80</td>
<td>0.76</td>
<td>0.80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-C</td>
<td>0.59</td>
<td>0.67</td>
<td>0.53</td>
<td>0.60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-A</td>
<td>0.65</td>
<td>0.73</td>
<td>0.59</td>
<td>0.80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-D</td>
<td>0.00</td>
<td>0.00</td>
<td>0.47</td>
<td>0.60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-A</td>
<td>0.00</td>
<td>0.00</td>
<td>0.59</td>
<td>0.60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-E</td>
<td>0.88</td>
<td>0.93</td>
<td>0.35</td>
<td>0.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E-A</td>
<td>0.88</td>
<td>0.87</td>
<td>0.24</td>
<td>0.33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-F</td>
<td>0.06</td>
<td>0.00</td>
<td>0.59</td>
<td>0.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-A</td>
<td>0.00</td>
<td>0.07</td>
<td>0.82</td>
<td>0.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-D</td>
<td>0.53</td>
<td>0.93</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-B</td>
<td>0.88</td>
<td>0.93</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-F</td>
<td>0.82</td>
<td>1.00</td>
<td>1.00</td>
<td>0.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-B</td>
<td>0.94</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-D</td>
<td>0.76</td>
<td>0.80</td>
<td>0.76</td>
<td>0.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-C</td>
<td>0.53</td>
<td>0.93</td>
<td>0.82</td>
<td>#VALUE!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-F</td>
<td>0.82</td>
<td>0.93</td>
<td>0.71</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-C</td>
<td>0.94</td>
<td>0.93</td>
<td>0.53</td>
<td>0.93</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-E</td>
<td>0.59</td>
<td>0.73</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E-D</td>
<td>0.47</td>
<td>0.73</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-E-C</td>
<td>0.88</td>
<td>0.80</td>
<td>0.65</td>
<td>0.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-E-A</td>
<td>0.65</td>
<td>0.87</td>
<td>0.35</td>
<td>0.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-F-B</td>
<td>1.00</td>
<td>0.87</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B-F-A</td>
<td>0.88</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-F-C</td>
<td>0.94</td>
<td>0.93</td>
<td>0.88</td>
<td>0.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-F-A</td>
<td>0.82</td>
<td>1.00</td>
<td>0.82</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>0.68</td>
<td>0.80</td>
<td>0.74</td>
<td>0.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Route Retraces

Now let us consider situations where individuals were required to travel between A and B in each direction. Here we are concerned with the question of whether the same route was retraced, and if so, what this did to the route selection criterion. As an example, results are presented for the "longest leg first" criterion.

First in the simple grid environment, route retrace was not usually followed. For example, 44% subjects chose longest leg first when traveling from A to B when A was located in the Nth. However, 61% chose this strategy on the return route. This means the return route could not have been a retrace of the original! More confusion occurs when we change perspectives and pursue a path when A is in the Sth to a northerly located B. Here, only 29% used this criterion. In the reverse task, however, 64% chose the strategy!

On the map with curves, 35% chose this strategy when traveling from a distant origin to a close destination, but only 12% chose the strategy on the retrace task. When the origin was close and the destination distant, 13% chose it on the outbound journey and zero chose it on the retrace. When diagonals were included, a similar outbound and retrace pattern occurred, but with a close origin, differences again fluctuated widely from 7% to 20%.

When considering trip chaining, differences in criteria selection become marked depending on orientation. In a simple grid, 33% chose longest leg first when traveling from a distant origin towards a close destination, but zero percent did this on the return trip. When traveling from a close origin to a distant destination, 14% chose the strategy, but zero percent chose it when traveling the reverse route.

On the map which included some diagonals and again required traveling through an intermediate point, when the origin was distant, 35% used longest leg first, but on the return trip zero percent used that strategy. When the origin was in the Sth, 33% used longest leg first and again on the return trip zero percent used it. In the curvilinear condition 15% chose the strategy when A was distant while zero selected it on the return. It might be suggested that in these cases, a pure retrace strategy may have been used, thus precluding any "longest leg first" strategies from
being implemented. Visual examination of subjects’ maps tends to confirm this explanation. The occurrence of zero percent choice on the return trip does indicate that exact route retracing was a possible option as a route selection strategy.

Preference for Curved and Diagonal Routes

The question examined next was whether people have a preference for routes involving curves. For each pair of points the number of people who indicated routes including at least one curved portion were averaged. Each unique route was recorded. Preference for curves was quite high (74% chose a route with curves in routes with A in the Nth and 90% chose a route with curves in routes with A in the Sth). There was some variation between routes. However, this measure does not take into account how many curved routes were possible between each pair of points; data is only for routes actually chosen by subjects.

Preference for diagonals proved to be similar to the preference for curves results. Again, the overall preference for taking a diagonal was quite high (68% chose a route using at least one diagonal when traveling Nth to Sth, 80% chose a route with at least one diagonal when moving from Sth to Nth).

DISCUSSION

Perception of Criteria:

As part of the general information collected from our subjects I asked them to rate on a five point scale with values ranging from “quite unimportant” to “extremely important”), what criteria they thought they used when performing the route selection task both in the laboratory and in general practice. The alternatives given were those inferred from the maps they had compiled. The responses indicated that shortest distance was given the highest rating across the sample group (mean score of 4.2) with shortest time close behind (mean is 4.1) (Table 6). Fewest turns was rated 3.6 and the most scenic or most aesthetic route received 3.5. This table shows there is then a noticeable drop to the remaining criteria.

When asked what criteria they “normally” used when selecting routes in their real world
Table 6
Mean Ratings of Criteria Used in Single O-D Route Choice Task

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Mean Rating of Criteria Used in Task</th>
<th>Mean Rating of Criterion &quot;Usually Chosen&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest Distance</td>
<td>4.2</td>
<td>4.4</td>
</tr>
<tr>
<td>Least &quot;Time</td>
<td>4.1</td>
<td>2.6</td>
</tr>
<tr>
<td>Fewest Turns</td>
<td>3.6</td>
<td>3.5</td>
</tr>
<tr>
<td>Most Scenic/Aesthetic</td>
<td>3.5</td>
<td>1.9</td>
</tr>
<tr>
<td>First Noticed</td>
<td>2.5</td>
<td>4.3</td>
</tr>
<tr>
<td>Longest Leg First</td>
<td>2.3</td>
<td>2.3</td>
</tr>
<tr>
<td>Many Curves</td>
<td>2.3</td>
<td>1.6</td>
</tr>
<tr>
<td>Most Turns</td>
<td>1.8</td>
<td>2.7</td>
</tr>
<tr>
<td>Different from Previous</td>
<td>1.8</td>
<td>2.1</td>
</tr>
<tr>
<td>Shortest Leg First</td>
<td>1.7</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Source: Golledge, Experimental Data
Ratings were scored on a 5-point scale
activity patterns, shortest distance again received the highest rating (4.4) but the “first experienced” or noticed route was rated second (4.3). This was invariably a route heading in the “general direction” of the destination. Routes with the fewest turns (3.5) and routes with the shortest leg first (3.4) followed in importance. Obviously the map route selection task was perceived as being something different to what would normally be experienced in real world interaction patterns. What is interesting, however, is the lack of relative significance given to variables which are often perceived to be “popular” such as minimizing time (2.6) and scenic/aesthetic routes (1.9). The significance of the first route experienced or chosen between an origin or destination is quite noticeable and supports suggestions made by Golledge & Zannaras (1973) that when choosing routes people are likely to limit experimentation and quickly develop a firm preference for a route to be followed on a regular basis after a small number of trials, regardless of its economic, temporal or spatial optimality, usually the first leg of this route heads in the general sectoral direction of the destination.

Although there have been questions raised regarding the suitability of using maps in wayfinding tasks (Lloyd & Cammack, 1995), this set of exercises provides evidence that human path selection may not be the simple process that is usually assumed in network flow solution algorithms. While shortest path and least time were most highly ranked, it was also obvious that as one changed the complexity of the environment, and as trip making became more complex because of chaining several nodes together, path selection criteria changed. Also, there was no clear evidence that trip retracing was carried out except in some complex environments where chaining was required. Thus, it seems that some accounting for well known behaviors such as taking different routes to and from a given destination, or perceiving that routes in some direction are more acceptable than those in different directions (i.e., that there is an orientation bias in selecting routes) that can be partly accounted for by changing route selection criteria.

Given these laboratory based results, we now turn to a field experiment to see if they are duplicated or whether the experimental situation produced “artificial” behaviors.
Experiment #2:

Path Selection in a Real Environment:

A second study was consequently undertaken to examine path selection criteria in a real world rather than laboratory setting.

Using information derived from the laboratory experiment, possible routes between two pairs of origins and destinations on a Western United States campus were used. Subjects were all familiar with the study area and were asked to select routes in both forward and reverse directions between the chosen points. Paths conforming to the criteria types identified in the laboratory experiment were defined and matched against the routes actually selected by subjects. Research questions again focused on inferring which criteria were used in path selection, whether route retraces were used, and what criteria were used most frequently. Only single O-D pairs were used; no trip chaining was investigated.

The principal hypotheses were similar to those examined in the map experiment. It was hypothesized that: (i) shortest distance and shortest time would be the two primary criteria; (ii) route retraces would occur frequently on both routes; and (iii) people will use the same criteria in this real world experiment that they use in everyday activities.

Subjects/Environment

The study was conducted on the campus of a Western United States university in the area between Ellison and Cheadle Hall (see Figure 4). The environment consists of a central open courtyard containing large regularly spaced planters. The courtyard adjoins Ellison Hall in an area divided by pathways and grassy areas. Two routes were selected for the study. The Stairs Route (A-B) consisted of the origin/destination pair of the flagpole at the north east corner of Cheadle Hall and the stairway door at the west end of the north wing of Ellison Hall. The Elevator Route (X-Y) consisted of the origin/destination pair of the flagpole at the north east corner of Cheadle and the elevator entrance at the east end of the north wing of Ellison. Each of these round trip routes was subdivided into forward and reverse components resulting in four route conditions:

- **forward stairs:** here the subjects’ first task began at the flagpole, traveled to the
stairs and returned; his or her second task began at the flagpole and traveled to the elevator and returned.

- **reverse stairs**: here the subjects' first task began at the stairs, traveled to the flagpole and returned; his or her second task began at the elevator and traveled to the flagpole and returned.

- **forward elevator**: here the subjects' first task began at the flagpole and traveled to the elevator and returned; his or her second task began at the flagpole and traveled to the stairs and returned.

- **reverse elevator**: here the subjects' first task began at the elevator and traveled to the flagpole and returned; his or her second task began at the stairs and traveled to the flagpole and returned.

All subjects were university staff or students (both graduate and undergraduate). An equal number of men and women, and geography, non-geography students were selected. Subjects were chosen by convenience from responses to fliers advertising the study.

**METHODS/PROCEDURE**

Thirty-two subjects were scheduled for the experiment during daytime hours. All subjects were very familiar with the study area. Subjects were randomly assigned to the four different conditions while ensuring that equal numbers of male and female, and geography non-geography students were placed in each condition. A sample of the subject assignment strategy of each route for each group is given in Figure 5.

Subjects were taken to the origin for their assigned route condition and then were read the appropriate directions. They then began to walk a route of their choice to the assigned destination. This route and the time taken to travel it was recorded by the researcher on a map of the area. This procedure was repeated for the reverse section of the route. Subjects then completed a questionnaire on the criteria they used in selecting their route and normal activity behavior, plus evaluations of self confidence in spatial tasks and normal modes of travel.
Figure 5

Decision Trace of Paths

Route #1, 1st

Flag → Elev. (n=8)

Elev. → Flag (n=8)

(n=16)

Route #2, 2nd

Flag → Stair (n=8)

Stair → Flag (n=8)

(n=16)

Route #2, 1st

Flag → Stair (n=8)

Stair → Flag (n=8)

(n=16)

Route #1, 2nd

Flag → Elev. (n=8)

Elev. → Stair (n=8)
The average group response for rating route choice criteria usually used and perceived to be used in this field experiment are presented in Table 7. According to questionnaire responses, subjects rated shortest route, route taking the least time, and route proceeding in the direction of destination as being the most important. Criteria of fewest turns, first noticed, and “usual route” were next in importance. In general the criteria values are consistent between those used on the task and those commonly used.

To analyze the route choice behavior based on traveling in the environment, all routes used between origin and destination pairs were determined and coded. Figures 6 and 7 show routes chosen between A and B, and X and Y. All possible routes were coded by identifying segments and choice points, and a separate route code was provided for each possible route that could be taken on each task. The number of times a given route was taken was recorded. The maps produced by recording the routes subjects traveled during the experiment were then used to produce Table 8 which shows the route chosen, time taken to complete, whether the same route was taken in the forward and reverse directions and which direction was traveled more quickly for each subject.

For the Flagpole to Stairway route, 62.5% of the subjects traveled the same route in both directions. For the Flagpole to Elevator route, 15.6% of the subjects traveled the same route in both directions. This is a significant difference in route retrace between the two origin/destination pairs. This is apparently due to the existence of some route choice criteria present in this environment that produces a distinctly different route choice decision to be made depending on the direction of travel. Of particular importance is the layout of features near the elevator at Ellison Hall, including the presence of a central grassy area dividing travel into one of two paths. While traveling from the elevator to the flagpole 75% of the subjects chose a route that took them to the north of the grassy area that is encountered when leaving Ellison for Cheadle Hall. While traveling from the flagpole to Ellison Hall 75% of the subjects chose a route that took them south of this same grassy area. (i.e., route choice was dependent on direction of travel). One interpretation of this route choice behavior is that subjects chose a route that took them away from Ellison Hall as
Table 7

Mean Ratings of Criteria Used in Single O-D Route Choice Task

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Used in Task</th>
<th>Usually Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest Path</td>
<td>4.09</td>
<td>4.13</td>
</tr>
<tr>
<td>Least Time</td>
<td>4.03</td>
<td>3.97</td>
</tr>
<tr>
<td>Direction of Travel</td>
<td>3.97</td>
<td>4.13</td>
</tr>
<tr>
<td>Fewest Turns</td>
<td>3.53</td>
<td>3.53</td>
</tr>
<tr>
<td>Usual Route</td>
<td>3.41</td>
<td>3.50</td>
</tr>
<tr>
<td>First Noticed</td>
<td>3.34</td>
<td>3.47</td>
</tr>
<tr>
<td>Most Aesthetic</td>
<td>2.84</td>
<td>2.80</td>
</tr>
<tr>
<td>Longest Leg First</td>
<td>2.06</td>
<td>2.00</td>
</tr>
<tr>
<td>Shortest Leg First</td>
<td>1.84</td>
<td>1.81</td>
</tr>
<tr>
<td>Many Curves</td>
<td>1.66</td>
<td>2.16</td>
</tr>
<tr>
<td>Alternate Route</td>
<td>1.66</td>
<td>1.84</td>
</tr>
<tr>
<td>Most Turns</td>
<td>1.44</td>
<td>1.66</td>
</tr>
</tbody>
</table>

Source: Golledge, Experimental Data
<table>
<thead>
<tr>
<th>Code</th>
<th>Geo/Non</th>
<th>MF</th>
<th>A-B</th>
<th>B-A</th>
<th>X-Y</th>
<th>Y-X</th>
<th>A to B Route Chosen</th>
<th>Time (mins)</th>
<th>B to A Route Chosen</th>
<th>Time (mins)</th>
<th>X to Y Route Chosen</th>
<th>Time (mins)</th>
<th>Y to X Route Chosen</th>
<th>Time (mins)</th>
<th>Route Same?</th>
<th>Fastest Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>Geo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.19</td>
<td>4</td>
<td>1.18</td>
<td>25</td>
<td>1.33</td>
<td>22</td>
<td>1.34</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.10</td>
<td>4</td>
<td>1.04</td>
<td>22</td>
<td>1.18</td>
<td>21</td>
<td>1.20</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.15</td>
<td>4</td>
<td>1.15</td>
<td>23</td>
<td>1.26</td>
<td>22</td>
<td>1.28</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.15</td>
<td>4</td>
<td>1.13</td>
<td>26</td>
<td>1.27</td>
<td>27</td>
<td>1.28</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.23</td>
<td>4</td>
<td>1.22</td>
<td>23</td>
<td>1.38</td>
<td>23</td>
<td>1.39</td>
<td>1</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.16</td>
<td>4</td>
<td>1.21</td>
<td>22</td>
<td>1.30</td>
<td>22</td>
<td>1.30</td>
<td>1</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.30</td>
<td>5</td>
<td>1.32</td>
<td>23</td>
<td>1.49</td>
<td>22</td>
<td>1.50</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.24</td>
<td>3</td>
<td>1.30</td>
<td>23</td>
<td>1.34</td>
<td>22</td>
<td>1.37</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.11</td>
<td>4</td>
<td>1.11</td>
<td>26</td>
<td>1.24</td>
<td>22</td>
<td>1.37</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>1.36</td>
<td>1</td>
<td>1.27</td>
<td>23</td>
<td>1.39</td>
<td>24</td>
<td>1.49</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.15</td>
<td>4</td>
<td>1.17</td>
<td>23</td>
<td>1.29</td>
<td>25</td>
<td>1.30</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>1.33</td>
<td>4</td>
<td>1.30</td>
<td>25</td>
<td>1.42</td>
<td>27</td>
<td>1.43</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.12</td>
<td>4</td>
<td>1.14</td>
<td>25</td>
<td>1.27</td>
<td>22</td>
<td>1.27</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>1.40</td>
<td>7</td>
<td>1.45</td>
<td>23</td>
<td>1.50</td>
<td>21</td>
<td>1.51</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.19</td>
<td>1</td>
<td>1.15</td>
<td>23</td>
<td>1.25</td>
<td>21</td>
<td>1.27</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>1.31</td>
<td>9</td>
<td>1.11</td>
<td>23</td>
<td>1.30</td>
<td>28</td>
<td>1.50</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.20</td>
<td>4</td>
<td>1.19</td>
<td>23</td>
<td>1.44</td>
<td>22</td>
<td>1.48</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.17</td>
<td>4</td>
<td>1.16</td>
<td>23</td>
<td>1.31</td>
<td>22</td>
<td>1.31</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1.30</td>
<td>2</td>
<td>1.33</td>
<td>23</td>
<td>1.39</td>
<td>22</td>
<td>1.41</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.25</td>
<td>4</td>
<td>1.23</td>
<td>25</td>
<td>1.43</td>
<td>27</td>
<td>1.41</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.25</td>
<td>1</td>
<td>1.23</td>
<td>23</td>
<td>1.34</td>
<td>21</td>
<td>1.34</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1.26</td>
<td>11</td>
<td>1.35</td>
<td>29</td>
<td>1.52</td>
<td>22</td>
<td>1.45</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>1.22</td>
<td>4</td>
<td>1.12</td>
<td>23</td>
<td>1.33</td>
<td>22</td>
<td>1.27</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.03</td>
<td>4</td>
<td>1.04</td>
<td>25</td>
<td>1.19</td>
<td>25</td>
<td>1.22</td>
<td>1</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.15</td>
<td>12</td>
<td>1.21</td>
<td>23</td>
<td>1.32</td>
<td>27</td>
<td>1.33</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.18</td>
<td>4</td>
<td>1.19</td>
<td>21</td>
<td>1.45</td>
<td>21</td>
<td>1.39</td>
<td>1</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.29</td>
<td>2</td>
<td>1.28</td>
<td>21</td>
<td>1.48</td>
<td>22</td>
<td>1.42</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>Geo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.28</td>
<td>1</td>
<td>1.33</td>
<td>23</td>
<td>1.55</td>
<td>23</td>
<td>1.49</td>
<td>1</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.25</td>
<td>4</td>
<td>1.22</td>
<td>23</td>
<td>1.41</td>
<td>22</td>
<td>1.38</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.14</td>
<td>4</td>
<td>1.11</td>
<td>23</td>
<td>1.24</td>
<td>22</td>
<td>1.27</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1.29</td>
<td>4</td>
<td>1.26</td>
<td>21</td>
<td>1.42</td>
<td>25</td>
<td>1.45</td>
<td>0</td>
</tr>
<tr>
<td>✓</td>
<td>NonGeo</td>
<td>F</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.31</td>
<td>4</td>
<td>1.33</td>
<td>23</td>
<td>1.34</td>
<td>25</td>
<td>1.45</td>
<td>0</td>
</tr>
</tbody>
</table>

* 0 = different route
1 = same route

** 0 = neither faster
1 = first traveled faster
2 = second traveled faster
soon as possible when leaving the elevator and took them close to Ellison Hall as quickly as possible when approaching their destination. In this interpretation one could hypothesize that the building represented the destination on a larger scale of route planning and that leaving Ellison Hall represents leaving the elevator and conversely reaching Ellison Hall represents reaching the elevator.

