Modeling cadastral spatial relationships using an object-oriented information structure

Daniel Kjerne
Portland State University

Let us know how access to this document benefits you.
Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds
Part of the Geographic Information Sciences Commons

Recommended Citation

10.15760/etd.5605

This Thesis is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. For more information, please contact pdxscholar@pdx.edu.
This thesis identifies a problem in the current practice for storage of locational data of entities in the cadastral layer of a land information system (LIS), and presents as a solution an information model that uses an object-oriented paradigm.

A land information system is a type of geographic information system (GIS) specialized to handle land records. The cadastral layer of such a system stores, manipulates, and displays spatial and attribute data for property parcels, and is typically the concern of a local government assessment office. Current practice for storage of location of cadastral entities — e.g. property boundaries and survey monuments — does not allow for storage of certain data generated in the course of determining location for mapping, which data is essential for updating location as well as for certain queries which may be directed to the system.
The thesis uses a method of data store design that essentially translates the knowledge used by experts in cadastral location mapping into a description suitable for accurate translation into a computer language. This description, called an information structure, is based on a recently developed class of computer languages having a basic, object-oriented, approach more amenable than older, procedural, languages to modeling many of the problem areas understood by human experts.

After presenting the information structure description, the thesis examines an example mapped area to see how current cadastral mapping procedures would be affected, and presents (in an Appendix) a working computer program, written in an object-oriented language, illustrating some of the points developed.
MODELING CADASTRAL SPATIAL RELATIONSHIPS USING AN OBJECT-ORIENTED INFORMATION STRUCTURE

by

DANIEL KJERNE

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE in GEOGRAPHY

Portland State University 1987
TO THE OFFICE OF GRADUATE STUDIES AND RESEARCH:

The members of the Committee approve the thesis of Daniel Kjerne presented February 18, 1987.

Kenneth J. Ducker, Chairman

Joseph Poracsky

D. Richard Lycan

APPROVED:

Thomas M. Poulsen, Chair, Geography Department

Bernard Ross, Dean of Graduate Studies and Research
TABLE OF CONTENTS

LIST OF TABLES ............................................. v
LIST OF FIGURES .......................................... vi

CHAPTER

I  INTRODUCTION ........................................... 1
   Background ............................................ 1
      Cadastres and Cartography
      Land Record Modernization
      Designing the Cadastral Layer: Goals and Tools
   Relationship of Fields of Research; Literature ............. 10
   Approach and Organization of the Present Work ............ 13

II  THREE VIEWS OF THE ASSESSOR'S LAND INFORMATION SYSTEM ............................................. 14

III  PROCEDURE AND PARADIGM ............................. 20
   Levels of Description in Data Store Design ................. 21
   Object-oriented Languages ................................ 24

IV  TWO DESCRIPTIONS ...................................... 30
   The Information Reality of the Cadastral Cartographer .... 30
   An Object-oriented Information Structure .................. 36
<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>48</td>
</tr>
<tr>
<td>CADASTRAL MAPPING PROCEDURES</td>
<td>48</td>
</tr>
<tr>
<td>Background: Present Practice</td>
<td>48</td>
</tr>
<tr>
<td>Taking a Map Apart for Storage</td>
<td>51</td>
</tr>
<tr>
<td>Example Area Description</td>
<td>51</td>
</tr>
<tr>
<td>Definition of Cadastral Objects</td>
<td>51</td>
</tr>
<tr>
<td>VI</td>
<td>58</td>
</tr>
<tr>
<td>SUMMARY AND FUTURE DIRECTIONS</td>
<td>58</td>
</tr>
<tr>
<td>Summary</td>
<td>58</td>
</tr>
<tr>
<td>Implementation</td>
<td>59</td>
</tr>
<tr>
<td>Further Research</td>
<td>60</td>
</tr>
<tr>
<td>SOURCES CONSULTED</td>
<td>63</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>67</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>70</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>Description</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>System-level Inputs and Outputs for the Assessor’s Land Information System</td>
<td>17</td>
</tr>
<tr>
<td>II</td>
<td>Six Levels of Description of a Cartographic Data Base</td>
<td>22</td>
</tr>
<tr>
<td>III</td>
<td>Locational Characteristics of Cadastral Objects</td>
<td>31</td>
</tr>
<tr>
<td>IV</td>
<td>Number of Cadastral Objects in the Example Area</td>
<td>57</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Relationship of Fields of Research</td>
<td>12</td>
</tr>
<tr>
<td>2. The County Assessor's LIS as an Element in a System</td>
<td>15</td>
</tr>
<tr>
<td>3. Information System — System Level View</td>
<td>16</td>
</tr>
<tr>
<td>4. Information System — Subsystem View</td>
<td>18</td>
</tr>
<tr>
<td>5. Template for FloatingPoint Object Class</td>
<td>27</td>
</tr>
<tr>
<td>6. Template for Point Object Class</td>
<td>28</td>
</tr>
<tr>
<td>7. Templates for Rectangle and RoundCorRectangle Object Classes</td>
<td>29</td>
</tr>
<tr>
<td>8. Example Class Inheritance Hierarchy</td>
<td>29</td>
</tr>
<tr>
<td>9. Object Class Inheritance for Cadastral Location</td>
<td>37</td>
</tr>
<tr>
<td>10. Template for ControlPoint Object Class</td>
<td>40</td>
</tr>
<tr>
<td>11. Template for MonumentPointAngleDistance Object Class</td>
<td>41</td>
</tr>
<tr>
<td>12. Template for MonumentPointOffsetIntersect Object Class</td>
<td>41</td>
</tr>
<tr>
<td>13. Template for MonumentPointOnLine Object Class</td>
<td>42</td>
</tr>
<tr>
<td>14. Template for PropertyPointAtPoint Object Class</td>
<td>42</td>
</tr>
<tr>
<td>15. Template for SurveyLineStraight Object Class</td>
<td>43</td>
</tr>
<tr>
<td>16. Template for SurveyLineSimpleCurve Object Class</td>
<td>44</td>
</tr>
<tr>
<td>17. Template for MonumentLineTopoStream Object Class</td>
<td>45</td>
</tr>
<tr>
<td>18. Survey Location Method Modeled by MonumentLineTopoStream Object</td>
<td>46</td>
</tr>
<tr>
<td>19. Multiple Inheritance of Object Characteristics</td>
<td>61</td>
</tr>
<tr>
<td>20. Object-oriented Language Data Structures</td>
<td>68</td>
</tr>
<tr>
<td>21. Screen Dump of Demonstration (1: Control Points)</td>
<td>80</td>
</tr>
<tr>
<td>22. Screen Dump of Demonstration (2: Azimuth-distance Monuments)</td>
<td>81</td>
</tr>
<tr>
<td>FIGURE</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td>23. Screen Dump of Demonstration (3: Property Points)</td>
<td>82</td>
</tr>
<tr>
<td>24. Screen Dump of Demonstration (4: Lines)</td>
<td>83</td>
</tr>
<tr>
<td>25. Screen Dump of Demonstration (5: Redefine Control Point Location)</td>
<td>84</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

This thesis identifies a problem in the current practice for storage of locational data of entities in the cadastral layer of a land information system (LIS), and presents as a solution an information model that uses an object-oriented paradigm.

This introduction gives, first, a brief background of the problem context (cartography, land information systems) and the tools to be used in developing a design solution (database design theory, object-oriented language). This is followed by a sketch of the relationship of this work to other areas of research. Finally, a synopsis of the succeeding chapters is presented.

BACKGROUND

Cadastres and cartography

Input, storage, manipulation, and display of locational data are functions understood and performed in the context of cartography, typically using a storage device called a map. Robinson and Petchenik [1978, p. 119] have advanced a classification that uses two two-valued modes operating jointly to produce four categories of mapmaking: large-scale/thematic; small-scale/thematic; small-scale/general; and large-scale/general. It is interesting to note that academic cartography (as is found in university and college geography departments) has focused on the study of the first three of these, while work on large-scale/general (or reference) mapping goes on elsewhere. In particular, surveying, geodesy, and photogrammetry tend to be associated (in the academic environment) with engineering or engineering technology departments [Dahlberg and Jensen, 1986].
Cadastral cartography, or the mapping of land ownership, falls into this fourth category of mapping, and also tends to be ignored by cartographers in geography. Indeed, there is some evidence of mutual disregard, since the usual career path for cadastral mappers is through the assessment, surveying, or drafting fields, and the position is regarded as one requiring technical, rather than professional, skills and training [ODOR, 1981, p. i]. Nonetheless, it is a calling of no little significance and antiquity.