The two routes further produced interesting differences when retracing is considered. On route A-B (flagpole to stairs), 62.9% took the same route in both a forward and reverse direction. On route X-Y (flagpole to elevator), only 15.6% took the same route.

For both routes 43.7% of the subjects traveled the first direction faster than the return direction. For the stairway route 46.9% of the subjects completed the return portion of the route faster than the first. For the elevator route, 43.7% of the subject completed the reverse segment faster than the first traveled. This doesn’t support the intuitive position that subjects would travel the return route faster after having learned the route and environment on the first leg.

Table 9 provides information supporting a more thorough study of the differences between the two origin/destination pairs (in addition to the differences in route retrace frequency). Between the flagpole and stairs, two routes (#16 and 9) accounted for 75% of subject’s route choices regardless of direction traveled. Between the flagpole and the elevator five routes (#23, 25, 22, 24, and 26) accounted for 75% of subject’s route choices regardless of direction traveled. However, a total of twelve routes were needed to account for all travel between the flagpole and stairs while only nine routes were needed to account for all travel between the flagpole and the elevator. It is interesting to consider this data in light of the differences in the spatial layout of the two route areas. The stairway route is primarily across a plaza that has regularly spaced planters which are obstacles to travel. These planters allow a generally straight line route between origin and destination but to some extent force the traveler to choose ‘channels’ to a destination. The elevator route differs in that only a portion consists of the plaza with planters. The rest of the route area consists of pathways restricting travel between buildings and around grass areas. Furthermore, these pathways radiate out from the elevator, causing diverging paths away from the elevator and
## Table 9

### Route Frequency

<table>
<thead>
<tr>
<th>A to B Routes</th>
<th></th>
<th></th>
<th>B to A Routes</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Route</td>
<td>Frequency</td>
<td>Percentage</td>
<td>Route</td>
<td>Frequency</td>
<td>Percentage</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.530</td>
<td>4</td>
<td>19</td>
<td>0.59375</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>0.28125</td>
<td>1</td>
<td>5</td>
<td>0.15625</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.0625</td>
<td>2</td>
<td>2</td>
<td>0.0625</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0.0625</td>
<td>3</td>
<td>1</td>
<td>0.03125</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.03125</td>
<td>5</td>
<td>1</td>
<td>0.03125</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0.03125</td>
<td>7</td>
<td>1</td>
<td>0.03125</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.03125</td>
<td>9</td>
<td>1</td>
<td>0.03125</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>0.03125</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>0.03125</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Y to X Routes</th>
<th>X to Y Routes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route</td>
<td>Frequency</td>
</tr>
<tr>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
</tr>
</tbody>
</table>
converging paths toward the elevator.

**DISCUSSION**

Other researchers have pointed to the facts of asymmetric distance cognition (e.g., Sadalla, Burroughs & Staplin 1980). These experiments add to their findings by focusing on the paths actually chosen, the criteria apparently most relevant to that route choice, and noting if there are differences between what criteria were used in a field experiment versus those used in daily travel. Some interesting results developed.

First, when comparing the two experiments, laboratory and field, one notices the similarity between the rating of criteria used in the experiments. In the field, minimizing time was given more support belying the result from the lab experiment in which subjects claimed they did not minimize time in everyday activities.

When considering route retracing, two things stand out. Even in this restricted environment, choice of routes varied depending on direction traveled and with respect to the nature of the environment. The fact that on one route (A-B) 62.9% took the same route both ways was significantly different from the result obtained from the other route (X-Y) when only 15.6% took the same route both ways. In the former, minimizing distance or time or turns could provide reasonable explanations for the observed behaviors. For the other route it appeared that route selection criteria changed indicating that a single selection criteria would seriously under predict the paths chosen. No significant differences were observed among males/females and geography/non-geography groups. Also, it did not appear that any one of the end points (flagpole, stairs, elevator) was considered to be a primary reference point and the others secondary. This implies that, in addition to the previously discovered asymmetry of distance perception among anchoring and other nodes, perceptions of the configuration of the environment itself (particularly different perspectives as one changes direction) may influence route choice. Thus, a route that seems shorter or quicker or straighter from one end may not be so perceived from the other end, thus inducing a change of route. The real question is whether the route selection criteria also change;
from examining the actual paths taken and recording response times and other variables, it seems that they often do.

Although the field experiment did not directly test the influence of orientation direction as did the map test, there is room to infer that once again orientation direction played a part in route choice and the criteria used to select that route. Certainly the commonly used assumption that trips will be retraced and that the same criteria will be used for different trips, must be brought into question.

**DIRECTIONS FOR FURTHER ANALYSIS**

Further study in this research project is designed to develop route classification procedures for the various routes actually taken by people in their everyday travel activities. This will determine if the route choice criteria listed in our questionnaire (shortest distance to travel, has fewest turns, longest leg first, most aesthetically pleasing, shortest leg first, has many curves, takes least amount of time, first noticed, has most turns, usual route, alternative to usual route, and always proceeds in direction of travel) are comprehensive or partial. However, it may not be possible objectively to classify routes based on some of the criteria such as: usual route, alternative to usual route, most aesthetic, and first noticed without extensive survey research. However, classification using the other criteria would allow comparisons to be made between the stated criteria used and the actual criteria used. This could be used to answer various questions including: what was the varying importance of the choice criteria when actually traveling in the environment? How does this rating vary for the different conditions? How does this compare to the varying importance of perceived criteria? For non-route retrace what was the criteria that caused a different route choice for the return trip? What difference does it make to predicting travel when one uses different route selection criteria for outbound and inbound trips? Does route selection criteria change with every change of trip purpose? Travel mode? And whether simple or chained trips are anticipated?

I think it would also be interesting to pursue what characteristics of the route areas have caused the differences between route retrace and differentiate route selection between origins and
destinations. With this information as a knowledge base, it would then be appropriate to extend this work to a driving situation (i.e. using motorists as subjects). One could also determine if one or more trip purposes tend to produce route retraces more than others, or if increasing complexity of trip chains produced simple or multiple criteria for each route segment or for the entire trip sequence. A final problem would be to evaluate the degree of realism that can be attributed to conventionally used path selection criteria built into transportation models or the network models built into today's GIS.
REFERENCES


Section VI

Computational Process Modelling of Disaggregate Travel Behaviour.

Computational Process Modelling of Disaggregate Travel Behaviour

by

Mei-Po Kwan
Department of Geography
The Ohio State University
Columbus, Ohio 43210-1361

and

Reginald G. Golledge
Department of Geography
and
Research Unit in Spatial Cognition and Choice
University of California at Santa Barbara
Santa Barbara, California 93106-4060

Paper presented at the IGU-Commission on Mathematical Models
International Geographical Congress, The Hague, August 5-10, 1996
Abstract. In this paper we review attempts to develop Computational Process Models (CPMs) of individual travel behaviour. CPMs represent a linked set of computer programs together with appropriate databases which are designed to capture the essence of human decision making in different spatial situations. They are used to simulate and to predict individual or household travel behaviour, or to provide a means for testing hypotheses about travel behaviour. Several different types of models are identified including information acquisition models, activity scheduling models, wayfinding and navigation models, micro-simulation models, and multi-criterion equilibrium traffic assignment models. The potential for incorporating GIS into some of these models is discussed, and an evaluation of the most promising models for purposes of understanding travel behaviour and predicting travel patterns is undertaken.

Keywords. Computational Process Model (CPM), disaggregate modelling, GIS, decision making, activity scheduling, feasible opportunity set.
Computational Process Modelling of Disaggregate Travel Behaviour

Met-Po Kwan$^1$ and Reginald G Golledge$^2$

$^1$ Department of Geography, The Ohio State University, Columbus, OH 43210-1361, USA
$^2$ Department of Geography & Research Unit on Spatial Cognition and Choice, University of California Santa Barbara, Santa Barbara, CA 93106-4060

1 Introduction

It is common for researchers adopting an activity based approach to travel behaviour to differentiate between behaviours that are routinized and behaviours that result from deliberate choice. For example, a significantly large part of work-trip behaviour is routinized; individuals tend to use the same mode for each trip, to leave their home base at approximately the same time, to aim at arriving at approximately the same time at their work place, and to follow the same route and the same path segments that make up that route. Some other trip purposes are similarly routinized such as trips for religious purposes, and trips for medical or health related reasons - routinized in the sense of using the same mode and following the same path, even though the times at which travel may be undertaken might vary because of temporal differences in the scheduling of appointments by health professionals. Other trips such as grocery or food shopping may be routinized to a lesser extent. Instead of choosing a single destination and following a repetitive path to that destination, several alternative destinations may be kept as part of a feasible alternative set. Trip making on any given day then becomes more of a deliberate choice both in terms of selecting a specific destination and in terms of selecting the travel path. Variation can also occur in terms of travel mode. Many other trip purposes fall within the deliberated choice purview. In particular trips for social or recreational purposes, trips to meet with friends, trips undertaken for the purpose of dining away from home, business trips, and so on, all may be scheduled with different episodic intervals or frequencies, different lengths or durations, different destinations, different temporal units, different priorities, different sequences, as well as being undertaken either as single purpose or multiple purpose trips with single stop or multiple stop destinations.

While the modelling of the routinized choices and the prediction of consequent travel behaviour has been achieved with a considerable degree of success using discrete choice models, dynamic Markov models, and even via variations of the fundamental spatial interaction or gravity type models, less success has traditionally accrued when trying to model behaviours resulting from deliberated choice. As part of the effort to model, explain, and predict trip making, geographers and transportation scientists generally have developed or adopted a number of strategies that focus either on network characteristics (shortest path models), aggregate behaviours (spatial interaction model and entropy models),
individual preferences (compositional and decompositional preference modes), and choice models (discrete choice models, models of variety seeking behaviour, compositional and decompositional choice models) (Timmermans and Golledge, 1990) Discrete choice models have also been used in transportation science for the modelling of choices of modes, departure times, or other characteristics relating to how single trip choices and choice alternative characteristics match up, or to the extent to which individual trip making behaviour matches the behaviour of a group to which they are assumed to belong.

The seminal work of Jones, Koppelman and Orfeul (1990) firmly established a mutual dependency between travel choices and household or individual agendas of activities. Previously, Root and Recker (1983) had suggested that choices of destinations, departure times, and frequency and duration of activity participation should all be treated in a single conceptual framework that entails behavioural assumptions accounting for the process of making these interrelated choices. In other words, they developed the idea of focusing on the activity scheduling process and defining the type of model whose input consisted of components of this scheduling process. While this approach was conceptually and theoretically appealing, it proved difficult to implement within the context of the existing transportation and behaviour models, particularly the dominant discrete choice modelling framework that existed at that time.

As an alternative, a new form of modelling of travel behaviour began to develop based on the idea of a set of interacting computer programs which would relate elements of real and perceived environments, factors influencing choice of destination, household preferences for scheduling activity sequences, and a variety of authority, coupling, and capability constraints that had been offered as part of the emerging field of time geography (Hagerstrand, 1970. Carlsent, Parkes and Thrift, 1978). Simultaneously, awareness of the limitations of simple discrete choice models encouraged the development of tools suitable to model interdependent or joint choices, which include the nested logit (Ben-Akiva and Lerman, 1985, McFadden, 1979) or structural equations models (Golob and Meurs, 1988) evolved. Axhausen and G"{o}rting (1992) have summarized other attempts at estimating discrete choice models in which activities are important components and include summaries of the works of Kitamura (1988), Thill and Thomas (1987), and the trip-chaining models of Damm and Lerman (1981), Kitamura, Kazuo and Goulias (1990), and activity choice and activity duration (Kitamura, 1984) Following the lead of Root and Recker (1983) activity-based choice models emerged (Recker, McNally and Root, 1986a and 1986b), along with some econometric research on time allocation (Winston, 1987). Much of this work, however, invariably rested on utility maximizing assumptions. Questioning of this assumption had been extensive in psychology (Simon, 1955 and 1990, Tversky and Kahneman, 1991 and 1992) as well as in transportation research (Supernak, 1992). In general, these models were limited in that they specified the factors affecting final choice but neglected the processes resulting in these choices. Obviously if the primary aim is to forecast travel choices, this criticism is not an important one, but if the goal is to understand the entire process and to develop appropriate relevant theory, then the shortcoming does become significant.
In a style similar to the STARCHILD model developed by Recker, et al. (1986a and 1986b) another alternative model format called Computational Process Modelling (CPM) emerged. In an activity scheduling context, these Computational Process Models (CPM) focused on interdependent choices where choice involved acquisition, storage and retrieval of information, including retrieval from long-term memory, tradeoffs between accuracy of recalled information concerning locations, hours of business, and remembered paths, in terms of effort (time or distance) expended in order to make the tradeoff and achieve a goal. It also included the possibility for a conflict resolution where uncertainty may exist in terms of competition for a travel mode (e.g., who gets the household car), which activities are considered primary and which secondary and therefore take precedence in scheduling, and which destination choices provide the greatest flexibility and judge success in terms of completing a planned activity schedule. These models are built on some of the seminal ideas of Hayes-Roth and Hayes-Roth (1979) who produced a production system model which was accepted as a feasible alternative to existing discrete choice models for travel behavior analysis.

Production systems were initially developed by Newell and Simon (1972) as elaborations of how people think when they solve problems. They are frequently used in theories involving the higher cognitive processes (Anderson, 1990, Newell, 1992). Essentially a production system is a set of rules in the form of condition-action pairs that specify how a task can be solved. If the task requires an individual to choose one alternative in a choice set, the rules may specify what information is searched under different conditions, how the information is evaluated, how the evaluation or judgments are integrated. The system is usually realized in terms of a cognitive architecture comprising a perceptual parser, a limited capacity working memory, a permanent long-term memory, and a system for effecting a behavior. An operational CPM is a production system implemented as a computer program. The resulting CPM offers a testbed for assessing the consequences of different policy measures, or as a mechanism for facilitating the development of different testable hypotheses. One may also incorporate different testable assumptions into the model to examine their effects on potential choice and consequent behavior.

Essentially a CPM was assumed to be capable of providing a detailed description of the individual's choice process, but there were some questions as to whether or not it was amenable to travel diary data input by which the late 1980's was becoming accepted as a dominant and detailed form of obtaining information for models of activity scheduling and choice behavior (Gärting, Kwan and Gollendege, 1994). The emphasis on this highly disaggregate base also raised questions as to whether or not output could be aggregated in order to provide some reasonable basis for forecasting and policy development. Two parallel alternative approaches have been suggested: microsimulation has been developed for forecasting from systems of disaggregate discrete choice models (e.g., Kitamura and Goulas, 1989) and combining CPM and discrete choice models in a single complementary context (Ettema, Borgers and Timmermans, in press). In either case, one can argue that the intrinsic value of the CPM or production system approach would be to provide the theoretical basis for the microsimulation or CPM/discrete choice model approach. In the following
sections, therefore, we will review a selection of Computational Process Models, discuss a recent contribution from geography which combines the Computational Process Model idea with a Geographical Information System (GIS/CAS) (Kwan, 1994, 1995), and comment on both the microsimulation and CPM plus discrete choice model combination approach.

2 Review Of Computational Process Models

One of the first attempts at developing a CPM of travel choices was that offered by Kuijpers (1978) - the TOUR model. This focused on an individual's memory representation of the environment (i.e., cognitive map), the acquisition of environmental information through search and exploration, and the use of experience and stored memories for making route choices. The model was developed in an artificial simulated environment and lacked empirical application using actual examples of cognitive maps, spatial orientation capabilities, and wayfinding procedures.

A similar type of model (NAVIGATOR) (Gopal, Klatzky and Smith, 1989, Gopal and Smith, 1990), is based more on the principles uncovered during empirical research on the spatial knowledge acquisition and wayfinding abilities of children travelling through a well known neighborhood (Collinge Smith, Pellegrino, Doherty and Marshall, 1985). Using this practical knowledge base, the route planning procedure in NAVIGATOR is modeled by various choice heuristics. When information for making this particular segment choice is lacking, a general route selection criteria such as “moving in the same general heading” or “make a random turn at an intersection” represent options for next segment selection in the path following process. Again, however, there was considerable input from prior empirical testing of human behaviour in route selection tasks in a real environment; the model was still developed in a small hypothetical space.

Route following in a static environment is also the focus of another CPM, TRAVELLER (Leiser and Zilberschatz, 1989). An equivalent type of model in a dynamic environment was labeled ELMER (McCalla, Reid and Schneider, 1982). TRAVELLER simplifies the route selection problem by assuming that the relative locations of origins and destinations are known. This, of course, is perfectly reasonable in most routinized travel activities, it may not be quite so acceptable when we look at the question of deliberated choice where choosing a destination from a set of feasible options is part of the travel planning process. TRAVELLER then constructs a route from origin to destination via a process of search. In this case the production system consists of a set of rules which constrain how search and exploration can take place. In comparison to this the ELMER model conceives of routes as sequences of instructions for how to travel (e.g., go ahead 200 yards, turn right at the intersection). These instructions are retrieved when a particular need arises - for example when one must make a decision about a turn that could result in heading towards or away from a potential destination. Thus, route following is seen as a dynamic decision making process in which choices for segment selection and turns are made en route upon recall of appropriate constraining rules.
The above models, however, did not stress the dependence between travel choice and activity choice. CARLA (Jones, Dix, Clarke and Heggie, 1983) and STARCHILD (Recker et al., 1986a, 1986b) do attempt to address this problem. CARLA in general has the fewest behavioural assumptions but is tied more strongly to various time geographic concepts (e.g., Lenntorp, 1978). In particular it incorporates a variety of time geographic interaction constraints including capability and coupling constraints. Its output consists of sets of feasible activity schedules and sets of possible activity patterns.

STARCHILD is a very comprehensive CPM and emphasizes modelling of the choice between activity schedules. It incorporates a conventional discrete choice model to make such selections, but the authors agree that other theoretically sound choice models could be appropriately substituted. The emphasis in STARCHILD is on the utilities associated with each activity and the sum of the utilities that comprises a particular schedule. Thus, utilities of waiting time and travel time are important features to consider. Perhaps the conceptually weakest part of STARCHILD is its acceptance of utility maximizing assumptions and its use of combinatorics to evaluate all possible feasible choice patterns. In practice, of course, people have limited capability both for considering a finite number of options and for accessing what is truly optimal. A boundedly rational selection mechanism could however be incorporated into STARCHILD thus bringing it much closer to the realities of human decision making.

In an attempt to incorporate more realistic behavioural assumptions and to begin the process of including a perceptually valid environment in the model, Gärling, Brännäs, Garvill, Golledge, Gopal, Holm and Lindberg (1989) outlined a conceptual framework which could be implemented into a model they called SCHEDULER. This model focuses on an individual’s choice of activities, selects from feasible set of destinations, examines possible departure times which are critical in forming a travel agenda for a particular time period. Activities were generally stored in a long-term memory system called “the long term calendar.” Associated with each activity is a priority for waiting time and a maximum duration for completing the activity. A specific activity is retrieved from long-term memory and scheduled on the basis of its relative priority weight and the expected duration required for its completion. Spatio use and temporal constraints including the hours of business and the determination of sets of reachable locations, are retrieved from a memory representation or stored cognitive map of the environment. This is obtained a priori and stored in long-term memory. Choice of the location of a feasible destination and a potential departure time is then determined by the SCHEDULER as it works in a top-down fashion scheduling the highest priority and most repetitive needs first. A possible activity schedule is stored in a short-term memory (the short-term calendar) for possible later execution, depending on whether or not critical input variables such as the duration of time allocated to the activity remain constant or are changed. In this model members of the feasible alternative set are evaluated according to a nearest neighbor heuristic (Hirtle and Gärling, 1992, Gärling and Gärling, 1988). Since in the SCHEDULER the constraints on when activities can be performed (e.g., open hours for business purposes) are part of the initial filtering process, the choice of a feasible location is at least partly determined by the duration allocated to the activity and business open hours during which the
activity can be performed. If a possible sequence of activities is defined but cannot be executed because of a temporal overlap, the activity sequence is redefined by resolving conflict among the competing activities. Here, higher priorities are given precedence. This conceptual model has been expanded by Kwan (1994, 1995) by hosting the scheduling module in a GIS context. We now explore this in more detail.

GISICAS represents an attempt to overcome some of the limitations of existing CPMs in terms of their lack of geoprocessing capabilities to handle the vast amount of real-world location and route data, and to perform spatial search using information about the objective and cognitive environments of the traveler. Although the scheduling algorithm of GISICAS benefited considerably from that of the SCHEDULER, it extended the framework of the SCHEDULER in several ways. Besides an individual's home and work locations, and the priority, duration and timing of activities, a person's preferred and fixed destinations were also included in GISICAS as important elements of the cognitive environment.

The procedures of activity scheduling were integrated with a comprehensive geographic database of a real urban environment and interact dynamically with the module of spatial search heuristics in relation to that environment. Realizing the simplicity of the nearest neighbor heuristic used by the SCHEDULER, several new spatial search heuristics were developed in GISICAS for handling the effect of locational preference and the binding effect of fixed destination on an individual's travel behavior. Basically, they were high-order spatial search heuristics which, instead of looking for the nearest activity locations locally, search for globally satisfying locations in relation to the next fixed destination to which an individual has to travel. GISICAS's procedure for delimiting the choice set in explicit spatial terms also represents a departure from the largely aspatial or pseudo-spatial method of the SCHEDULER. The concept of feasible opportunity set was first formulated by Golledge, Kwan and Gärling (1994) for expressing the effect of bounded rationality on the choice set and implementing the satisfying principle in explicit spatial terms. It was defined with respect to a person's home and work locations. In GISICAS the feasible opportunity set was defined dynamically with respect to the current location and the immediate spatio-temporal constraints of the traveler (e.g., time allowed for the activity and travel to the destination, distance willing to travel, etc.).

This sequential identification of the choice set regarding feasible destinations enables the dynamic interaction of the planning and execution of the activity agenda. The focus of GISICAS on the spatial dimension of activity scheduling and travel decision making suggests an alternative way of modeling travel behavior at the individual level. With the data handling and geoprocessing capabilities of GIS, modeling and analysis of travel behavior does not depend on any prior schema of zonal division of the study area. This may open up a new arena for future research on the use of CPM in transportation science.

3 Micro-Simulation Models

A modified form of the CPM has recently been developed by Ettema, et al. (in press) and Ettema, Borgers and Timmermans (1995). Like SCHEDULER,
SMASH emulates the scheduling process by computing utilities for choices that result in inclusion, deletions, or substitution of activities. In this model, as the number of potential choices increase, disutility occurs. The decision-making process is cumulative and it terminates when no choice results in a positive utility. The utilities associated with the choice of activities depend on the value of priorities for activities in the schedule, travel distance or travel time, the attractiveness of possible locations, the pressure on the individual to complete the activity within a certain time range, and the waiting time before the activity can be implemented and completed. Once a schedule is determined, its realism is evaluated. The SMASH model includes factors that can be assumed to affect activity scheduling and in some respects may be said to have similarities with STARCHILD. It appears to be a more complete model than SCHEDULER. However, whereas in STARCHILD utility maximization is an end product, in SMASH utilities are maximized at each step in the scheduling process. Thus, as each activity is evaluated for possible inclusion, deletion, or substitution in a schedule, the utilities associated with the appropriate process are assessed and ordered. It is thus a more disaggregate and implemented model than STARCHILD and it raises the question as to global versus local maximization. Once again the original SMASH model required all possible choices to be evaluated at each scheduling step which would in reality place a considerable burden on the human traveler. Substituting a feasible opportunity set reduces the magnitude of this presumed computational component.

A significant added strength of SMASH, however, is that the authors have envisaged a way of empirically testing the model using mathematical statistical modelling (Ettema and Timmermans, 1993). The fundamental characteristics of this empirical testing focuses on the choice processes of including, deleting, or substituting activities. These actions are predicted from variables describing the current state of the scheduling process such as the number of activities already scheduled and attributes of the activities that still remain to be scheduled. While some inferences about these actions can be made from examining detailed travel diary records, the authors point out that there still remains much to be learned about the scheduling process itself. Particular items suggested for inclusion in future versions of the model include incorporation of mode choice, the planning of time spent at home, the addition of constraints and the sequence of activities so that some activities are not planned before undertaking others, and the linking of a production system to observed behaviour so that specific parameter values can derived for the model and tested in different environmental, socio-economic, demographic, and mode choice mix environments. This would allow comparison of the outcomes of simulations of a selection of activity schedules using different parameter settings, with the observed scheduling behaviour of humans. Extensive testing in this way should be able to provide a range of "best specification" for parameter values in different environments. As an alternative, parameter values for the model could be collected during interactive simulations in which subjects would be required to complete a task consisting of a number of clearly specified steps that are part of the scheduling process. However, modelling each step separately could provide substantial problems in terms of integrating the results into a single final comprehensive model. At this stage, however, it appears that SMASH, STARCHILD, and the SCHEDULER/GISICAS alternatives are
equally feasible alternatives to further examine the process of activity scheduling and the act of tying together scheduling and travel behaviour.

Epstein (1996) has built a pragmatic navigation model. This model acquires facts about a two-dimensional environment in order to travel through it without an explicit map in its memory prior to travel. Spatial information is accumulated during travel which can be described as sequential trip-making through a fixed maze with specific barriers. Insights about basic points (landmarks), jagged or smooth barriers, bottlenecks, and other obstructions are learned during a sequence of trials. What is learned on one trial is stored and used to help make decisions on the next trial.

As with the other computational process type models previously discussed, this device is tested in a simulated environment consisting of cells of a specific size (i.e., a grid network). The argument is that during experimentation individuals do not build detailed maps of the environment through which they wander but rather encode pragmatic representations of environmental features that assist in path-finding. In this type of world, a robot or other travelling agent can learn its way around in an efficient and effective manner. While doing so, it creates a spatial representation consisting of selected parts of the environment and in some ways can be regarded as similar to the cognitive maps that humans would build in following the same set of tasks. These cognitive maps would be incomplete, in parts fuzzy, and may not lend themselves to exact solution procedures such as minimizing distance or time. In some sense, therefore, the solution is a satisfying or perhaps boundedly rational one. A critical feature, however, is the limited nature of the space and its dimensionality, but it is interesting to note that navigation can eventually take place in a rapid and effective manner even though the environment through which travel takes place is always a partially obstructed space. In the solution process, Epstein uses a “FORR” (“FOr the Right Reasons”) process. This type of program gradually acquires useful knowledge using a hierarchical reasoning process. It relies on different tiers of “Advisors” which are domain-specific but problem-class independent. They provide rational or support criteria for decision making (e.g., get closer to your destination). Each Advisor is implemented as a time-limited procedure and must operate within a constraint set which limits the number of permissible actions. Advisor recommendations are in the form of comments each of which has a weight, salience, or strength attached to it (i.e., an integer from 0 to 10 that measures the intensity and direction of the Advisor’s opinion). Advisors usually do not recommend extensive search activities. Many of the Advisors embody within them the results of comprehensive spatial cognition work from recent decades. This includes advice on alignment, direction of travel, orientation, navigating around obstacles, and neighborhood definition. The results of the application of the model would appear to have relevance for human navigation in that in general there is a tendency to select routes such that there are no substantial changes of direction during the wayfinding process, initially the strongest desire is to move toward the goal, travel is preferred along the main rather than orthogonal axes, there is a tendency to avoid neighborhoods with limited entrance and exit possibilities. On the whole, however, as with many of the other simulation models that could be found in the artificial intelligence literature, this is more a model for navigation and wayfinding than a model that...
explicitly attacks the question of activity scheduling and route selection in a spatial and temporal context that facilitates successful completion of an activity schedule.

Another computational process model recently developed is that by Chown, Kaplan, and Cortenkamp (1995). Chown, et al., offer a model called PLAN (Prototype, Location, and Associative Network model) which is an integrated representation of large scale space which they claim is commensurate with a cognitive map. PLAN is used as a means for including wayfinding as a process which is concurrent with the rest of cognition not apart from it. The model is seeing-based because it takes advantage of the properties of the human visual system which provides continuous answers to the spatial updating or "where" problem. It synthesizes parts of prototype theory, associate network theory, and locational principles together in a connectionist system. The views in the PLAN model are not from an aerial or survey prospective but reflect what an observer would see from different head positions at a single location. This is similar in some ways to the model suggested by Golledge, Smith, Pellegrino, Gopal, Doherty, and Marshall (1985) which differentiates the environment into view (straight ahead linear observation) and scenes (more detailed local observations that might occur with head movements to the side as one is walking along a path). In the latter, views represent what might be seen when looking down the street into the distance, while scenes might be the explicit characteristics and features of a single house that would be observed if one turned their head from a facing direction while travelling. In the PLAN model, path overlap provides the mechanisms for developing new path segments which combine sections of previously experienced routes, and also provide the appropriate geometric base for integrating separate but partially overlapping paths into a more general knowledge of spatial layout (see also Golledge, Ruggles, Pellegrino, and Gale, 1993). Chown argues that a significant advantage of PLAN is that it only stores a fraction of the available information internally and relies on the perceptual system to fill in gaps when PLAN goes into action. This appears to be a reasonable way to operate a robot travelling through a learning environment but it also has some direct similarities concerning the way humans collect, encode, store, and use information as they travel through complex environments. Particularly the idea of storing a minimal spatial representation as a cognitive map and perceptually updating one's location as travel takes place, represents a reasonable wayfinding and search strategy for human behaviour. PLAN thus responds to the criticisms of computational process models offered by Garling, et al. (1989) who complained that an appropriate mix of attention paid to the processes of spatial cognition used by humans to extract and process information about environments as well as encapsulating the essential components, barriers, and paths of the environment itself is required. The model GISCAS (Kwan, 1995) also takes these suggestions seriously and integrates cognitive processes, real-world environmental systems, and the intervening mechanisms associated with Geographic Information Systems (GIS) as a way of handling both wayfinding and activity scheduling problems in complex real-world systems.
4 Multi-Criterion Equilibrium Traffic Assignment Models

Multi-criterion equilibrium traffic assignment models developed in part because of a lack of attention to any realistic interpretation of the value of time problem in traffic assignment. Virtually every transportation model is ultimately evaluated in terms of how users interpret the value of time (VOT) that the model requires them to expend. This is true for mode choice models, congestion pricing models, and traffic assignment models. It is generally recognized that each individual has a different VOT depending on factors such as the person's economic resource base or the time that one is willing or able to spend on a trip. For the most part, transportation planning models acknowledge this by developing an average view of time figure and as a result they invariably produce large estimation errors and inaccurate forecasts. Recently, Ben-Akiva and Bowman (1995) developed a logit mode choice model by assuming that there was a distribution of the value of time characteristics rather than assuming a similar VOT for all users. The result was a significant improvement of goodness-of-fit between predicted choices and actual choices. Dial (1995) consequently proposed a similar remedy for traffic assignment models. His model admits that VOT is best captured by a distribution and it uses a bi-criteria user optimal equilibrium traffic assignment model which generalizes classic traffic assignment by relaxing the VOT parameter in the generalized cross function from a constant to a random variable with an arbitrary probability density function (Dial, 1995). His model, called T2, is said to respond to a variety of difficult existing problems including the mode/route choice problem, parking policy, and congestion pricing. T2 models mode choice by assigning trips to paths in a multi-modal or hyper network. The latter combines walking, riding, transit and highway links. It is able to selectively route auto trips to parking lots that have a specific range of charges associated with them (cheaper lots that may require a longer walk to a destination), or to other higher priced lots that reduce the walking component of a multi-modal trip. It is also touted as being an appropriate model for determining where to place toll booths and what prices to levy in order to reduce congestion. In discussing his model, Dial captures some of the time/cost tradeoffs of a variety of different forms of transportation, but it satisfies none of the behavioural criticisms levied against econometric and mathematically optimizing traffic assignment models. T2 works as follows: Assume it is necessary to make a trip from an origin O to a destination D. The problem is to determine the mode choice for the trip. Dial first assumes that it is possible to enumerate all feasible paths for this trip and to know the time and cost of each path. Each path is then plotted at a point in a graph according to its time and cost. One might show fifteen feasible "paths" in terms of time and cost between a given O-D pair. Today, a helicopter is often the fastest mode and the most expensive while walking is the slowest and cheapest. Dial then examine the process of selecting among possible mode/time/cost possibilities. It develops a path-finding algorithm which examines all possible combinations of mode/time/cost and determines an optimum. In an argument similar to that used to determine feasible alternative destination sets from among all possible sets that was developed in SCHEDULER and GISICAS, Dial differentiates between mode/time/cost combinations by determining a likelihood that a particular combination will be...
chosen. In this way many potential combinations are eliminated and only those few feasible paths connecting a given origin and destination are examined. This final traffic assignment model accumulates trips for all O-D pairs and defines a user-optimal traffic assignment. This solution is termed a traffic equilibrium, and focuses on the relationship between travel times, cost, and volume of flow along arcs of a given path.

This model is a return to the classical mathematical/econometric model of traffic assignment, and does not deal with travel behaviour. It does make some concessions to the CPM and discrete choice related research which emphasized the significance of disaggregate units and individual differences in evaluating activity schedules and travel paths by including a value of time distribution characteristic rather than a single VOT estimate across an entire population. The question remains, however, whether it is suitable primarily for the routinized travel behaviour or whether it is robust and versatile enough to also incorporate path selection, mode choice, and travel behaviour for the other activities we have previously described as resulting from deliberated considered choice, with conflicts being resolved in a dynamic on-route environment.

5 Summary And Discussion

In this paper we have reviewed and assessed a variety of approaches to the activity scheduling and travel behaviour problem. Among these were a range of computational process models and some recently emerging alternatives including microsimulation, multicriteria traffic assignment models, and combined CPM/discrete choice models. Of this set we have argued that the CPM approach allows one to move closest to the real world decision making and choice situation and allows us to incorporate elements of both objective and cognitive environments as the matrix on which activity schedules and travel choices are made. For traffic assignment in a multimodal environment, the T2 model appears to have significant promise. Of the various CPMs reviewed, SMASH and GISicas both appear to be flexible, expandable, developed at the individual level but capable of aggregation, and suited for testing in real-world environments. STARCHILD also has been tested on real-world travel diary data with considerable success. It is quite possible that with some modifications such as the inclusion of a procedure for determining feasible alternative sets rather than going through complete enumeration of all possible activities, embedding the model in a Geographic Information System in a real-world environment, and allowing a boundedly rational selection criteria to replace the simple maximizing utility assumption, then STARCHILD could be expected to achieve a considerable success at predicting and forecasting travel behaviour. Of the models examined, SMASH attempts to integrate the recent CPM approach with the more traditional discrete choice model approach and by combining the most powerful aspects of each, provides significant hope for successful application - which appears to have recently been completed (Ettema, 1995).

In order to continue evaluating and assessing the different avenues of current research, specific empirical testing will be required. This requires explicit testing of the behavioural assumptions entailed by the different models as well as by
assessing their ability to consider different mode combinations, and different combinations of activities in a schedule. GISICAS argues that scheduling must take place over a longer period than a day for some high priority activities (e.g., food shopping) may only be undertaken every second or third day. At that time, however, their importance is extremely great and scheduling must be adapted to allow the activity to take place.

The increasing volume of travel diary, panel, and survey data is providing more and more insights into the process of travel behaviour and mode choice. However, detailed testing of how activities are selected and how the selections are combined into schedules is still a critical point for future research. GISICAS is the only model so far to explicitly incorporate a Geographic Information System (GIS) into its structure. Obviously the potential for GIS must be further examined. This examination should include the suitability of GIS as a host for recording diary, panel or survey data, as well as its potential for developing an interactive framework for the assessment of priorities associated with activities and the choice of an appropriate path selection model. As far as the latter is concerned, more work needs to be done on the criteria that can conceivably be used in the path selection process. All too often assumptions are readily accepted that minimizing distance or minimizing time or cost are the only criteria worth considering. Recent research (Golledge, 1995a and 1995b, Kwan, 1994) has indicated that there are a number of other feasible path selection criteria and spatial search heuristics that may have to be made available in a group of models that could potentially be used in successful path selection and travel behaviour prediction. A GIS seems to be a reasonable host for incorporating a variety of models which can satisfy criteria such as minimizing turns, always heading in the direction of your destination, selecting the longest or shortest leg first, maximizing the aesthetic value of the route, minimizing perceived or actual costs, minimizing perceived or actual distance, minimizing perceived or actual time. Most model frameworks adopt one or another of these path selection criteria in their trip selection criteria. It appears that different criteria may well be used for different trip purposes. If this is the case then standard models based on single criteria cannot possibly hope to satisfactorily predict travel behaviour. A GIS that includes a set of path selection algorithms which could be initiated by a predisposition of a traveler to select certain criteria for certain purposes, could add significantly to our ability to understand and perhaps even to forecast the complex set of trips that make up the activity patterns of population aggregates. It is our intention in the future to continue working on these problems and, in particular, to determine the type of GIS (e.g., object oriented or relational) that will best lend itself to the procedures defined above.

References:


Carlstein, T., Parkes, D., and Thrift, N (1978) Timing space and spacing time, 3 Volumes London Edward Arnold


Dial, R B (1996) Multicriterion equilibrium traffic assignment: Basic theory and elementary algorithms Part I - T2 The bicriterion model. Transportation Science (accepted)

Epstein, S L (1996) Spatial representation for pragmatic navigation Personal Communication, Department of Computer Science, Hunter College and the Graduate School of The City University of New York


Gopal, S, Klatzky, R L and Smith, T R (1989) NAVIGATOR A psychologically based model of environmental learning through navigation *Journal of Environmental Psychology*, 9, 4 309-332


Hayes-Roth, B, and Hayes-Roth, F (1979) A cognitive model for planning *Cognitive Science*, 3, 275-310


Jones, P M, Dix, M C, Clarke, M L, and Heggie, I G (1983) *Understanding travel behavior* Aldershot Gower


Kitamura, R, and Goulias, K G (1989) MIDAS A travel demand forecasting tool based on dynamic model system of household car ownership and mobility Unpublished manuscript


McFadden, D (1979) Quantitative methods for analyzing travel behavior of individuals. Some recent developments In D Hensher and P Stopher (Eds ), Behavioural travel modelling London Croom Helm, pp 279-319


Recker, W W., McNally, M., and Root, G S (1986b) A model of complex travel behavior Part II An operational model Transportation Research A, 20, 4 319-330

Root, G S, and Recker, W W (1983) Towards a dynamic model of individual activity pattern formation In S Carpenter and P Jones (Eds ), Recent advances in travel demand analysis Aldershot Gower


Winston, G C, 1987, Activity choice: A new approach to economic behavior
Journal of Economic Behavior and Organization, 8, 567-585
Section VII

Information Representation for Driver Decision Support Systems

Information Representation for Driver Decision Support Systems

Mei-Po Kwan
Department of Geography
Ohio State University
Columbus, Ohio 43210-1361

Reginald G. Golledge
Department of Geography
and
Research Unit in Spatial Cognition and Choice
University of California at Santa Barbara
Santa Barbara, California 93106-4060

Jon Speigle
Department of Psychology
University of California at Santa Barbara
Santa Barbara, California 93106


Acknowledgment: This research was funded by a grant from the University of California Transportation Center Grant DTRS92-G-0009 “Object-Oriented Dynamic GIS for Transportation Planning”
Abstract:

The successful development of Intelligent Transportation Systems (ITS) depends on the capability of incorporating a vast amount of information about the location of facilities which generate travel as well as a realistic representation of elements of the transportation network in which travel occurs. An integral part of this system is an Advanced Traveler Information System (ATIS). Such a system can be based on an innovative and comprehensive Geographic Information System (GIS). Whereas current ITS primarily use simplified transportation networks as their basis, using an object-oriented GIS allows us to provide a more realistic representation of elements of the network and the ways that people perceive them. We can represent the network by defining roads or street hierarchies and by storing environmental data as layers which can be overlain, aggregated, or decomposed at will. Storing the transportation network as a hierarchy facilitates the calculation of different paths through the network and allows the introduction of different path selection criteria. A long-run aim of ITS is to develop a real-time multi-strategy travel decision support system over a multi-modal network. We examine the advantages of an object-oriented system over the link-node system in pursing such a goal. We also identify the shortcomings of link-node technology that are overcome by using an object-oriented data model. And finally, we discuss some of the theoretical and applied implications of our suggestions.