A cadastre is “an official register of the quantity, value, and ownership of real estate within an area administered by a government unit.” [McEntyre, 1984] Modern cadastres take as their unit of concern the land parcel, defined as “a contiguous area of land described in a single description in a deed or as one of a number of lots on a plat; separately owned, either publicly or privately; and capable of being separately conveyed.” [McEntyre, 1984]

A cadastral mapping system consists of maps portraying the size, shape, and location of land parcels. It is compiled from cadastral records, and serves as an adjunct to the cadastre proper to aid administrators and property owners in its use.

Cadastres, and cadastral maps, date far back into history. As the primary element of every known system of property taxation — which, itself, is an institution of long standing — they may be found on Babylonian clay tablets and Egyptian papyri. The first modern cadastre was instituted by Napoleon Bonaparte in 1807. This system was distinguished by the use of the individual land parcel as the unit of recordation; by the equitable treatment of all parcels, regardless of ownership; and by its design as a multipurpose public record [ODOR, 1979, p. 3].

In European countries, cadastres have been assigned more and more functions over the years. Beginning as fiscal registers, they were used in planning as industrialization proceeded during the nineteenth century, and took on an ownership-reference function early in this century [Barwinski, 1984].
Differences between European and North American legal relationships of people and land were mirrored in corresponding differences in the registers, records, and documents used to keep track of those relationships. The most profound divergence was the evolution, in North America, of a confusing array of separate land record systems, each maintained by a different governmental body for its own purposes, each with its own system of organization. Thus, the tendency in Europe was for new uses to be added to an existing system, resulting in a single multiple-purpose cadastre, while in North America a number of single-purpose systems appeared. For instance, administrative cadastres, the maps and registers used in conjunction to assess the value and tax status of land, evolved as one kind of special-purpose cadastre, separate from legal or juridical cadastres pertaining to legal description of rights in the land and their ownership and transfer. The property surveys recorded with local government agencies form the survey cadastre.

The administrative, legal, and survey cadastres are typically maintained respectively by the local government assessor, title record office, and survey record office. Other governmental and quasi-governmental agencies maintain extensive sets of land records, "spatially related documents that record governmental interests in the physical, legal, and environmental aspects of land — whether in, on, above, or under the surface of the earth." [McEntyre, 1984] These records are required, collected, and maintained for real estate, taxation, land transfer, environmental protection, and land management purposes.

For each cadastre, and set of land records, there may be a separate mapping system portraying locations of items of interest to the agency involved. This could potentially involve a tremendous redundancy of effort, but in typical practice, for cadastres at least, the assessor's map series (showing parcel locations) is used as a base, with other agencies using copies on which to depict their own data. There only remains the problem of maintaining currency with the original map base — which is typically no small task.
The amount of cadastral mapping activity worldwide is of mind-boggling proportions. In the United States alone it is very large. Depending on the laws in a particular state, local jurisdictions (counties, cities, parishes, townships, etc.) or some combination of them may be empowered to levy property taxes. The number of these assessment districts in the United States has been estimated at 13,426: 9,205 at the township level, 1,777 at the municipal level, and 2,444 at the county level [NRC, 1980, p. 19]. Each of those districts must maintain maps of the taxable land within its boundaries, in the course of which it is usually necessary to map the whole area, including the non-taxable portions. Most of the mapping is at a fairly large scale — from $1'' = 50'$ to $1'' = 2,000'$ ($1:600$ to $1:24,000$). Since each map sheet shows a relatively small area at such scales, each assessment district must maintain hundreds, perhaps thousands, of map sheets. For example, a medium-sized county in a Western state could easily require 2,000 map sheets without any mapping at a scale larger than $1'' = 400'$. These maps are continually being revised to reflect changes in ownership, new partitions or subdivisions, and refinements in location as new surveys of parcels are made and filed. And as old maps physically wear out through use, new ones must be constructed.