Purpose

Intelligent Transportation Systems (ITS) utilize advanced communication and transportation technologies to achieve traffic efficiency and safety. There are different components of ITS, including Advanced Traveler Information Systems (ATIS), Automated Highway Systems (AHS), Advanced Traffic Management Systems (ATMS), Advanced Vehicle Control Systems (AVCS) and Advanced Public Transportation Systems (APTS). Development of a system for ITS depends on our ability to deal with a vast amount of information about the locations of places as well as with the complex representation of the transportation network linking those places, and to incorporate these into a geographic database. The system therefore needs to be constructed based upon the foundation of an integrated and comprehensive Geographic Information System (GIS). As compared to the simplified node-link graph theory representations of transport networks used by current ITS, GIS are able to provide more realistic representations of elements of the complex environment.

Transportation Science has an expressed goal of increasing accessibility for all groups of people with regard to the environments in which they live and interact. A significant component of these goals is to further develop Intelligent Transportation Systems (ITS) through multi-level and multi-modal research and testing. This includes contributing to research and transportation system architecture, technology development, policy formation, and operational tests of various systems including ATMS, ATIS and APTS. In this paper we focus on ATIS.
ATIS Characteristics

Traffic congestion is a problem that appears to be increasing in a world-wide context. In recent years considerable effort has been paid to the investigation of methods to reduce such congestion and the accidents and hazards that are usually associated with it. Collectively these efforts come under the aegis of Intelligent Transportation Systems (ITS). A critical part of ITS are the Advanced Traveler Information Systems (ATIS). Essentially these consist of in-vehicle information and guidance systems which help the driver to select routes which will reduce congestion, to find parking in areas where it is sparse, and to facilitate rescheduling of activities when congestion makes this a feasible alternative. It is argued that such assistive information will benefit individual drivers in terms of helping to achieve their scheduled behaviors and activities as well as benefiting the system by improving traffic flow.

The objective of an in-vehicle guidance system is to assist the driver to select routes which will help reduce congestion. Such a move will benefit both drivers in terms of helping to achieve their scheduled behaviors and activities, as well as benefiting the system by improving traffic flow. Before the information given by the in-route guidance system can be effective, however, it must be perceived as being valuable by the driver. The driver must recognize that information so obtained is valid and reliable, the driver must accept that the action if taken will not increase stress levels or route him through fearful or dangerous places, and the driver must be willing to execute an appropriate action promptly.

The continuing increase in vehicle miles of travel and the corresponding environmental degradation, personnel time loss, increased congestion, and decreases in safety, have lead to the suggestion that ATIS may represent a feasible solution to many of these problems. The ATIS is usually conceived as an in-vehicle route guidance system with supplementary information that allows changes in activity scheduling. The objective of an ATIS is to reduce the impacts of congestion by offering information to drivers that will help them select alternate routes that should benefit them individually as well as benefiting the operation of the system as a whole. While there is no doubt that drivers may respond in markedly different ways to in-vehicle information and route-guidance data, it is still suggested that a versatile ATIS will be capable of being personalized by each driver by acting as a decision support system and informative supplement to the driver’s knowledge base.

ATIS as a Decision Support System for Travel

Decision Support Systems (DSSs) are integrated sets of tangible and intangible information that are designed to supplement a decision maker’s personal knowledge base during problem solving activities. The principal objective of a DSS is to support decision making by humans, not to replace it completely with computerized recommendations. Use of such a system is presumed to bring to bear on a problem the strengths of personalized expert knowledge and comprehensive exogenous knowledge that may not normally be available to the decision maker. The result should be an informed and intelligent decision.
An ATIS can be considered a DSS that is designed to provide a set of information to a driver while on-route to help solve problems (e.g., what to do when faced with congestion, hazards, or other barriers to movement such as construction). To achieve this goal, the ATIS should not provide just a single set of commands or directions, but rather allow a driver to use whatever criteria is most acceptable in terms of deciding whether to wait out the delay effects, or to undertake travel changes such as rerouting, rescheduling activities, replacing activities, changing destinations, and so on. But ATIS should be considered as a supportive tool designed to help the user, without automating the total decision process according to previously established sets of objectives or by imposing solutions which may be unacceptable to the driver (e.g., recommending rerouting that takes the driver through what may be imaged as a dangerous neighborhood). When using an ATIS as a DSS, it must be designed to be easy to use, it must have a user friendly in-car interface, it must help drivers achieve travel objectives and not divert them from attaining those objectives, and they must be designed to enable a user to benefit fully from the types of information dispensed. The ATIS then becomes a decision aid which might allow a user to generate a series of alternatives prior to making a critical decision. It thus provides the traveler with an opportunity to find a good or satisfactory solution without imposing the need to seek an optimal solution.

In the domain of surface travel by road, implementing an ATIS as a DSS can be complicated. The first problem arises in determining whether the information dispensed is most relevant to solving a system problem (e.g., clearing a point of congestion, hazard, or traffic delay) in as timely a way as possible, or providing information to allow drivers to make their own decisions on rerouting, rescheduling, and so on. Up until this time, the best quality and most quantity of information that has been geocoded and stored has related to the environmental base (i.e., transport network). Given this emphasis, information has been available to handle a system problem in a timely way, by determining alternate routes around a barrier or obstacle.

In recent years, however, there has been an increasing effort to focus on the driver as a recipient of advanced traveler information by developing a series of simulators which allow manipulation of environments and observation of different driver behaviors (see Koutsopoulos, et al. 1995). The use of such simulators is most helpful when attempting to decide the likelihood that drivers will accept information dispensed through their in-vehicle guidance system.

The development of driver simulators and the consequent modeling of driver behavior has been an attempt to enrich the potential of ATIS methodology. Simulators themselves range from PC based pseudo game systems in which a participant moves a cursor on the screen (representing a vehicle) through a two-dimensional network or maze in attempts to achieve some goal that is constrained by à priori determined barriers. Other simulators are more comprehensive and involve video or time-sliced slide displays of actual road conditions. Of significant importance, however, is the increased acknowledgment that these simulations work best if the environment is designed to have as many important real-world features as possible (e.g., traffic lanes instead of undifferentiated network arcs, signalized or otherwise controlled intersections instead of unconstrained nodes in a graph theoretic representation of a system; indicators of traffic speed and driving headway; etc.). Emphasis on driver behavior has also developed.
beyond simple observation of actions to the stage of providing supplementary
information to the driver so that normally considered criteria can be used in decisions
concerning rerouting, rescheduling, destination substitution, and so on. The most
productive driver simulations are those in which the supplemental information allows
decision makers to use their personal skills and expert knowledge in a flexible and
adaptable way. But, while acknowledging these needs, it is also necessary to realize that
an interactive system (e.g., in-vehicle guidance) can only be effective if it works in a
realistic time frame and does not delay the decision-making process of a potential
traveler. Thus, not only do our drivers require timely decision support, but they must also
be able to comprehend the information provided, integrate it with their goals and
objectives as they exist at a given time and place, and make a decision that will benefit
both the driver and the system.

Regarding an ATIS as a driver decision support system provides a framework for
developing a potentially useful ATIS. The first filter provided by this framework is
whether a driver is capable of synthesizing the information provided in such a way that
personal and system-wide benefit will occur after a decision has been made. A second
component of this framework concerns the system’s construction, which involves
selection of a type of database, a user interface, the type of display for distributed
information, the amount and type of information, the range of alternatives that may be
considered to replace current actions, the user friendliness of the display system, the
complexity of the information provided, and the probability that a driver will use such
information. With these guiding principles in mind, we now turn to suggest one way of
developing such an ATIS using a Geographic Information System (GIS) as a host. We
will also discuss the relative advantages and disadvantages of two different types of GIS —
the longer existing relational based GIS which uses link-node network structures, and the
more recently emphasized object-oriented database and data model.

**Development of a GIS for ATIS**

The first requirement of a GIS for ATIS is focused on the ability to represent the
transportation network in detail in order to apply different routing algorithms for both
modeling of movement and simulations of flows. In the real world, a transportation
network has different types of roads (e.g., highways, arterials, local streets, etc.) and
different types of intersections (e.g., signalized and non-signalized) that are of interest to
ITS builders (Figure 1). For some applications, information on intersections, lanes and
lane changes such as turn lanes and merging lanes, highway exits, etc. may be important.
We may also require geometric representation for road curvature and incline.

Current GIS data models usually represent the network as a collection of links and
nodes. Whenever two links intersect, a node is created. Unless additional structure is
added, the link-node nature of topological data models is basically planar and cannot
distinguish an intersection at grade from an intersection with an overpass or underpass
which does not cross at grade. It is difficult to accurately represent
overpasses/underpasses and this situation may lead to problems when running various
routing algorithms.

There are other situations in which we may want to differentiate types of nodes.
First, if there is a decision point that the traveler needs for movement, the node-link
model usually slaps that point to the nearest node regardless of type. Thus, this approach ignores the different types of links and nodes making up the complex network. Moreover, when the traveler needs to make a decision to move in the network, there are many decision points that he/she needs to encounter in addition to intersections (e.g. merging lanes, divided lanes). Sometimes, different nodes may lead to different behaviors. For example, we usually make a U-turn at a dead-end node. This behavior is commonly not available in much of the link-node software.

Second, since ITS have to operate at various spatial scales, a multi-level representation of a transportation network is needed. At the local level (i.e., city or county), the transportation network needs to be represented in very great detail for navigational purposes, whereas only the major elements such as the interstate on of state highways need to be represented in the case of regional, state or interstate travel. Further, relationships between data at different levels have to be established (e.g., solving the aggregation problem) Only through the use of a more efficient geographic data model can we hope to overcome these and other related problems (Roberts, Gahegan, Holt, & Hoyle, 1991). Several recent attempts to tackle this problem have focused on the object-oriented approach (e.g. Mainguenaud, 1995, van Oosterom and Schenkelaars, 1995).

A third problem concerns a system’s ability to handle the real-time update of traffic patterns and transmission of information. To provide travelers with timely decision support, an ATIS needs to receive and transmit data in real-time through a communications network connecting a vast number of system elements, such as traffic sensors and location-tracking devices. Even for a small city like Santa Barbara, which has about 14,000 segments and approximately 28,000 nodes in its street network, the real-time update and transmission of data is a very challenging problem. To keep information on the dynamic environment current and self-consistent even in the presence of concurrent updates and queries from around the network, we need to go beyond existing data handling technologies. The system has to be interoperated with other transportation software and requires certain spatial partitioning for effective access.

To overcome these limitations, we explore some of the problems that must be faced when building a multi-strategy travel decision support system using object-oriented data modeling and database technologies. In such a system, the realistic representation of elements of the real world travel environment will be handled by an object-oriented GIS data model. In another paper we explore how the system can handle transit routes in addition to highway traffic in a multi-modal environment.

**Implications of Using GIS in an ATIS Context**

Geocoded data systems are complex. Incorporating such systems into a dynamic real-time multilevel ATIS system makes them even more so. Objects, when geocoded, have both geographically and temporally referenced geometry. That is, objects are located in time and in space. These spatio-temporal characteristics tie all the geocoded data together. But, spatial databases are usually extremely large. For example, in the Santa Barbara area, with a population of about 180,000, the street system consists of 14,000 arcs and 28,000 nodes. When we add descriptor and attribute information to every segment and node, the size of the database escalates by many orders of magnitude. In geocoded data accuracy and relative accuracy are important. Relative accuracy is more important than global
accuracy because the real-world activities that are modeled or simulated within the database are usually local in nature.

In databases of transportation networks that are tied to link-node representations, real-world accuracies (e.g., curved streets and multi-path intersections) are not always represented accurately, but are stylized into graph theoretic renderings And, with respect to the current state of knowledge, we have very little idea of the amount of inaccuracy and error that may be present in many of these large databases. The presence of such error will, of course, give unreasonably bad performance when attempting to either model travel behavior or to send information that will cause revision of activity schedules and consequent changes in spatial behavior. However, it appears that, when considering these problems, the use of object-oriented database systems and object-oriented data modeling strategies now have the greatest potential for use when attempting accurate and usable information transmission and the consequent modification of travel behavior.

Helping the Traveler Resolve Conflicts

We have suggested that one of the most common uses of an ATIS will be to provide information that will help a traveler overcome the problems associated with congestion. These problems can be summarized as: rerouting, rescheduling, destination substitution; activity compression, and activity deletion.

Rerouting decisions require information on the nature of the roads that make up the path selected as an alternative to that currently being traveled. The database must therefore have the capability of incorporating all levels of a road hierarchy, from highways to local streets. In addition, information must be available on the type of flow (e.g., one-way or two-way flows) and the directionality of the flow (i.e., towards or away from a potential destination). Data such as the number of lanes, whether on-street parking exists, the type of neighborhood the selected streets pass through, and the location of facilities (e.g., commercial, business, or other establishments) must also be accessible. In addition, access to information concerning the expected time of travel along the new route requires accumulation of expected travel times on each level and segment of the general road hierarchy making up the selected path. Other features that might be captured in a visual display (e.g., whether the ultimate route parallels the new one or proceeds orthogonally or diagonally away from it, as might be expected if the alternate route represents a shortcut), is required.

The occurrence of congestion may cause gridlock. In that case rerouting may not be a feasible alternative. If the driver is locked in a tightly linked lane of traffic with no immediate chance of escape (e.g., on an interior lane of a multi-lane highway), it makes little sense to provide information about alternate routes that could be taken. Rescheduling of activities then becomes a possible alternative. Most drivers do this mentally. However, an ATIS option that assist in rescheduling. Such an option has been offered elsewhere by Garling, Kwan, and Golledge (1994), and Kwan (1995), in the form of a Computational Process Model linked to a Geographic Information System (GIS). These devices, called SCHEDULER and GISICAS respectively, allow conflict resolution in both temporal and spatial domains by rescheduling prioritized activities to give precedence to those with the greatest needs, providing temporal constraints are not violated.
Along with potential rescheduling of activities comes the possibility for substituting among destinations. Rescheduling, in fact, might encourage the traveler to exit the congested area at the earliest opportunity and to undertake travel to a new destination which is in a vastly different locale that the original one. To make these destination substitutions, the ATIS must contain sufficient information to allow choice of alternative places to go. Destination substitution may have only a limited possibility with respect to work trips, but for many other trips it can become a viable alternative (e.g., substituting a different restaurant for the one originally planned to patronize).

Congestion invariably involves time loss. Time loss may prevent extensive rescheduling, just as the nature of the surrounding road system may make rerouting a less desirable alternative. One option, therefore, is to undertake activity compression. This would simply maintain the original activity schedule but reduce the amount of time allocated to each remaining activity. Thus, a two hour recreational period might be reduced to forty-five minutes, a one hour shopping trip reduced to fifteen minutes, and so on. Often activity compression is also associated with destination substitution. For example, one may have planned to visit a specialty food store to pick up goods for an evening meal, after experiencing traffic congestion and the need for activity compression, a destination such as a fast-food restaurant may be chosen as an alternative to the original one.

If time loss because of congestion or barriers is significant, a final option is activity deletion. In this case the ATIS would need to provide sufficient information (or to have sufficient information contained within its programming) to allow the decision maker (or to recommend to the decision maker) that low priority activities be deleted. For example, if at lunch time one was planning to get a haircut, but was caught up in congestion while traveling to the barber, that activity could simply be deleted from the day’s schedule and reinserted at some later appropriate time. Again, however, information necessary to allow deletion to take place must be provided in a timely, clear, and usable manner.

Requirements of a GIS for use in an ATIS context.

Perhaps the most fundamental requirement when considering building a GIS for use in an ATIS context is the degree to which it is a realistic representation of existing network elements. Such a representation would need to include a road hierarchy, directionality of traffic flow, number of lanes, turn restrictions at intersections, signaling and signage (e.g., traffic lights and stop signs), existence of pedestrian crossings (particularly school crossings), presence or absence of bicycle lanes and on-street parking, the presence or absence of dividing strips, the opportunity for making U-turns, the nature of the road’s surface, allowable and safe speeds, street gradients, driveway access from streets, whether or not through traffic is possible, and so on. Many existing transportation networks used in ATIS simulations and model building have been developed simply as node-link graphs, and while some of the above attributes are included, many of them are not.

In addition to the realistic representation of the street network, information is needed concerning the multimodality of each system element. In other words, the potential for linking with other transportation modes (e.g., buses, trains) must be included. In addition, the ATIS and its GIS components must be amenable to spatial and temporal update as permanent changes occur in the environment or as significant temporary
barriers or changing circumstances come into existence (e.g., closure of entrance or exit ramps on freeways for certain time periods to allow widening, change of grade, or other features)

Perhaps the main purpose, however, of having a good GIS in an ATIS is to provide alternative models for vehicular routing and navigation. Any existing ATIS, whether designed for use in practice or in simulated conditions, contains only a restricted number of routing algorithms. These are usually based on minimum distance considerations and are most easily implemented in a simple node-link network. However, unless a very comprehensive set of attributes are associated with each node or link, this simplified graphical solution to navigation and routing may prove not so simple after all. For example, if one is being allocated to a route segment which has a lane divider that prevents U-turning, and U-turning is impossible at the next intersection, how does one access a destination on the other side of the road? Similarly, in the simplified graph theoretic network representation, a route may require turning at a node which in reality is not an intersection but an overpass or underpass. Obviously, without knowing the constraints of traffic flow directionality, one may further come up with a routing solution that indicates travel should occur against the legal direction of flow. Or simply, the driver may wish to implement a routing algorithm that uses a path selection criteria other than shortest path. This alternative may depend on trip purpose (which may change if activity rescheduling occurs), and has been examined elsewhere (Golledge, 1995).

Obviously many of the existing GIS that rely on link-node structures have difficulty in meeting the various informational needs mentioned above. Their information base is incomplete, their potential for conflict resolution is limited, and their ability to use human perceptions is likewise limited.

In comparison to the link-node systems, object-oriented Geographic Information Systems (OOGIS) allow for class hierarchies among objects and provide for the possibility of the inheritance of attributes among different levels of a class. For example, a general level of the class “road” might be “highway.” Many of the characteristics of a highway can be inherited by the “children” or lower level members of the general class of roads—such as arterial streets or neighborhood streets.

An object-oriented database allows a map to be divided into layers, each defined by a common set of attributes. Each layer may consist of entities encoded as circles, lines, points, polylines, nodes, links and line attributes. Object-orientation also provides a means of partitioning a map and of displaying the zoomed-in partition. Partitioning allows the rendering of only those sections of the map which overlap with a currently defined viewing box. As the viewing box expands to incorporate more layers, it also incorporates adjacent partitions. Attribute lists relevant only to each partition (partitioned entity lists) can be displayed and consequently the time and effort of examining unnecessary data is conserved. Partitioning can take place in a number of different ways; we have used quad-trees.

Our object-oriented GIS can also undertake hierarchical layering. For example, if we define a road layer class we may then be able to derive more specific types of roads from this class (e.g., highway, local street). This establishes a kind of “parent-child” relation between objects. The “child” class inherits the properties of the “parent” class including the parent class’ attributes and its associated functions. For example, a relation might be
established between a class labeled "car" and a class labeled "vehicle." It could then be said that "car" is derived from "vehicle" and that "car" is a type of "vehicle." As a type of vehicle it would inherit many of the attributes associated with the class vehicle.

An additional favorable characteristic of an object-oriented GIS is called 'polymorphism.' This represents the quality or state of being an object which is able to assume different forms. In a programming context it means the same construct can be used to manipulate objects of different types. For example, one can conduct "overloading" in which the same operator ("+" or "print") can perform different functions depending on the recipient's object class.

In the link-node GIS structure there is an assumption that the transportation network is represented as links and nodes and there is no differentiation between these links and nodes. Consequently the delay function at intersections, for example, cannot be easily modeling differently for signalized intersections versus non-signalized intersections, or highways versus local streets. With polymorphism in an object-oriented context, these different elements in a network can be modeled using different functions, but with the same operator ("cost") used to calculate the delay function in all circumstances (Khoshafian and Abnous, 1990). An object-oriented system also allows for parametric polymorphism. This provides for the construction of abstract classes, with different type parameters. For example, a delay function can be formulated as COST(D) with D as a parameter of different type of distance measures. In transportation research, distance is often measured either as travel time or over-the-road distance. So the parameter D can be implemented as a type of integer (when used in terms of minutes of travel time) or as a type of real number (when used in terms of meters of physical distance). Both sets share the code as it is implemented in COST(D).

The relevance and reliability of the information contained in an ATIS is a key component of such travel aids. Thus, for both travel behavior simulations and for real-world applications, the need for development of a database and data model that reflects complexities of real-world conditions is essential. It is in this context that we argue for the development of Geographic Information Systems (GIS) as an appropriate host technology. In the remainder of this paper we will examine some of the characteristics of existing and potential GIS databases for ATIS use. And, finally, we will discuss the theoretical implications and model building implications of having access to more comprehensive and more realistic databases that reflect transport network conditions and driver behavior conditions in ways that are closer to real-world situations.

Koutsopoulos, Polydoropolou, and Ben-Akiva (1995, 144) have raised a number of questions concerning the benefits and effectiveness of ATIS. Since our paper similarly addresses some of these questions and tries to suggest how GIS can play a role in such matters, we repeat their questions as follows:

- "What role can ATIS play in alleviating traffic congestion?"
- "What impact can ATIS have in altering traveler's decisions concerning their trip and destination?"
- "What impact can ATIS have in model shifts?"
- "What impact can ATIS have on departure time and parking choices?"
- "How much are users going to pay to gain access to different ATIS services?"
• “How do users evaluate different ATIS features?”

Their suggested answers to these questions include concerns such as the degree of awareness or level of knowledge of a traveler about the existence of ATIS services; the traveler’s willingness to acquire equipment or subscribe to services providing ATIS information, the probable frequency of usage of ATIS services; the probability that a traveler will make a response requiring a change in travel behavior after receipt of information, and the degree to which multiple exposures to an ATIS system and the consequent learning experiences, are evaluated as useful and favorable. They also further argue that because of the difficulty of collecting data in real-world environments, travel simulators have become more popular. However, their design and usefulness depends to a large degree on the user interfaces representing travel, network, and information environments.

It is our contention that by developing the capabilities of GIS, particularly using object-oriented databases and data models, that some of these disadvantages and limitations, relating to the traveler environment, the network structure, and the activity schedules and travel behaviors of individuals, will be positively enhanced.

Towards an Object-oriented GIS

Worboys (1992a & b, 1994a & b), and Worboys, Hernshaw, & Maguire (1990), surveyed the current state of the object-oriented paradigm as it applies to the handling of geo-referenced information. He outlined the major concepts behind the approach and its application in handling spatial information. These concepts have also been presented in many of the pure research papers, some of which will be discussed below.

Worboys defined a geo-object, which unifies spatial, temporal, graphical and textual/numerical objects in his conceptualization. He also pointed out the little use of proprietary object-oriented systems, except in cases like Milne, Milton & Smith (1993) and David, Raynal, Schorier & Mansart (1993). Research on the extended relational system has also faced difficulties. Projects in extended relation DBMS on applications of geo-referenced information include work by Lohman, Lindsay, Pirahnech, & Schreifer (1991), and Rowe & Stonebraker (1987).

Some object-oriented systems have been implemented using object-oriented modeling and commercial OODBMS. Williamson & Stucky (1991) developed a generic GIS supporting Earth resource imaging analysis. The system consists of (1) a graphic, raster and text interface; (2) a database containing maps, images, and graphical and textual descriptors; and (3) a collection of processes which transform the representation and content of the data objects in the GIS. The system purportedly improved quality of reports, database updates, and analysis and more timely access to widely dispersed information. They argue that a more intuitive interface with the information in the database also reduces the necessary minimum training level for analysis.
Comparison of object-oriented and relational approaches

Typical modeling approaches use relational databases for data modeling and as the base for programming languages they use to model different transportation processes. However, the object-oriented approach has been said to have superior modeling power because the DDL and DML are merged with the OO programming language. Because of this there is a tendency for the use of OODBMS to expand in comparison to relational models. While the latter models simply describe system states, object-oriented models can describe both system states for data and system processes or behaviors in an integrated context (Booch, 1991, Rumbaugh, et al 1991).

Some limitations of existing DBMS (relational) have been identified by various writers such as Herring (1992). First, relational RDBMS lack extensibility to provide for special application needs (e.g., provisions for the user to add new data types and methods, addition of user-defined code, design of new storage methods, and access to standard packages). A table is created for each entity type, a row corresponds to an entity and columns contain the attribute values. However, a relational schema ends up with many additional tables because an attribute is restricted to being a simple built-in type. An attribute may not be a set of values, and relationships are also modeled using tables. For GIS spatial and non-spatial data, they are complex enough that spreading entities and attributes into numerous tables is undesirable (Wiegand & Adams, 1994).

Some of the advantages of OO modeling are that related data can be kept together and relationships can be directly modeled (Medeiros & Pires, 1994). The new geographic data models based on a set of feature objects provide the needed framework for a scaleless and seamless database (Mainguenaud, 1995, Guptill, 1989). It also has the DBMS characteristic of extensibility needed by GIS (Haas & Cody, 1991). For example, the ability to add new data types (e.g., points) and operations on them (e.g., distance functions) and the ability to have a new set of operations as part of the query language (e.g., overlay) are added benefits (Gunther & Lamberts, 1994).

The use of relational databases has been popular in the past and because of this, a set of reliable and working software tools have been developed. This software includes the core database engine, modeling tools, and application development utilities. It usually incorporates a powerful query language (SQL) which has a sound mathematical basis in relational calculus (Date, 1985) and, usually operating on collections of fixed format tables. As opposed to this, the tools in an object-oriented approach draw on a semantically richer background. Unfortunately, this has allowed greater personalization of the approach and a standard query language has not yet emerged. As a result, two different types of object-databases have developed, one of them following many of the ideas of the relational model and the other designed to be integrated with an OO programming language (such as VERSANT). The goal of VERSANT’s type of OODBMS is to transparently provide persistence to the classes set in the OO programming language. The relational model provides enhanced relational databases with an object interface. Those tied to VERSANT or other similar languages emphasize persistent objects. Other tools include translators between objects on the program side and relational sets on the storage side. Regardless of the tools selected, there is a need for more application-oriented technology. The development of this software,
however, must focus on questions of its modularity, its performance, its scaleability, its openness, its robustness, and its ease of use (Gollu, 1995).

The advantages and capabilities of object-oriented systems include: (a) The representation of elements in the object-based environment can be categorized into a hierarchy of object classes. The ability to perform various functions on different classes through polymorphism can further differentiate network elements and modal choices in ITS applications. This ability is not readily available without adding additional structure in the existing relational GIS. (b) An object-oriented representation will be more congenial to the perceived travel decision environment of an individual. It will therefore provide a more intuitive and user-friendly environment for the implementation of an ITS. (c) It overcomes the difficulties of the planar data structure and will allow for a finer differentiation of various geographic objects in the system. Running routing and spatial search algorithms will become less problematic. (d) It will facilitate the representation and processing of a multi-level transportation network through introducing new classes across different levels and new functions for these classes, spatial search and queries will be handled more efficiently. (e) It will facilitate the integration of a wide variety of geographic objects and relations within a comprehensive geographic database.

There are also important advantages when implementing such object-oriented systems in the real world: (a) Less costly data integration - Because of the costs of acquiring and maintaining geographical data, the cost of developing an ITS can be greatly reduced if part of the data can be shared and integrated across many other applications. An object-oriented approach can greatly facilitate this through a unified scheme of object abstraction and classification. (b) Less costly maintenance and expansion - The modularity of object-oriented systems renders them highly extensible, reusable and maintainable. (c) Higher data access efficiency and reliability of the system.

The major objectives when building an object-oriented system include the establishment of an object-oriented data model through constructing the key abstractions, class structure and functions that are required. Thus, we suggest that it is possible to build up an object-oriented GIS for ITS applications by generating a set of high-level abstractions of the ITS environment through domain analysis. These abstractions may include the transportation network, the mobile and non-mobile users of the systems, and the activity schedules and the adjustment strategies of travelers. Key mechanisms involved in the system would include processes like data acquisition, message parsing, activity scheduling, routing, spatial search and information display. The class structure and the module architecture of the system will then be constructed based upon the above analysis. Specifically, elements of the complex transportation network (including routes that utilize various modes such as transit routes) at various spatial scales could be represented in terms of abstract data types supporting inheritance, polymorphism and dynamic binding.

**Experimenting with an Object-Oriented ATIS**

The first step in this experiment was to conceptualize the class hierarchy that could be applied to an ATIS, and develop an object-oriented GIS that could handle inheritance, polymorphism and dynamic binding. We then developed a script language associated with this object system using C++ and C language, it can handle various mathematical...
and conditional statements, as well as defining classes and functions. The system is designed to be fully functional within the IBM PC environment.

Based on this preliminary work, we tried to utilize a real-world transportation network and operate it in our object-oriented system. We used an ETAK database of the Santa Barbara area as the primary host for our efforts. This is a comprehensive database available commercially.

Our first step was to read the ETAK mapbase file structure for the Santa Barbara network into our object class hierarchy. Next, we undertook the following steps: (1) completing modification of the ETAK database to include attributes such as travel times and splits, weights or penalties on turns at signed or lighted intersections, one-way streeting, lane changes, and temporal summaries of traffic volumes, and (2) expanding the class hierarchy in order to accommodate these realistic traffic characteristics.

Methods

A. Layering:

The first task was to divide the base map of the Santa Barbara area into layers. Most of the layers can be defined according to the attribute types defined within the ETAK MapBase format. Each L-record attaches information to a point, line, or region and has a code which determines the type of attribute. A layer is created for each type of attribute; and each layer contains lists of the following types of entities: circles, lines, points, polygons, nodes, links, and line attributes. The first four are the graphical entities common to most CAD and GIS systems. The nodes and links are the entities used to define the street network and will be described below. The line attributes are records used to describe the line and link entities. They contain a subset of the information attached to each line element within the MapBase format: they include the left and right street addresses, census block numbers, topological information, etc.

Because the MapBase format defines many more attribute types than actually occur within the Santa Barbara County Map, most of the layers ended up empty. An important work that was to add some of the detailed attributes in the ETAK database and build the layers. Transit routes were stored in another layer with pointers referring to the street network in order to perform routing. Space will be conserved by allocating only the entities which actually contain items in order to minimize storage.

B. Partitioning:

The major purpose of partitioning is to enhance the efficiency of spatial search. Some means of partitioning the map is required if displaying the zoomed-in map takes nearly as long as displaying the entire map. The reason is that storing the map as a "flat" file requires comparing each entity to the viewing box. The extent of the viewing box is determined by a translatable origin and axes scale factors. This problem can be alleviated by partitioning the contents of each layer. Partitioning allows the rendering of only those sections of the map which overlap with the viewing box without "touching" all of the entities. The increase in performance resulting from accessing only the necessary entities can be observed whenever a spatially localized subset of the map is to be referenced. This
will occur not only during the tasks of displaying the map but during many other analyses, including route computation.

The description given above of a “layer” stated that each layer contained a set of entity lists, one for each type of entity. The partitioned entity lists replace that description. Each layer consists of either the lists of entities or a set of partitioned entity lists. A hierarchical spatial tree can be used to implement the partitioning. The tree consists of a set of hierarchically arranged bounding boxes. The bounding box for the entire map is used as the boundary of the “root.” In a sense, the unpartitioned entities are stored at the root of the tree because all of the entities are contained within its associated bounding box (by definition). The root bounding box is divided into N-by-N quadrants to define the 1st level. The original box is the “parent” and the smaller N-by-N boxes are the “children.” Each child is itself subdivided into N-by-N boxes. This recursive subdivision is repeated until a specified “depth” is achieved. The bounding boxes at the first level define the tree’s “leaves.” The contents of each layer is partitioned by deciding which leaf’s bounding box contains the entity. This decision is made using the entity centroid. The entity is copied to the appropriate entity list in the leaf and is subsequently removed from the unpartitioned list. Although it is possible to use a different spatial partitioning for each layer, the same spatial quadtree is used for all layers.

In the first phase of building the system, only the leaves of the tree will be used to determine whether to draw each partition’s entities. The more sophisticated and efficient approach (to be pursued later) would be to compare the bounding boxes at each level of the tree to the viewing box. A saving in processing time would arise whenever the remaining depth of the tree could be ignored because the bounding box at the given level did not overlap the viewing box.

C. Hierarchical layering:

We require the capability to hierarchically relate different road types for use in route selection. In a multi-modal situation, we also require the different layers of routes to be related. Code can be written to create these relations at the “layer” level. We build on the recent work of Gollu (1995) for this purpose. The first step is to define a “road-layer” class and then to “derive” more specific types of roads from this class. “Derive” is the object-oriented term meaning to establish a “kind-of” or “parent-child” relation between objects (Egenhofer & Frank, 1989). The derived or “child” class is said to “inherit” the properties of its “parent” class, including the parent class’s attributes and its associated functions. A relation might be established between a class “car” and a class “vehicle.” It could then be said that “car is derived from vehicle” and that “a car is a type of vehicle.”

The class definitions are stored in text files which, when processed, define the layer hierarchy. The class definitions describe both the hierarchical relation as well as the attributes attached to “instances” of each class. In our case, the “instances” correspond to items placed into the given layer. Defining the layer hierarchy is taken care of by the first line, “class road . layer.” It is useful at the same time to define the potential attributes of entities contained by the layer. The attributes are only “potential” because the entities are not required to have defined attributes.
D. Associations between entities and attributes:

There seem to be several approaches to associating entities and attributes, each having different drawbacks. One is for each entity to contain a pointer to its attribute. When the entity possesses an attribute, the pointer would be set to the attribute’s memory address; when the entity does not, the pointer would be set to NULL. A second method, similar to the first, would be to place a pointer to an entity within each attribute. A third method would be to assign a unique identification code (i.e., "handle" or "object identifier") to each entity. Attributes would then refer to their entity using this code. A single-bit flag might be set within an entity to indicate that an entity possesses an attribute.

The first two of these approaches are pointer-based. They have the drawback that they allow only a single attribute to be associated with each entity (unless a more complicated scheme is implemented to allow lists of attribute pointers for each entity). One or the other approach would be preferred depending on from which side most of the computation would be done. If most computation involved only the attributes with infrequent reference to the entities, then the pointers might be better placed within the attributes. An additional consideration is whether the entity-attribute relations will be sparse or dense. The space consumed by NULL pointers under sparse conditions might be unacceptable. For the third method, actually associating an entity with its attribute requires searching the attribute list for the attribute having the entity’s identification code. This required computation is what is made implicit within the pointer-based approaches.

E. Generic attribute hierarchies

The capability to define class hierarchies is not specific to layers, but may also be used to define hierarchies of any information (e.g., types of activities, types of destinations). The use of the word “layer” in the road-class definition causes some special handling that creates internal data structures related to the map. Another way to say this is that the keyword “layer” acts as a built-in “base” class. The trick will be to decide what we want hierarchies of, what the attributes should be, and how we will fill in the attributes.

Theoretical and Applied Implications of Using OOGIS in ATIS

Transportation Science is host to a variety of theories concerning (among others) network structure, routing algorithms, traveler activity patterns, mode choice, demand forecasting, vehicle or traffic assignment, trip allocation, and traveler behavior. Our paper today has direct relevance for trip allocation and prediction, activity scheduling, and traveler behavior, and indirect relevance to other theories.

Most transportation models derive basic assumptions from econometric models. These include assumptions of economic rationality and utility maximization of travel related choices. Simplified spatial assumptions include those of shortest-path route selection and selection of closest alternatives for interaction purposes. In addition, travelers are assumed to be low risk takers, creatures of habit (i.e., invariably repeat behaviors), and to have perfect information. Such assumptions are often necessary to produce normative or optimal solutions for transportation problems.

In practice, information is usually partial or incomplete, is not available instantaneously, and may be difficult to interpret or integrate into an existing knowledge
Knowledge bases are imperfect, as are our mental models (i.e., cognitive maps) of environments through which travel takes place. Environments are partially represented. Drivers often have well formed habits that are difficult to change; these are easy to model. But much driver behavior is not habituated, but is better described in terms of problem solving. Driver decisions may be rational, but often we don’t know the appropriate criteria to judge them. Consequently, we limit those criteria to a few that are mathematically tractable. Sometimes we develop theories or models and assumptions that are sensible for dealing with freight traffic, or for dealing with fleet vehicular movements that have little or no driver independence involved, then assume that individualized vehicular traffic can be likewise constrained.

When discussing ATIS, we must remember that we are dealing with individual drivers pursuing their own activity schedules, rather than fleet operators seeking cost minimization solutions to driving problems. There is, therefore, a need to work in a more realistic rather than a more abstract and simplified environment; there is a need to incorporate more realistic assumptions of driver behavior and even network structure, there is a need to consider individual differences in risk-taking propensity and stated preferences for travel. Allowance must be made for both active movers (those seeking solutions to traffic problems) and passive stayers (those waiting *in situ* for someone else to solve the problem).

A viable ATIS must be more realistic than normative. As such, it is difficult to imagine a good usable and practical ATIS being driven by existing theoretical principles or concerns. The potential for developing a real-world real-time ATIS may be significantly increased by using a GIS as a host.

A host GIS can allow different levels or classes of objects. Starting with the most basic feature - the transport network - a GIS can share an accurate geocoded database complete with curves, gradients, non-planar intersections, and movement controls, or it could develop cognitively warped and viewpoint-specific systems. This introduces versatility into the network representation.

A GIS should provide alternative path selection algorithms with a variety of route selection criteria (e.g., shortest path, fastest path, longest leg first, most aesthetic route, etc.). Such criteria may be most important in selecting alternate routes and alternate destinations after experiencing congestion, hazards, construction, or other travel delay scenarios.

By more faithfully and realistically encoding routes and linking sets of attributes to them, rerouting and changing destinations should become more acceptable procedures. There is little sense in recommending an alternative route that is restricted to local streets with numerous stop signs, if a traveler’s criteria for path selection is “fastest route.” It makes little sense to recommend an alternative route that crosses many arterials if a traveler wishes to avoid traffic signals or wants to minimize left turns. Most ATIS to date provide only one alternative routing model (often a modified Dijkstra algorithm), many of today’s GIS offer a set of alternative route selection models which allow individuals to express their preferences for different controlling criteria.

Object-oriented GIS in particular allows for detailed attributes to be attached to routes. Routes can be conceptualized as continuous lanes over the entire length of a street, rather than as multiple arcs with beginning and ending intersection nodes. And, as
shown previously, features such as hierarchical layering, class structures, inheritance of attributes, polymorphisms, and encapsulation processes give much versatility and considerable replication of how we perceive the reality of transportation systems.