In addition to the multitude of public-sector land records and maps, a private-sector business, the title insurance company, has grown up to fill the gap caused by the multiplicity of confusing and often conflicting public records. These companies maintain a complete set of ownership descriptions and parcel maps for the areas they operate in, duplicating the public records. In some cases a title insurance company's title plant — the term for its set of ownership records — is more current than that of the local jurisdiction's.

Property and engineering surveyors in private practice often have extensive files of surveys and plats of projects which, for one reason or another, have not been recorded with the public survey office. These data also fairly regularly are of interest to governmental, quasi-governmental, and private actors involved in land.
Land record modernization

Over the years, vast numbers of land records and maps have been collected and stored at the local government level by private- and public-sector actors. Generally, the systems into which these records are organized are not coordinated with each other, making it difficult or impossible to relate data from one dataset to that from another. With the closing of the frontier, with the rise in environmental concern, with the increase in local planning efforts, has come a realization that land records in North America were generally in a state of duplication, decrepitude, and disarray [Portner and Niemann, 1984]. The perceived need for an organizational solution and the increased availability of computer technology raises the possibility that a unified land record system for local governments—an updated version of the European cadastre—might appear in North America.

And, in fact, in one major report [NRC, 1980] the approach proposed as a solution to the land records problem was called the multipurpose cadastre. As the term implies, this would be a cadastral system that integrates the single-purpose cadastres used by separate local agencies for administration, taxation and recording interests in land.

Although none of the components of a multipurpose cadastre need necessarily be in any other format than paper maps and files, it has long been apparent to workers in the new field of land information research, or land record modernization, that advances in computer technology hold out some promise of improving the ability to handle the large amounts of data involved—especially as it is desired to compare and analyze different datasets held by different actors or agencies. Specifically, the kind of technology that appears to be applicable is that of geographic information systems (GIS).

Dueker [1979] has defined GISs as a special case of information systems where the database consists of observations on spatially distributed features, activities, or events, which are definable in space as points, lines, or areas. A geographic information system manipulates data about these points, lines, and areas to retrieve data for ad hoc queries and analyses.
Clarke [1986] gives a functional definition based on common elements found in GISs. These elements are:

(1) a large body of data which have spatial or locational properties; (2) a unity of sorts, i.e. a GIS is not a set of stand-alone computer programs; and (3) a common set of sub-components which perform the functions of data collection, data storage, retrieval, analysis, and display.

GISs are specialized according to the type of problems they handle, reflecting the different users of the systems. A GIS specialized, through definition of data types, software, and hardware to answer questions about a particular spatial dataset consisting of cadastral or land records is a land information system (LIS).

Obviously, one of the major components of an LIS is the data contained in cadastral maps. As noted above, an enormous amount of data are recorded in these maps, and they are involved in some way in practically every aspect of cadastral and land records management — either as direct sources of data or as part of the framework for spatial location of other datasets. The centrality of this dataset has been recognized in a series of reports [NRC, 1980 and 1983; Cadastre Task Force, 1985] and papers [Coleman and McLaughlin, 1986; Held, 1986; Gaudet and others, 1984; Kitchen, 1984]. This dataset is called the cadastral layer, and forms a component of the land information system used by a local government (county or equivalent, township, or municipal) assessor.

Given the crucial nature of the cadastral layer, it is ironic that present approaches to its implementation as part of an LIS do not adequately capture some important data: the spatial relationships between cadastral objects as determined by property surveyors in the field and as used by cadastral cartographers in the construction of the maps. Essentially, the object location is known in the cadastral layer only by its x,y coordinates on a plane, while the knowledge of why it is there — although it was used to derive the x,y coordinates — has been lost.

The objects of concern in the cadastral layer are such things as property corners, boundaries, right-of-way lines, monuments, and property parcels. The location of these
objects is determined (in both operational and legal senses) using procedures and philosophical approaches very different from those which mapmakers in other branches of cartography are used to. For instance, in adjusting a photogrammetric image to ground control (a standard step in preparing an orthophotomap), the image areas between ground control points are convoluted using a continuous mathematical function. This type of continuous deformation of the map image would be inappropriate to use in adjusting a cadastral map to ground control, because there is not necessarily a structural (locational) relationship between adjacent cadastral objects, while there may be one between objects distant from each other. In addition, many objects depicted in a cadastral map are not physical objects at all, but legal ones. Their locational behavior follows legal rules, rather than those characteristic of other types of maps. For example, the location of a boundary between property parcels will depend upon the relative date and manner of creation of the parcels as well as upon the physical evidence of location.