Thus, as technology (such as GIS) allows us to model and describe transportation systems more completely, there emerges a need to re-think existing models and theories which were developed in times of sparser knowledge and understanding of both system complexity and human behavior. While not denying the continued usefulness of (normative) theories and models as ways to formalize the state of our knowledge, there is also little doubt that as we enter this new age of information distribution, new sets of constraining assumptions are needed. As we develop more comprehensive tools for obtaining environmental information (e.g. remote sensing, satellite imagery) our need to simplify for the sake of mathematical elegance may be reduced. And as we develop technologies for detailed examination of driver activities and behavior (e.g. traffic and driver simulators) our knowledge of the range of feasible and likely behaviors should escalate - again requiring revisions of limiting behavioral assumptions.

Thus, as we explore the use of current and future technology, there should be direct and immediate feedback to the domains of theory and model building.

Conclusion
We expect that this research will have important significance on both basic and applied levels. We have conceptualized and developed an object-oriented geographic information system. Our continuing effort will focus on the data modeling issues of a multi-modal network, and the implementation of a multi-strategy travel decision support system built on this object-oriented system. We have discussed the advantages and disadvantages of an object-oriented approach to transportation modeling. Apart from its improved capability of handling the transport network as more of a perceptually accurate system, the object-oriented data model allows us to incorporate hierarchical layering within the basic network that ties to the normal engineering way of interpreting road systems. Polymorphism allows us to perform ITS functions on various classes of objects in the transportation network, an ability that is not easily obtained in existing systems. The approach also appears likely to substantially decrease the time involved in interacting with the database particularly by using partitioning and inheritance characteristics. Cost of operation should consequently be reduced. Both these factors are important when considering the primary aim of the ATIS component of an ITS is to get useful information to travelers in as timely a manner as possible so that on-route decision making can be undertaken.

There are many basic research problems relating to the development of workable object-oriented data models for use in transportation planning and this research has examined some of these. We also expect that the data model as developed will greatly facilitate the implementation of ITS by more quickly resolving conflicts with respect to ultimate selection of routes, substituting destinations, changing activity patterns, and rescheduling activities.

Our project was designed to contribute to the next generation of traffic management technology, particularly in terms of dispensing information to travelers in a pre-planning or on-route phase via an ATIS. ITS generally appears to be moving more towards
Object-Oriented data structures and models and we believe our work is in line with these nationwide trends.

Some of the critical features involved in evaluating the worth of any object-oriented data model or object-oriented data structure include its ease of use, the relevance of the attributes defined in the system, whether or not the model deals with real or artificial concepts, the degree to which there is a clear translation between model entities and the actual objects, and whether or not there are acceptable matches between those activities undertaken in the real world and those activities incorporated into the model. Other criteria relate to the number of modules embedded in the system that have to be changed in order to work in a real environment, the number of steps that operation of the system requires, the degree to which one must know and accept a process model of the system, and the time that is required to integrate changes in the system to ensure it is a dynamic real time one.

References

Adler, J L, Recker, W W, and McNally, M G (1992a) A conflict model and interactive simulator (FASTCARS) for predicting enroute driver behavior in response to real-time traffic condition information Working Paper, UCTC No 127 The University of California Transportation Center (UCTC), University of California at Berkeley.


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


Koutsopoulos, H N., Polydoropoulou, A, and Ben-Akiva, M (1995) Travel simulators for data collection on driver behavior in the presence of information *Transportation Research C*, 3, 3 143-159


Section VIII

A Review of Object-Oriented Approaches in Geographic Information Systems for Transportation Modeling

Submitted to *Transportation Research, Part C*
A Review of Object-Oriented Approaches in Geographic Information Systems for Transportation Modeling

Mei-Po Kwan
Department of Geography
Ohio State University
Columbus, Ohio 43210-1361

Reginald G. Golledge
Department of Geography
and
Research Unit in Spatial Cognition and Choice
University of California at Santa Barbara
Santa Barbara, California 93106-4060

Jon M Speigle
Department of Psychology
University of California at Santa Barbara
Santa Barbara, California 93106

Please send all correspondence to the first author.

Acknowledgement: Funding for this project was provided by UCTC Grant # DTRS92-G-0009
Abstract:

The objective of this paper is to review object-oriented (OO) approaches to data modeling and data handling and their usefulness in transportation planning and modeling in general and Intelligent Transportation Systems in particular. The paper begins with a discussion of the current GIS data model for representing a transportation network and the most common database management systems used in the context of transportation planning. We then discuss object-orientation, its different properties and its usefulness in representing a multi-modal, multi-scale network with hierarchical road types. We also discuss alternative database management schemes. Finally we review some existing systems and discuss the implications of adopting an object-oriented perspective.
Purpose

Intelligent Transportation Systems (ITS) utilize advanced communication and transportation technologies to achieve traffic efficiency and safety. The different components of ITS include Advanced Traveler Information Systems (ATIS), Automated Highway Systems (AHS), Advanced Traffic Management Systems (ATMS), Advanced Vehicle Control Systems (AVCS) and Advanced Public Transportation Systems (APTS) (Figure 1). Development of a system for ITS critically depends on our capability to incorporate a vast amount of information about the locations of features and to maintain a complex representation of the transportation network for various activities within a geographic database (Watling and Vuren, 1993; Schofer, Khattak, and Koppelman, 1993; Kwan, 1994, Kanninen, 1996). The system therefore needs to be constructed based on an integrated and comprehensive Geographic Information System (GIS) (Worboys, 1992, 1994; Tomlin, 1990, van Oosterom and van den Bos, 1989; Wiegand and Adams, 1994).

Unlike the simplified planar graph theoretic, transport network representations used by current ITS, GIS are able to provide a more realistic representation of elements of the complex transportation environment. As such, there is considerable potential for GIS to become prominent for constructing a real-time multi-strategy travel decision support system as we conceive it.

[Insert Figure 1 about here]

The first ITS requirement which is met by a GIS is the capability to represent the transportation network in detail sufficient to perform different network algorithms, modeling and simulations. In the real world, a transportation network has different types of roads, (e.g. highway, arterials, local streets, etc) and different types of intersections (e.g. signalized and non-signalized) that are of interest to ITS builders. For some applications, information on intersections, lanes and lane changes such as turn lanes and
merging lanes, highway entrances and exits, etc., is also important. Other applications may require geometric representation of road curvature and incline.

Current GIS data models usually represent a network as a collection of links with nodes created at the link intersections. Unless additional structure is added, the link-node nature of the current commercially available topological data model is basically planar and cannot distinguish an intersection at grade from an intersection with an overpass or underpass which does not cross at grade. The difficulty in accurately representing overpasses or underpasses may lead to problems when running various routing algorithms (e.g., recommending that a traveler make a left-turn at an intersection that proves to be an overpass).

There are other situations in which we may want to differentiate different types of nodes. In a link-node structure, a node is created when there is an intersection. However, when the traveler needs to make a decision to move along the network, there are many decision points that they need to encounter in addition to intersections (like a barrier along a street). Sometimes, different nodes may lead to different behaviors. For example, we usually make a U-turn at a dead-end node or a barrier. Currently existing software does not handle this differentiation. An object-oriented approach is thus promising in this aspect.

A second ITS requirement is that, since ITS applications have to operate at various spatial scales, a multi-level representation of a transportation network is needed. At the local level (i.e., city or county), the transportation network needs to be represented in very great detail for navigational purposes, whereas only the major elements such as the interstate or state highways need be represented for travel at the regional, state or interstate levels. Further, relationships between data at different scales have to be established. Only through the use of a more efficient geographic data model can we hope to overcome these and other related problems (van Oosterom and van den Bos, 1989; Roberts, Gahegan, Hogg, & Hoyle, 1991). Several recent attempts to tackle such
problems within transportation modeling have focused on the object-oriented approach (e.g. Mainguenaud, 1995; van Oosterom and Schenkelaars, 1995).

A third requirement is that ITS must handle the real-time updating of the traffic patterns and transmission of information. To provide travelers with timely decision support, an Advanced Traveler Information System (ATIS) needs to receive and transmit data in real-time through a communications network which connects a vast number of system elements, including traffic sensors and location-tracking devices. Even for a mid-sized city like Santa Barbara (approximately 180,000 people and 13,000 segments in its transport network), the real-time update and transmission of data is already a very challenging problem. To keep information on the dynamic environment current and self-consistent even in the presence of concurrent updates and queries from around the network, we need to go beyond the existing data handling technologies of current ITS and GIS. The system will likely be distributed and will require spatial partitioning tailored for effective access.

To meet these requirements, we explore here the use of object-oriented data modeling and database technologies. Below, we review the object-oriented approach in relation to transportation planning and modeling, examine and compare object-oriented systems with others, and discuss the implications of adopting an object-oriented GIS.

Object-oriented approach - An overview

General

Object-orientation (OO) is a technology that integrates various programming and modeling techniques for analyzing the world. An object-oriented model is a collection of discrete objects, their associations and their interrelationships. An object-oriented model also is an enumeration of the ways in which a system transforms its values and an
elaboration of the timing, sequencing and control of events. It is a methodology which formalizes many of the operations performed in any modeling exercise.

Object-oriented modeling has been reviewed and discussed extensively (e.g., Rumbaugh, Blaha, Premerlani, and Lorensen, 1991, Martin, 1993, and Booch, 1994) as well as in several texts specific to database systems (Khoshafian & Abnous, 1990; Kim, 1990, Gupta and Horowitz, 1991, Khoshafian, 1993, Loomis, 1995). The major concepts are well-documented in these books and reports. Major constructs of the OO approach, including object identity, encapsulation, inheritance, generalization, specialization, aggregation, and polymorphism, are discussed also in this literature and in an increasing number of GIS related journal articles (e.g. Worboys, 1992, 1994b, Worboys, Hearnshaw, & Maguire, 1990, Egenhofer and Frank, 1989). We summarize this material below. Although we will use OO terminology, some of the concepts predate the development of object oriented methods while others are particular to it.

Object identity

Object-oriented models look at the real world in terms of tangible objects and how they behave. For example, the transportation network can be represented as different objects such as highways, intersections, and ramps. Each object has its own methods which specify the different manipulations possible by an object of that type. A message or request sent to an object will invoke the object’s methods. For example, calculating the shortest travel time between two locations may be represented as a method of a street network object.

Note that existing GIS usually represent the transportation network in terms of links (i.e. road segments) with nodes (i.e. street intersections). This is called the link-node structure. Each line segment typically has a unique identification. A relational database such as ARCINFO may be used to link the spatial objects with other attributes. In such a database, each record is distinguished by the data values it contains. Its identity is not
easily retained if some of its properties change over time (Korth and Silberschatz, 1991). In the object-oriented system, each object corresponds to an entity in the modeled world. Its identity is retained even if its properties change over time. For example, if an object migrates several stages (i.e., change as the transportation network changes), it will still keep the unique identification through versioning. As such, we say that the object identity (OID) is persistent. We can use versioning to represent the changes of, for example, census tracts over time (Worboys, 1992).

Classification

Objects can be classified into different object types. A type description generally consists of a description of the attributes and methods supported by objects of that type. A type may also have additional attributes which are not associated with any individual object but with the type as a whole. The type description is also referred to as defining the interface of the type. An object class is a software implementation of an object type. A type may be represented by several classes. Classification in object models is an abstraction method and models the real world in terms of a hierarchical taxonomic structure. Basically all of the following OO concepts concern aspects of classification.

In the object model, a class can be further divided into subclasses. Elsewhere, we have experimented with a system that consists of a spatial object hierarchy which examines the network as a hierarchy of point, polyline, polygon and polyline list objects (Kwan, Speigle and Golledge, 1996). Figure 2 details the conceptualization of a hierarchy for the subclasses Node and Link to represent the transportation network. In Figure 2 we graphically depict a subset of the basic elements of a vector-based, transportation model. We use a subset of the notation of Rumbaugh and colleagues’ Object Modeling Technique (OMT) (1991). Each box represents a different class and has three sections: the class name, class attributes, and class methods. Attributes are indicated either graphically or by listing their labels and types in the section below the
class name The lines connecting different classes represent relationships between those classes. The type of the relationship is indicated by the symbols at the ends of the attaching lines. Inheritance is indicated by a triangle connecting the superclass to its subclass. Aggregation is indicated by a diamond on the side of the class which contains the aggregated attributes. These relations are labelled with the name of the aggregated attribute. A one-to-one relationship is indicated by an unadorned line; a zero-to-many relationship is indicated by a circle.

The Point and Polyline classes are derived from the base class Spatial. Spatial defines the interface supported by all derived classes. These methods are abstract because they are not actually supplied by Spatial, but are supplied by the derived class. The class Node is derived from Point, and class Link, from Polyline. With the exception of "turn costs" for Node, the attributes are indicated graphically. A Link includes references to a pair of Nodes, in addition to the list of intervening vertices inherited from Polyline. We can further model the real-world Node as Stop and No-stop Node subclasses as in the real-world transportation network (Figure 3). The Link class may be further modeled as Highway, Interstate and Addressed road types (Figure 4).

[insert Figure 2 about here]

[insert Figure 3 about here]

[insert Figure 4 about here]

Generalization

Several classes of objects, which have some properties and operations in common, can be grouped by forming a more general super class. Each subclass would describe a specialization of the super class. This concept is especially important for the abstraction.
of a real transportation network. For example, the object type Highway, Interstate and Addressed can be grouped to a super class Road (Figure 4).

Association

A relationship between similar objects can be grouped together to form a higher level object (Brodie, 1984). The term set is used to describe the association, and each associated object in the set is called a member (Egenhofer and Frank, 1989). This is important to GIS because spatial relationships may be modeled as associations. In the transportation context, an association can represent the relationships between network elements and other spatial objects. For example, a highway may be inside a city and together this forms a set (highway in city). An ordered collection of intersections can be grouped to form a road and this is called an ordered association.

Aggregation

Several objects can be grouped together to form a composite object, creating a higher-level object which may itself be referred to. The model of this process is called aggregation. For example, the object type Highway is an aggregation of lanes, road sign positions, exit entrance and ramps, and so on.

Details of objects are suppressed in the constituent or aggregated objects. Every instance of an aggregate object can be decomposed into instances of the component objects while keeping its own functionality. However, the operations on the aggregate may not be compatible with operations on its components (Egenhofer and Frank, 1989).

Inheritance

Inheritance is a major-type/sub-type relationship. It is intrinsically related to generalization. Inheritance is the mechanism by which a type inherits properties (data structure and methods) of its super type. A type Road may represent the generalization of arterial, collector and highway. The arterial, collector and highway may be considered specializations of Road. The supertype/subtype relation is indicated in the
subtype definition. The subtype is said to be *derived from* or to be the *child of the parent* type. For example, a highway derived from Road would *inherit* the properties of Road (Figure 4 and 5). By being a subtype of Road, it has the attributes of length, number of lanes, speed limit, direction and divider type. Inheritance is one of the most powerful and distinctive characteristics of object orientation. The act of inheriting properties from more than one super type is called *multiple inheritance*.

Polymorphism

Polymorphism generally represents the quality or state of an object, which is able to assume different forms. In a programming context, polymorphism is the mechanism by which the same sequence of programming statements can be used to manipulate objects of different types. One form of polymorphism is called overload. With overload, the same operator (e.g., the "addition", +, operator or the "print yourself" message) can perform different functions depending on the class of the object receiving the message. It is a very important feature for transportation modeling. For example, the link-node structure assumes that the transportation network is represented as links and nodes, and there is no differentiation between these links and nodes. The delay function in this type of system cannot be easily modeled differently for signaled intersections versus non-signaled intersections, or for highways versus local streets. Polymorphism within an object-oriented context allows these different network elements to be modeled using different functions, but with the same, overloaded operator representing the delay function. Khoshafian and Abnous (1990) listed the advantages of overload in terms of programming: (1) extensibility - the same algorithm or function may be applied to instances of many classes (some of which may be defined in the future) without modifying its code; (2) development of more compact code because there is no need to
list out all the situations for different classes; (3) clarity - the code is more readable and comprehensible. A penalty is paid for the run-time binding of messages to functions which depend on an object's type. Nevertheless, specially designed constructs such as hash tables and indexes have been developed to lessen the performance penalty of dynamic binding (Khoshafian and Abnous, 1990).

In some languages, parametric types can be used to define a class's methods. *Parametric polymorphism* allows the construction of several functions of the same name with type parameters which distinguish them. For example, a delay function can be formulated as COST[D] with D as a parameter of different types of distance measure. In transportation research, distance can be measured as either travel time or actual distance. So the parameter D could be either a type of integer (in terms of minutes of travel time) or a type of real number (in terms of meters of physical distance). Different procedures may implement the same functionality for different argument types. In OO languages with a separate *compile* phase, this type of binding occurs at compile-time while overloading polymorphism occurs at run-time. Thus, one of the strongest advantages of parametric polymorphism is code sharing for generic types with the power of strong typing.

**Encapsulation**

Good OO designs enforce encapsulation and information hiding. The state of a completely encapsulated object can be manipulated and read only by invoking operations that are specified within the type definition. For example, within an object-oriented GIS, we can implement a quadtree spatial partitioning structure (Kwan et. al, 1996) but this structure can be easily replaced with an R-tree (Guttman, 1984) or a combination of a splay tree and a quadtree structure (Cobb, Chung, Shaw, and Arctur, 1995) based on the data needs for faster access. Replacements, such as these, are made easier by encapsulation. Encapsulation leads to a design which is modularized at the level of the
object classes. Implementation changes will not ripple outside a class's member functions as long as all access to an object's state is by means of its public interface. In some languages (e.g., C++) the class definition may also include private methods which are used internally by the class but which are not accessible from outside the class.

**Types of OO**

When we talk about object-orientation, we accept that the concept may involve different dimensions. OO includes modeling, programming, and database management. One widely held view is that the object-oriented approach to database management allows a more natural integration of the database with the object-oriented programming language used to manipulate it (Loomis, 1993a, Loomis, 1993b, Loomis, 1995)

a. **OO modeling**

OO models the real world as closely as possible (in the chosen code). It is a conceptual tool that can be used in virtually any problem domain. OO modeling refers to the development stage in which a complex system is described by a set of abstract types on which certain behaviors are defined. Unlike other modeling techniques which focus on functions, OO focuses on the essential constructs within a problem domain. The OO model views the world as objects, and starts with objects (e.g., lanes, intersections) with distinct attributes (e.g., directionality, signalization) and behaviors (e.g., linking segments or lanes and intersections to make an O-O trip). The method/message metaphor is similar enough to natural language and human problem solving that the mismatch between the conceptualization and specification is minimized.

Traditionally, transportation networks have been modeled as a planar graph (Guting, 1991). The disadvantages of a planar network include: (a) not being able to handle multiple representation levels, and (b) not handling non-planar junctions. An additional problem not specific to planar networks is that many models do not allow the network to
be a mixture of one and two way streets. It is argued that object-orientation can
overcome some of these limitations. Mainguenaud (1995) modeled the logical graph
topology using an object-oriented approach to allow the handling of multi-level networks.
The existing digital network databases do not distinguish a local versus a regional
transportation network. Mainguenaud introduces a class hierarchy where two classes are
derived from the basic Node/Edge classes, called the MasterNode and MasterEdge
classes. These classes override the basic functionality of the Node/Edge classes and
allow the Master versions to be abstractions of sub-networks. This approach may be
contrasted with other conceptualizations of multi-scale such as those dealing with map
display (Cobb, Chung, Shaw and Arctur, 1994). In another paper, we have experimented
with an object-oriented system to overcome the problems of a planar network and a
network with mixed one-way and two-way directions (see Kwan, Speigle and Golledge,
1996).

b. OO programming

An OO programming language provides direct support for the OO modeling constructs.
As such, the gap between system requirements and system implementation is reduced.
Several different aspects of OO increase the ease of revising existing code (Carroll and
Ellis, 1995). The primary mechanism is inheritance. Another mechanism is aggregation
combined with delegating messages to an object's components. A third mechanism
supported by some languages (e.g., C++) is that of parameterized types. A parameterized
type or template class is a class which accepts as an argument in the type definition the
name of another class. The template class is used to represent an algorithm (e.g., a List or
Array storage class) which manipulates objects of an arbitrary type.

OO is also hailed for its ability to provide the abstractions which allow the programmer to
conceptualize more difficult problems than would otherwise be possible. The levels of
abstraction and encapsulation allow the programmer to design a system in terms of a set
of interacting types rather than being mired in the specifics of all of the specializations of the different types.

c. **OO Database Management Systems (OODBMS)**

Selection of an affordable Database Management System (DBMS) is very important in handling transportation data, especially in view of the increasing size of digital databases. An object-oriented DBMS (OODBMS) supports the object-oriented paradigm and stores the instances of classes. The systems support efficient storage of arbitrarily complex objects whose attributes are formed by inheritance and aggregation. A standard query language, however, is a topic of current research.

The limitations of existing DBMS, especially relational DBMS for storing geographic data, have been discussed by Wiegand and Adams (1994), and Kemper and Moerkotte (1994). Wiegand and Adams (1994) began with the argument that relational models do not easily handle the complexity of geographic data and a better GIS can be built by (1) using a DBMS as a base for the system, (2) using object-oriented modeling, and (3) having an extensible system. They used an actual OODBMS and transposed current relational GIS into an OO model.

A survey of commercially available OODBMS was recently given by Kemper and Moerkotte (1994). They include Gemstone, O2, Ontos, Itasca, Versant, Matisse, Objectivity/DB, and ObjectStore. The data model, control concepts, architecture, related literature, and performance enhancements are discussed. Wiegand and Adams (1994) examined several commercial OODBMS and extended-relational systems such as POSTGRES, Starburst, O2, Objectstore, and Gemstone. Advantages of OO according to their research include multiple representation (e.g., different scales), multiple geometries, and an increased amount of non-spatial information that can be stored within the feature object. Relationships between feature objects can be directly stored as part of the model.
rather than via an indexing key. These points will be expanded upon in a later section in which OODBMS will be compared with relational database management systems (RDBMS) implementations.

Towards an Object-oriented GIS

Worboys (1992a & b, 1994a & b), and Worboys, Hearnshaw, & Maguire (1990), surveyed the state of the object-oriented paradigm as it applies to the handling of geo-referenced information. They outlined the major concepts behind the approach and its application in handling spatial information. These concepts have also been presented in the majority of the pure research papers, some of which will be discussed below.

Worboys (1994b) defined a geo-object, which conceptually unifies spatial, temporal, graphical and textual/numerical objects. He also pointed out that there is little use of proprietary object-oriented systems, except in cases like Milne, Milton & Smith (1993) and David, Raynal, Schorter & Mansart (1993). Research on the extended relational system has also faced difficulties. Projects in extended relational DBMS on applications of geo-referenced information include work by Lohman, Lindsay, Pirahech, & Schuefer (1991), and Rowe & Stonebraker (1987).

Some object-oriented systems have been implemented using object-oriented modeling and commercial OODBMS. Williamson & Stucky (1991) developed a generic GIS supporting Earth resource imaging analysis. The system consists of (1) a graphic, raster and text interface; (2) a database containing maps, images, and graphical and textual descriptors; and (3) a collection of processes which transform the representation and content of the data objects in the GIS. The system purportedly improved quality of reports, database updates, and analysis and allowed more timely access to widely dispersed information. They argue that a more intuitive interface with the information in the database also reduces the necessary minimum training level for analysis.
Milne et al. (1993) discuss the construction of an OO GIS based on the general purpose OODBMS, ONTOS, and use the Spatial Data Transfer Standard (SDTS) to provide a model for the basic spatial object classes. They compare the performance of ORACLE and SIRO-DBMS with their extended ONTOS system and report an impressive performance gain. David et al. (1993) describe the implementation of an OO GIS using the OODBMS O2. They distinguish between mode (semantic geographical model and localization model) and structure (spaghetti, network, and maps structure). Scholl and Vossard (1992) implemented a spatial, relational-like query language on top of the OODBMS O2.

Acceptable performance from an object-oriented spatial database can be achieved only through the use of spatial indexing and clustering. These issues are receiving an increasing amount of attention in the object-oriented database literature (see Salzburg, 1994). Spatial indexing provides a way of locating objects based on spatial criteria. The numerous schemes which have been proposed include point and region quadtrees (Samet, 1989), R-trees (Guttman, 1984), and K-d trees (Bentley, 1975). These schemes are essential for avoiding a serial search of the entire database when handling queries which include spatial conditionals. Clustering refers to the actual layout of a set of objects either on the permanent store or within the server-side/client-side object managers. A general discussion of the issues in clustering an object-oriented database, as well as the implementation in O2, may be found in Benzaken and Delobel (1990). The basic goal of a clustering algorithm is to place together entities which are accessed together. Implementing these schemes within an object-oriented database is complicated because both relations between objects must be taken into account as well as the class hierarchy. Full implementations would integrate the indexing completely with the query processor. An efficient indexing implementation would also be mirrored in the clustering scheme.

Many systems rely on an object-identifier scheme for retrieving objects from the database. Unless the object identifiers (OID) also indicate in some way the spatial
position of the object, this retrieval could only be implemented using non-spatial indexing. One approach is to maintain a structure which maintains a current 'working' section of the spatial database. (Cobb et al, 1995) Searches based on OID would then proceed from the most-to-least currently used spatial nodes. Another possibility is to incorporate spatial referencing directly into the OIDs. (Mabuenaud, 1995) This approach, however, would become inefficient if the spatial indices were reorganized. The capability to allow dynamic restructuring is in fact one of the more complicated aspects of the problem. Because the different commercially available OODBMS allow a developer to customize the system's indexing scheme to different degrees and because the different systems incorporate different levels of support for spatial data, these capabilities should be an important component of any assessment for purposes of procurement.

Comparison of object-oriented and relational approaches

General

Typical approaches to modeling transportation processes have used relational databases for data modeling and as the base for programming interfaces. The object-oriented approach, however, has been said to have superior modeling power because of its ability to handle complex objects, behavioral data, meta knowledge and long-duration transactions (Korth and Silberschatz, 1991). New applications require the handling of a complex object that contains other objects within it. For distinct objects, they may also need to respond in different ways to the same command and thus require the handling of behavioral data. We may also want to handle meta knowledge since often the most important data about an application are the general rules about the application rather than the specific cases. Lastly, human interactions with the data become more and more important in CAD and CASE applications. Because of these interactions, there may be
more "what if" modifications in the applications. More long-duration transactions are thus necessary. Because of these requirements there is a tendency for the use of OODBMS to expand in comparison to relational models. While the latter models simply describe system states, object-oriented models can describe both system states for data and system processes or behaviors in an integrated context (Booch, 1991, Rumbaugh, et al. 1991).

Some limitations of existing DBMS (relational) have been identified by various writers such as Herring (1992) and Wiegand & Adams (1994). First, relational RDBMS lack extensibility to provide for special application needs (e.g., provisions for the user to add new data types and methods, addition of user-defined code, design of new storage methods, and access to user-defined code). A table is created for each entity type, a row corresponds to an entity and columns contain the attribute values. However, a relational schema ends up with many additional tables because an attribute is restricted to being a simple built-in type. The two most frequently cited criticisms are that an attribute may not be a set of values and that relationships are also modeled using tables. For a GIS's spatial and non-spatial data, the data are complex enough that spreading entities and attributes into numerous tables is undesirable (Wiegand & Adams, 1994). OO modeling allows related data to be kept together and for relationships to be directly modeled (Medeiros & Pires, 1994).

The new geographic data models based on a set of feature objects provide the needed framework for a scaleless and seamless database (Mainguenaud, 1995, Guptill, 1989). It also has the DBMS characteristic of extensibility needed by GIS (Haas & Cody, 1991). For example, the ability to add new data types (e.g., points) and operations on them (e.g., distance functions) and the ability to have a new set of operations as part of the query language (e.g., overlay) are added benefits (Gunter & Lamberts, 1994). With the OO paradigm's modularity comes the capability to easily interchange access methods such as grid-files (Nievergelt, Hinterberger, & Sevcik, 1984) or R-trees (Guttman, 1984) and to
support new data storage methods so that (e.g., a map) can be stored as a quadtree. Extensibility of the optimizer allows optimizing with different data types and operators. The ability to access user-defined code and standard packages is also very important.

The use of relational databases has been popular in the past and because of this, a set of reliable and working software tools have been developed. This software includes the core database engine, modeling tools, and application development environments. An RDBMS usually incorporates a powerful query language (SQL) which has a sound mathematical basis in relational calculus (Date, 1985), and usually operates on collections of fixed format tables. As opposed to this, the tools in an object-oriented approach draw on a semantically richer background. Unfortunately this has allowed greater personalization of the approach and a standard query language has not yet emerged (c.f., Catell, 1994). As a result, two different types of object-databases have developed, one of them following many of the ideas of the relational model and the other designed to be integrated with an OO programming language (such as VERSANT). The goal of VERSANT's type of OODBMS is to transparently provide persistence to the classes defined in the OO programming language. Ties to the relational model provide enhanced relational databases with an object interface. Those tied to VERSANT or other similar languages emphasize transparent persistence of objects. Other, middle-ware tools operate as translator between objects on the program side, and relational sets on the storage side. Regardless of the tools selected, there is a need for more application oriented technology. The development of this software, however, must focus on questions of modularity, performance, scaleability, openness, robustness, and ease of use (Gölli, 1995).

There appear to be a limited number of formal modeling methods relevant for object-orientation. These include finite state machines (FSM) (Özveren, 1989; Ramadge & Wonham, 1987), Petri-nets and data flow diagrams (Yourcen, E., 1989; Rumbaugh, et al. 1991; and Schlaer and Melor, 1988), the Calculation of Communication Sequences (CCS) (Milner, 1980), Communication Sequential Processes (CSP) (Hoare, 1985),

Gollü (1995) has reviewed some examples of these software frameworks including:

a. Ptolemy This is an object-oriented data flow based simulation tool (Ptolemy Manual, 1995) Ptolemy produces star maps in which objects called stars are represented by inputs, outputs, and pattern maps. The input output connections propagate messages that allow object interaction. Object evolution can be driven by some combination of time, events, or data tokens. It is most useful for specification and simulation of data flow among a static configuration of objects. However, it does not appear as suitable when large numbers of object relationships are concerned. It does not interface easily with other systems.

b. COSPAN This is a general purpose software tool that provides an automatic description syntax on a set of operations. The system combines automata in order to achieve coordination of operatives in a specified way. Its logical analysis consists of symbolic testing of a system for user defined behavior and in essence its analysis constitutes a mathematical proof of the stated system behavior (Har-El & Kurshan, 1987). COSPAN’s uses include the logic of discrete-event modeling in economics, and it has also been used for institutional strategic planning and epidemiological modeling in medicine.

c. CSIM. This is again a general purpose C-based process oriented environment. It is specifically designed to simulate a discrete event system and its most frequent use relates to the behavior of data communication networks (Schwetnam, 1989).

Advantages and Disadvantages of Object-oriented Systems

The advantages and capabilities of object-oriented systems include (a) The representation of elements in the object-based environment can be categorized into a
hierarchy of object classes. The ability to perform various functions on different classes through polymorphism can further differentiate network elements and modal choices in ITS applications. This ability is not readily available without adding additional structure in the existing relational GIS. (b) An object-oriented representation will be more congenial to the perceived travel decision environment of an individual. It will therefore provide a more intuitive and user-friendly environment for the implementation of an ITS. (c) It overcomes the difficulties of the planar data structure and will allow for a finer differentiation of various geographic objects in the system. Running routing and spatial search algorithms will become less problematic. (d) It will facilitate the representation and processing of a multi-level transportation network through introducing new classes across different levels and new functions for these classes; spatial search and queries will be handled more efficiently. (e) It will facilitate the integration of a wide variety of geographic objects and relations within a comprehensive geographic database.

There are also important advantages when implementing such object-oriented systems in the real world: (a) Less costly data integration - Because of the costs of acquiring and maintaining geographical data, the cost of developing an ITS can be greatly reduced if part of the data can be shared and integrated across many other applications. An object-oriented approach can greatly facilitate this through a unified scheme of object abstraction and classification. (b) Less costly maintenance and expansion - The modularity of object-oriented systems renders them highly extensible, reusable and maintainable. (c) Higher data access efficiency and reliability of the system, and (d) common use of the same database for different aspects of ITS - e.g. ATIS, ATMS, APTS.

Conclusions

The major objectives when building an object-oriented system include the establishment of an object-oriented data model through constructing the key abstractions, class structure and functions that are required. Thus, we suggest that it is possible to construct an object-
oriented GIS for ITS applications by generating a set of high-level abstractions of the ITS environment through domain analysis. These abstractions may include the transportation network, the mobile and non-mobile users of the systems, and the activity schedules and the adjustment strategies of travelers. Key mechanisms involved in the system would include processes such as data acquisition, message parsing, activity scheduling, routing, spatial search and information display (Kwan, 1994) The class structure and the module architecture of the system will then be constructed based upon the above analysis (Kwan et al., 1996) Specifically, elements of the complex transportation network (including routes that utilize various modes such as transit routes) at various spatial scales could be represented in terms of abstract data types supporting inheritance, polymorphism and dynamic binding.

Gollu (1995) argues that the validity of an object-oriented system model is likely to depend on our ability to validate the internal logic of the models, and the deployability of model component specifications. Further, he goes on to argue that the principal requirements for valid and usable software systems include:

- An ability to associate physical and logical representations (modularity)
- The ability to add new components to the system with minimal code rewrite (openness, modularity, robustness)
- The ability to collect arbitrary statistics during simulation (openness, modularity)
- The ability to run simulations with acceptable performance (performance)
- The ability to adjust simulation granularity (modularity, openness)
- The ability to simulate up to 100,000 vehicles (performance)
- The ability to specify system behavior in a straightforward language (ease of use) (Gollu, 1995, pp. 31,32.)

Although selection of a data model and database are likely to be context dependent (i.e., different potential users will have different sets of accumulated personnel skills,
software, and data records), there appear to be increasingly strong arguments in favor of object orientation as the most versatile system for transportation modeling and planning. OO approaches provide an opportunity for the modeler to use relevant perceptual and cognitive concepts (e.g., "streets" instead of "segments"), as well as to utilize the power of concepts such as modularity, inheritance and polymorphism. In addition, the ability to handle changes as "add-ons" with object ID maintenance rather than requiring system reregistration is a positive feature. Since object orientation is an emerging approach, we have tried to emphasize both its major underlying concepts and its probable strengths. As research and development proceeds, the development of a testbed network for evaluating the claims currently being made becomes of paramount importance.

References


Kwan, M.-P, Speigle, J, and Golledge, R.G (1996) Developing an Object-Oriented testbed for modeling transportation systems Discussion paper, Research Unit in Spatial Cognition and Choice (RUSCC), Department of Geography, University of California Santa Barbara


Loomis, M E S (1993b) Object programming and database management Differences in perspective between the two Journal of Object-Oriented Programming, May, 31-34, 67

Loomis, M.E S (1995) Object databases, the essentials New York Addison-Wesley,


multiformalism simulation in common LSL/CLOS. *Discrete Event Dynamic 
Systems Theory and Application*, 3, 2. 119-149.

Ptolemy Manual (1995) *The aimalimagest*, Volume 1-4, Version 0.5 2, College of 
Engineering, UC Berkeley.

Ramadge, P., and Wonham, W. (1987) Supervisor control of a class of discrete event 


oriented databases to geographic information systems. *Information and Software 
Technology*, 33, 1. 38-46

Conference, San Mateo, California, 83-96

modeling and design* Englewood Cliffs, NJ Prentice-Hall

Salzberg, B. (1994). On indexing spatial and temporal data *Information Systems*, 19, 
447-465.


and Evaluation of Advanced Traveler Information Systems *Transportation Research-
C*, 1, 2. 107-117

applications An experiment with O2* Paper presented at the Geographic Database 
Management Systems Workshop Proceedings, Capri, Italy. May 1991

Computer Technology Corporation, 3500 West Valcones Center Drive, Austin, Texas 
78759

Shinar, D. (1978) *Psychology on the road - the human factor in traffic safety* New York: 
John Wiley & Sons.


Worboys, M F (1992b) *Object-Oriented Models of Spatiotemporal Information*. GIS/LIS Proceedings, 2, 825-834


Figures

Figure 1  Different elements in ITS

Figure 2. Basic OO GIS entities for transportation network

Figure 3  An example of a node hierarchy for a transportation network

Figure 4: An example of a link hierarchy for a transportation network

Figure 5: Inheritance in a road hierarchy
Figure 1: Different elements in Intelligent Transportation Systems
Figure 2
Figure 3
Figure 4

- Link
- Road
- Highway
- Interstate
- Addressed
- Local
- Alley
- Lane
- Neighborhood
- Arterial
- Major
- Minor
Section IX

Developing an Object-Oriented Testbed for Modeling Transportation Networks

To be submitted to International Journal of GIS
Developing an Object-Oriented Testbed for Modeling Transportation Networks

Mei-Po Kwan
Department of Geography
Ohio State University
Columbus, Ohio 43210-1361

Jon M. Speigle
Department of Psychology
University of California at Santa Barbara
Santa Barbara, California 93106

Reginald G. Golledge
Department of Geography
and
Research Unit in Spatial Cognition and Choice
University of California at Santa Barbara
Santa Barbara, California 93106-4060

Acknowledgement: Funding for this project was provided by UCTC Grant # DTRS92-G-0009
Abstract

The objective of the paper is to discuss the development of an alternative representation of the transportation network using object-oriented GIS. This representation is important for the supply side of transportation planning and modeling. Object-orientation provides a way of solving the problem in a planar network for routing. It can facilitate the calculation of detailed network characteristics using properties such as inheritance and polymorphism. This representation is also closer to human perception of a transportation network. It is argued that by using an object-oriented GIS we can facilitate path selection using different criteria. We experiment with the design of the object-oriented system by developing an object-oriented representation of a transportation network and incorporating different path selection algorithms based on various behavioral assumptions. It is especially useful in the design for a versatile ATIS.

Introduction:

Transportation Science is host to a variety of theories concerning (among others) network structure, routing algorithms, traveler activity patterns, mode choice, demand forecasting, vehicle or traffic assignment, trip allocation, and traveler behavior. It has an expressed goal of increasing accessibility for all groups of people with regard to the environments in which they live and interact. A significant component of these goals is to further develop Intelligent Transportation Systems (ITS) through multi-level and multi-modal research and testing. This includes contributing to research on transportation system architecture, technology development, policy formation, and operational tests of various systems.

ITS objectives aim at utilizing advanced communication and transportation technologies to achieve traffic efficiency and safety. There are different components of ITS, including Advanced Traveler Information Systems (ATIS), Automated Highway Systems (AHS), Advanced Traffic Management Systems (ATMS), Advanced Vehicle Control Systems.
(AVCS) and Advanced Public Transportation Systems (APTS) Development of a coherent system for ITS depends on our ability to deal with a vast amount of data about the locations of places, as well as with the complex representation of the transportation network linking those places, and the incorporation of both of these into a geographic database. The system therefore, can be constructed based upon the foundation of an integrated and comprehensive Geographic Information System (GIS).

To achieve the ITS aim of using advanced information technology and processing to improve traffic efficiency, both static and dynamic information is needed. In addition, a representation of the real world environment is required, with all the streets and their properties, and with information about location of related objects being required. Vehicle routing and navigation is then based on this network representation. Further, dynamic traffic information updating in a short time interval must be included for accurate traffic forecasting. With recent advances in technologies, fast location and temporal updating are possible. Global Positioning Systems (GPS) can accurately fix and trace the location of a vehicle to within several meters. Movement detectors can provide current traffic counts. The central question is how to handle this fast temporal change and update of vehicular location and volume within an efficient database system.

Thus, the successful development of ITS depends on the capability of incorporating a vast amount of information about the location of facilities which generate travel, with as realistic a representation as possible of elements of the transportation network in which travel occurs. Such a system can be based on an innovative and comprehensive Geographic Information System (GIS). Whereas current ITS primarily use simplified transportation networks as their basis, using GIS allows us to provide a more realistic representation of elements of the network and the ways that people perceive them. We can represent the network by defining roads or street hierarchies and by storing
environmental data as layers which can be overlain, aggregated, or decomposed at will. Storing the transportation network as a hierarchy facilitates the calculation of different paths through the network and allows the introduction of different path selection criteria, and the ability to handle overlapping modal use of network elements (e.g., cars, busses, and freight carriers). Since a long-run aim of ITS is to develop a real-time multi-strategy travel decision support system over a multi-modal network, it is necessary to develop a data host and system model that is flexible, comprehensive, and realistic.