In constructing a conventional (paper) map, the cadastral cartographer reviews and analyzes deed descriptions and property surveys to determine the relative locations of property points, boundaries, and so on, before compiling those locations into a global set of spatial relationships on the cadastral map sheet. Since there is simply no room on the face of the map for all the information generated in the course of determining locations, only the location is shown.

Present-day GISs — and LISs — model their location storage on paper maps. They store only the derived location in global coordinates, rather than storing the intermediate knowledge of relative location. As Tomlinson and Boyle [1981] note:

The state of development of spatial data structures ... is still rudimentary. In essence, they attempt to preserve the traditional map in digital form and are not data bases where spatial relationships are more explicit. There are several reasons for this. A newly developing technology often mimics the products of the earlier technology before realizing its own capabilities. Moreover, several fundamental questions concerning the nature of spatial query, and indeed the nature of spatial relationships, have to be answered before new forms of spatial data structure, with more explicit spatial relationships, can be designed effectively.
Although the authors were speaking of systems available in 1980, the situation has essentially not changed since.

As the GIS spatial data structure follows the model of the paper map, so also does the updating of spatial data. When a paper cadastral map is updated in more than a minimal sense — when, that is, it is more than a case of adding a line to subdivide an existing parcel — the original deed description and survey data must be consulted and the map reconstructed from scratch. A similar process occurs with the cadastral layer in an assessor's LIS: updating the location of parcels in a mapped area requires deletion of stored map data, reconstruction from source materials, and installation of the reconstructed data, rather than simply adding new data [Dueker and others, 1985; Kjeme and Dueker, 1984; Kjeme, 1986].

**Designing the cadastral layer: goals and tools**

A land information system, like any information system, must accept questions and present answers in terms familiar to and appropriate for its principal users if it is to be maximally useful. For the location of a cadastral object, this implies two functions not performed by present LISs: 1) being able to trace the chain of legal and field survey operations that lead to that particular object being in that particular place; 2) being able to update an object's location (when, for instance, a more accurate survey is made) and to rely on the system to update locations of objects that depend on the updated object.

The goal in designing an information system is to produce a description of a portion of reality which corresponds in its structure to the view understood by the human user of the system, but which is used by the machine as it answers questions about that portion of reality. This involves, essentially, a process of translating a description from human language to machine code, and is best approached by producing a series of intermediate descriptions, each one developed from the previous one [Nyerges, 1981].

The initial (highest-level) description is understandable only by humans, and
consists of statements and observations, in a human language, about a portion of reality; the final (lowest-level) description is understandable only by a particular machine configuration, and consists of machine code. In the present work, concern is focused on the translation from the highest-level to the second-level description of a certain portion of reality. This second-level description, while still understandable by humans, is an abstract structure of the knowledge contained in the first description, and is usually presented as a set of diagrams of entities and the relationships among them.

Below this level the descriptions are phrased as database file specifications, access paths, etc. At each level, the descriptions are less easily understandable by humans. Fortunately, once the translation is made from the second level to the highest level at which a computer can accept the description as input, translation to lower-level descriptions can be made by the system itself.

The basic difficulty in this translation process has been that human languages operate under a different paradigm than computer languages. In particular, the second-level description is phrased in terms of entities which have certain properties and certain relationships to other entities. But most computer languages do not differentiate between entities, properties, or relationships in any fundamental way. All are treated as data, to be operated on by procedures existing, as it were, in another part of the program. That is to say, the connection in conventional computer languages between an entity and its properties is contingent on procedures defined in the application rather than being (as it is at the second-level description) inherent to its structure.

Recently, new computer languages have been developed which solve this difficulty; they are called object-oriented languages, because the basic paradigm under which the languages operate is that of objects and messages. The objects “know” their internal state, can change that state in response to a message, and can communicate that state to other objects. It is thus possible to form a correspondence between the entities and their
properties and relationships of the second-level description, and objects at the computer-language level.