GIS have the potential to handle human movement in space and time. Existing GIS routines have been used to perform basic network actions such as finding a shortest path, solving a traveling salesperson problem, and recently, handling location-allocation problems. Applications commonly available in existing GIS software like TRANSCAD and ARC/INFO NETWORK have largely been operated in data models that rely on a simple planar link-node representation of network structure. However, this structure does not satisfy the various requirements of today's traffic management. In this paper, therefore, we illustrate a way of developing a testbed of a transportation network that could provide a realistic base on which to graft many ITS components.

I. System Review

A. System Overview:

Our system is implemented in C++, under the Windows NT operating system, and has a user-extensible interpreted front-end. The interpreted environment is comprised of variables, classes, and functions. The user is able to create text files which define classes and named sequences of statements which manipulate one and two-dimensional variables. The variables may be either homogeneous arrays of a primitive type (i.e., integer, floating point number, character, or string) or arrays of class instances. The arrays
of instances may be either all of the same type (homogeneous) or of different types (heterogeneous). The operations supported by a given array depends, accordingly, on the content type. For the numeric types, basic matrix operations are defined as well as a number of standard mathematical functions. For the string type, the operations include concatenation, comparison and formatted assignment. For the instance arrays, the method functions may be called and the attributes may be accessed.

Figure 1 Interface components

The Edit and Graph classes derive from a common, MDI child class. The Command Window extents the functionality of the Edit class by adding the parser.

The interpreter supports complex data types by means of aggregation and inheritance. Each type corresponds to a class definition and a set of "metadata". The metadata characterizes the types and names of attributes possessed by all instances of the class, the names and input/output signatures of methods supported by the instances, and the inter-
class, parent/child relations The metadata also includes the set of all instances of the class, or class extents.

Classes are defined in one of two ways The first method is for the user to create a text file which when interpreted creates the class. A class “road” is declared as follows. “Road” has two single-precision floating point attributes and a member function called “Cost”. The members may be referred to as “road·name”, where “name” refers to either a method or attribute name.

```cpp
class "road" {
    float s,
    float t,
    function Cost;
}
```

The syntax for deriving a class “highway” from “road” is as follows. A derived class “inherits” the attributes and methods of its parent class “highway” has two integer attributes as well as the floating point attributes and Cost method of class “road”

```cpp
class highway · road {
    int x,
    int y,
    function Cost,
}
```

If a method defined in the derived class has the same name as a method of the parent class, then it “overrides” or “hides” the method of the parent class. This feature, combined with the capability to store heterogeneous arrays of instances, allows for a method name to evaluate to a different function call depending on the type of the instance. In the following, classes ‘road’ and ‘highway’ are instantiated, the instances are stored in a vector, and their cost method is invoked:
For the instance of Road, Road::Cost is called, for Highway, Highway::Cost is called.

This is referred to as polymorphism

The second method of creating classes is for the metadata to describe a "built-in", C++ class. The metadata in this case describes a mapping from the method names to an index into a dispatch table. The index is used to call actual C++ methods for the class. The distinction between attributes and methods is blurred, as the methods provided in the dispatch table are the only means of accessing the attributes. In some cases, the built-in classes may be instantiated at the command-line. The spatial data types which will be described below fit into this category. Any built-in class which may be instantiated may also be derived from. This is the primary method of adding attributes to the built-in types. In other cases the built-in classes do not allow instantiation (e.g., the "Graph" or "MetaData" classes). Instances of this sort are returned by some routines (e.g., "gr = GetCurrentGraph"). The user may then pass methods to the instances (e.g., "gr Zoom(2)"). The user interface to the two types of metadata are consistent.

As with the class definitions, the operations supported by the interpreter come in two flavors: built-in and user-defined. Built-in operations are implemented as C++ routines and perform such tasks as listing and deleting variables, retrieving instances of built-in types (e.g., Graphs or MetaData), and interacting with the file system. The user-defined operations are text files which may contain any number of statements. When the first word of the file is "function" a local environment is created when the function is called (e.g., "z = Square(2)").
function [y] = Square(x)
  y = x^2;

The first line of the file declares the input and output signature of the function. For "Square" the input variable is "x" and the output variable is "y". The newly created environment is initialized with the function inputs arguments. The output variables are culled from the environment when the function is exited. Alternatively, when the first word is not "function", the calling environment is used.

The member functions of user-defined classes are also text files. From outside a member function, the member variables may be accessed and the functions invoked by following an instance's variable name with the method name. When a method is invoked, only the member variables corresponding to the class defining the method are accessible. But from within a method, these other method or attribute names may be referenced directly. For the case of overridden functions, a syntax is defined for explicitly calling the overridden function.

Queries are supported both against a class's extents and against the results of previous queries. In the following example a class is defined, several instances are created, a query is made against the class, and a second query is made against the first query's results:

```c
class Road {
  int v_road;
},

Road a(1),
Road b(2);
Road c(3),
Road d(4);

def=findclass(Road);
res1=def.Find("v_road>=4")
res2=res1.Find("v_road>=3")
```
When a query is made against a named class's extent, it is also made against the extents of any derived classes. The conditional is evaluated for each instance of the base class and then recursively on any derived classes. The instances for which the conditional evaluates to 'true' are accumulated as a heterogeneous instance list. This list may be assigned to a variable and subsequently used as the extent for further queries. The evaluation process sets the environment state as in a method call, to the particular instance against which the query is being evaluated. This means that complex, "path" expressions may be used within the conditional, that is, involving the attributes of fields which are themselves instances (i.e., "road highway==1")

A query conditional may be composed of any number of sub-conditionalos. Each sub-conditional will be a function/method call which evaluates to true or false, a relational expression (i.e., "">", "<", ">=", "<="), or an equality expression (i.e., "==" or "!=") The sub-conditionalos will be joined by the logical "and" and "or" operators && and ||. The precedence of the arithmetic, relational and logical operators is as in C++. The bindings for method calls are determined at run-time. The following example would return the instances of class Road for which the attribute was between 1 and 4:

```
res=def Find("v_road>1 && v_road<4")
```

### B. Built-In, Spatial Types

The spatial type hierarchy is shown in Figure 2. The basic spatial types are the Point, Line, Polyline and Circle. The types specific to a network implementation are the Node and Link. Each of these types is implemented as a C++ class and has a corresponding metadata description. The user is able to instantiate these classes at the command-line, to access the instance's methods and attributes, and to derive from these classes. The classes used to represent a hierarchy of road types are described in the next section. The methods
common to all types derived from spatial object include drawing, topological comparisons, and serialization. Each spatial object possess a "location" (i.e., x, y, and z coordinate) and unique "identifier". Polyline adds a list of vertices (i.e., x, y, and z coordinates). The Link adds attributes for identifying the "junctions" at the endpoints of the list of vertices. The Node adds a list of links to the basic attributes.

Figure 2. Built-in spatial object hierarchy

The spatial classes and any classes derived from them support spatial indexing of the class's extents. A quadtree of depth N is created for each class. Each quadtree leaf contains a list of pointers into the class extents. The list contains pointers to the instances whose centroid fell within the bounding box for that leaf. We refer to this procedure as "partitioning". Figure 3 shows the hierarchical relation between the basic meta-class type and the specialized, spatial meta-class type.
As noted above, member functions may be utilized within queries as well as value-based relational or equality statements. The topological member functions may therefore be used to select instances. A fully optimized query evaluator would utilize the spatial indexing when available. Such optimizations are operational for simple queries (i.e., queries composed of only a single spatial method call) and are under construction for more complicated, compound queries. The following example illustrates instantiating a built-in spatial class, partitioning, and executing several spatially based queries.

```
Line a([0 0],[1 1]),
Line b([1 0],[2 1]);

def=findclass(Line),
bbox=def.ComputeBBox(),
def Partition(bbox,3);

disp("Testing spatial query: find a"),
res=def.Find("IsInBBox([0 0 1 1])")

disp("Testing spatial query: find b"),
res=def Find("IsInBBox([1 0 2 1])")

disp("Testing spatial query: find a and b"),
res=def Find("IsInBBox([0 0 2 1])")
```
C. Road Hierarchy

The spatial classes were extended by deriving user-defined classes. The derived classes added attributes as well as constructor/destructor member functions. As shown in Figure 4, the class “Road” was derived from Link.

![Road hierarchy diagram]

Figure 4. Road hierarchy

The class definitions for Road, Highway and Addressed are as follows. The remainder of the definitions do not add attributes. Their purpose may be viewed as providing a hierarchical classification scheme. Member functions may be added to distinguish the favorability/unfavorability of any of the road classes.

```cpp
class Road : Link {
    string name;
    int nLanes;
    int divider;
    int paved;
    int speedLimit;
    int oneWay;

    function Cost;
},
```
The attributes of class Highway are the "level", which may take values of STATE or INTERSTATE.

```cpp
class Highway {  
    int level,  
    function Cost;  
};
```

The AddressedRoad contains attributes for characterizing the address ranges along the segment. The member functions determine the address given a position and the inverse process of determining the position given an address.

```cpp
class AddressedRoad {  
    int lowAddress,  
    int highAddress,  
    function PositionToAddress,  
    function AddressToPosition,  
};
```

### D. Junction Hierarchy

This section is still relatively undeveloped because at this stage, only the approach toward Junctions has been conceptualized. Currently, the Links contain only pointers to Nodes. This will be converted to pointers to Junctions in later phases of the work.

The basic network behavior is represented by class Node, which inherits from Point. The Node class adds a cost function and a list of connected Links. The class Junction inherits from Node and adds the capability to describe a set of turn restrictions. The Cost method of class Junction determines the cost of traversing the junction given a pair of links.

Figure 5 shows several user-defined classes which further specialize class Junction. These classes basically override the Cost function for class Junction. The further derivation of this derivation is shown in Figure 6.
E. How does OO affect route finding?

First, polymorphism allows us to call different cost functions for different types of road. For instance, the cost function for Link (Link::Cost) might just return the length of the link. The cost function of Road, which derives from Link, might divide the length by the speed limit to have the cost reflect the time needed to traverse the link (i.e., “Link::Cost/speed”). Polymorphism is the OO capability to call different cost functions depending on the type of the instance. Each road type in the Road hierarchy may define
its own cost function. A Dijkstra algorithm requires the network topology and the capability to evaluate a cost for each instance. The algorithm does not care about exactly how the cost is assigned. With polymorphism we are able to call different cost functions for different types of roads or we are able to minimize different criteria by modifying the cost functions. The code which implements the network algorithm is “re-used” while only the cost functions are modified.

With our system it is possible to minimize other criteria besides distance or time. Each cost function in the Road hierarchy is based on the output of the previous level. If Highway does not supply a specialized cost function, then its cost will automatically call the Cost function of its parent class. This is “inheritance.” But if we wanted highways to be preferred over other types of roads, then Highway::Cost could be set to “Road Cost * 0.1.” The cost for a highway would be less than for another type of Road. In this way, it would be possible to use the Dijkstra route finding algorithm with the following criteria: shortest path, shortest time, at-the-highest-speed, minimum-traffic-volume, most aesthetic, maximum proportion on a road-class, maximum use of one-way travel (or on divided road).

A similar strategy may be used to specialize cost functions for different types of junctions. We use a modified Dijkstra algorithm which allows assigning costs to traversing a node. With this implementation we may minimize criteria such as the number of left turns. The node has associated with it a turn matrix. A specialized, “hate-left-turn” cost function could produce a higher cost for making a left turn. This allows us to incorporate “behavioral” criteria into the route finding process. These behavioral criteria may even be customized for different users. With a junction hierarchy we may deal with the following criteria: minimize left turns, minimize delay at intersection, minimize number of signaled intersections.
Some types of criteria will, however, lie outside the capabilities of the polymorphic Djikstra algorithm. Those types of criteria could not be handled because they require information to be passed to the cost function which is not normally passed by the Djikstra algorithm. For example, a criteria could be to “always proceed in a corridor toward the destination.” Evaluating this criteria would require the endpoint and the position of the last point, neither of which Djikstra automatically supplies. Other questionable criteria are “place the longest/shortest leg first”, “avoid high accident places”, “maximize number of destinations along a single route”, or “maximize length of a dominant leg”.

F. Versant

The Versant ODBMS is designed as a fault-tolerant, multi-user, and distributed system. Some set of databases may be distributed across a number of machines and may be accessed simultaneously by any number of users. Versant’s client-server architecture supports distribution, the capability of which is essential for scalability. Transactions are handled in a robust fashion by using “two-phase commits.” The multi-user requirements have resulted in a sophisticated set of object locking and concurrency mechanisms. Users may “checkout” items from a group database to a personal database, with the items being locked until the corresponding “checkin.”

Our use of Versant’s system does not utilize many of the advanced features. A fully operational ITS system would, however, require the system to be highly distributed and robust. Our use of the distribution capabilities was limited to placing the database on one machine and the client program on another. We also did not explore the issues concerning concurrent access by a large number of users. In the testbed, a single user possessed read/write access to the database. In a fully operational ITS, different classes of users would have different levels of access. “End-users” would have only read access.
Concurrency would be an issue only for the “administrators” because they are likely to be the only users able to modify the database.

Versant provides interfaces for a number of programming languages: Smalltalk, C, and C++. We utilized the C++ interfaces, including the capability to declare classes at runtime.

We selected a particular OO DBMS software - VERSANT - as our primary medium (see also the recommendation in Gollu, 1995). Part of using Versant software as the medium for an object-oriented GIS is the development of the partitioning scheme. This is possible because of Versant’s implementation as a client-server database. Our application is the “client” and would request data from the database “server.” Of the several ways in which a client’s request may be couched, one is to use simple equality/inequality statements on a given class’s attribute values. The server will return class instances for which the conditional is met. We believe that this is the highest complexity of queries supported by Versant. More complicated, user-defined selection procedures are not supported yet because such procedures would need to reside on the server side. This is what is meant as “storing class methods on the server.” Versant only provides the means to store the data, not the methods. The only way around this inability to conduct complicated queries (such as a bounding box comparison against an entity’s position) seems to be to “recompute” these necessary computations. This is why it seems necessary to partition the map within Versant. The need is probably greater than it was for displaying the map.

The methods described above for creating hierarchically defined attributes and a hierarchically structured set of layers were mapped to Versant. Versant’s role was to allow for storage of class instances between sessions. In a relational database, this amounts to the storage on disk of an application’s tables and their loading when next
required. In much the same way that Versant scans the application’s C++ header files to
determine the class hierarchy, the code that has been written for creating class hierarchies
does the same thing. The class definitions are used to create the “meta-class” information
used to describe the memory layout of the attribute fields. The procedures to store class
instances to disk were then written based on the meta-class information (i.e., the class
fields are stored as a stream of bytes which is exactly what is needed to store the instance
to disk). This approach is an interim solution while the partitioning, network and entity-
attribute relations are worked out. All of these components seem required prior to storing
class instances using Versant. First, the data to be stored in Versant must be instances of
C++ classes. Second, using Versant as the storage mediator appears to require storing the
map within the database. Storing the map requires partitioning. It is still possible even
with a partitioning approach that map storage in Versant will prove prohibitively
expensive in terms of space or access time. We will investigate possible solutions to this
problem, particularly one that would be to store only the partitioning structure within
Versant, store the map separately in some other file format, and then load the map into
the partitions at run time.

G. A Real-World System Experiment

1. Conversion of Etak map to Object-Oriented Data Model

[This section is very sketchy and will be elaborated upon.] The road hierarchy was
defined and the Etak map of the Santa Barbara area was read into it. There are
approximately 28,000 nodes and some 14,000 links in the Etak county map, and 7800
notes and 3900 links in the testbed area.

I have not dealt with converting the Etak map into the junction hierarchy, as this may
involve much more user intervention. Making this jump requires the additional user-
interface capability of mouse tracking and selecting objects in the graph. The task of
converting nodes to/from different junction classes might also be automated by a set of dialog boxes.

2. "Modified" Dijkstra Routing algorithm
A description of the "modified" Dijkstra algorithm should be inserted here and how it was implemented using the Road and Junction classes

Evaluating the Testbed:
Some of the critical features involved in evaluating the worth of any object-oriented data model or object-oriented data structure include its ease of use, the relevance of the attributes defined in the system, whether or not the model deals with real or artificial concepts, the degree to which there is a clear translation between model entities and the actual objects, and whether or not there are acceptable matches between those activities undertaken in the real world and those activities incorporated into the model. Other criteria relate to the number of modules embedded in the system that have to be changed in order to work in a real environment, the number of steps that operation of the system requires, the degree to which one must know and accept a process model of the system, and the time that is required to integrate changes in the system to ensure it is dynamic real time.

Our project was designed to contribute to the next generation of traffic management technology, particularly in terms of dispensing information to travelers in a pre-planning or en-route phase via an ATIS. ITS generally appears to be moving more towards Object-Oriented data structures and models and we believe our work is in line with these nationwide trends.
We expect that this research will have important significance on both basic and applied levels. We have conceptualized and developed an object-oriented geographic information system from transportation modeling. Our continuing effort will focus on the data modeling issues of a multi-modal network, and the implementation of a multi-strategy travel decision support system built on this object-oriented system. Elsewhere (Kwan, Golledge, & Speigle, 1996) we have discussed the advantages and disadvantages of an object-oriented approach to transportation modeling. Apart from its improved capability of handling the transport network as more of a perceptually accurate system, the object-oriented data model allows us to incorporate hierarchical layering within the basic network that ties to the normal engineering way of interpreting road systems. Polymorphism allows us to perform ITS functions on various classes of objects in the transportation network, an ability that is not easily obtained using existing software systems. The approach also appears likely to substantially decrease the time involved in interacting with the database, particularly by using partitioning and inheritance characteristics. Cost of operation should consequently be reduced. Both these factors are important when considering that the primary aim of an ATIS component of an ITS is to get useful information to travelers in as timely a manner as possible so that on-route decision making can be undertaken. There are many basic research problems relating to the development of workable object-oriented data models for use in transportation planning and this research has examined some of these. We also expect that the data model as developed will greatly facilitate the implementation of ITS by more quickly resolving conflicts with respect to ultimate selection of routes, substituting destinations, changing activity patterns, and rescheduling activities.
References


Date, C J. (1985) *An introduction to database systems* Reading, MA: Edison Wesley.


Noy, I.Y. (1990) Attention and performance while driving with the auxiliary in-vehicle displays. Road Safety and Motor Vehicle Regulation - Transport Canada, Ottawa, Report #TP10727(E), December


Section X

Persons associated with the project
Section X  PERSONS ASSOCIATED WITH THE PROJECT:

1. Dr. Reginald G. Golledge, PI, Professor of Geography, UCSB  
2. Dr. Mei-Po Kwan, Professor of Geography, Ohio State University  
3. Jon Speigle, Graduate Student in Psychology, UCSB  
4. Scott Bell, Graduate Student in Geography, UCSB  
5. Wils Corrigan, Graduate Student in Geography, UCSB  
6. James Marston, Graduate Student in Geography, UCSB 
7. Violet Gray, Graduate Student in Geography, UCSB  
8. Kurt McClure, Undergraduate Student in Geography, UCSB  
9. Mike DeGennaro, Undergraduate Student in Geography, UCSB  
10. Mary MacDonald, Geography and RUSSC, UCSB  
11. Howard Pommerening, Geography and RUSSC, UCSB
Section XI

Papers and presentations
Section XI PAPERS AND PRESENTATIONS

a. Publications


b. In Press.


c. Papers Submitted


d. Papers in Preparation


e. Presentations:


Kwan, M-P (1996) Visualization and analysis of individual space-time paths using GIS. Paper presented at The relationship between GIS and disaggregate individual and behavioral transportation modeling conference (NCGIA I-10), Santa Barbara, California, June 7-8