RELATIONSHIP OF FIELDS OF RESEARCH;
LITERATURE

The appearance of computer-aided mapping systems and geographic information systems presents us with a new set of problems in defining the scope of cartography and the meaning of the term “map.” What is the essential nature of a map? Is it only the displayed or plotted graphic visible to the user, or should the term be broadened to include the model in computer storage used to produce the display?

The ground-breaking study in what has been called “metacartography” [Robinson and Petchenik, 1976] defines the map as a “graphic representation of the milieu” (emphasis added). In their discussion of the “structure” of a map, the authors focus on the depiction of abstract relationships, rather than on the model that underlies the relationships. This emphasis on the picture that tells the story, rather than the story that tells the picture, is probably due to a focus on thematic cartography (in which a graphic representation allows a map percipient to mentally produce a structural relationship). In computer-aided cartography, on the other hand, the focus is rather the reverse: to find the appropriate abstract structure (data storage) to produce a graphic representation.

Moellering [1985], with the model of computer storage and display before him, presents the idea of “real” maps (graphic objects) and “virtual” maps (data structure and rules to produce graphics), but confuses the issue with an artificial distinction between maps with and without “permanent tangible reality.” All maps are more or less impermanent. Moellering’s four-fold classification is therefore not based on the interaction of two dichotomous modes, but on the interaction of one mode that discriminates objects different in kind (real or virtual maps) and another that seeks to dichotomize objects actually existing on a continuous spectrum.
Nygerges [1981], seeing the emergence of a general theory of database design, places the development of cartographic data bases within a general theory of information, specifically semiotics.

Youngmann [1979] applies to maps the idea of picture grammars derived from pattern recognition.

Further exploration of these questions will continue to benefit from the new metaphors available as the technology of geographic modeling develops. A model of the spatial relationships of cadastral objects based on their behavior as understood by property surveyors and cadastral cartographers is different enough from the model of spatial behavior of other objects mapped by other specialists that it may present the opportunity for more general insights to develop about the nature of maps.

This benefit places the present work in the geographic tradition, but is seen as a side effect of the attempt to make better cadastral mapping systems. As such, the present work is part of a multidisciplinary effort not wholly contained in any of the traditional disciplines. Figure 1 illustrates the relationship of several of the fields involved.

Geographers are deeply involved with land record modernization. Some anticipate beneficial outcomes for geographic research resulting from improved land records. Others hope to provide assistance in the process of modernizing land records based on their particular expertise — cartographers, for instance, might have some insight into the storage and display of geographic information. But there are other aspects of the land record modernization problem — e.g. intergovernmental relations, storage and manipulation of non-spatially related data, operations research — where the claims of geography are weak at best. As a problem included within computer-aided mapping, however, it would appear that the present work is safely within the geographic domain.

This is not reflected in the literature on the specific problem, however, which tends to be limited, sometimes fugitive (e.g. newsletters, questionnaires, unpublished papers),
and to appear in computer science, information systems, surveyors, and professional assessment venues quite as often as among those of geography and cartography. The author is aware of a good deal of recent interest in development of object-oriented GISs, but not of their application to the specific problem of modeling cadastral spatial relationships. This is partially due, no doubt, to the recency of development of object-oriented languages.

In addition to its involvement in a multidisciplinary rather than a purely geographic problem, the present work is somewhat unusual in a geographic research context in that it attempts to solve a design problem rather than present the result of scientific investigation. It finds its model not in works that present hypotheses, data, and conclusions, but in engineering, architectural, or — closer to home — cartographic, work in which a presentation or construction technique is explored.
This first chapter has presented a background to the problem of storing locational data for cadastral objects in the cadastral layer of an assessor's land information system, presenting it within the context of present research in land record modernization and investigations of cartographic data structure.

The next chapter presents views, at three different levels, of an assessor's land information system, in order to gain a better idea as to 1) what it is part of; 2) what it does; and 3) what it is composed of.

Chapter III introduces the procedure to be used and the paradigm to be followed in preparing a description of the information structure of cadastral location. The procedure makes use of the concept of multiple levels of description of a database, while the paradigm derives from the concepts of object-oriented languages.

Chapter IV presents two descriptions of the structure of cadastral location: the first as it is understood by the cadastral cartographer in the process of compiling or revising a map; the second using an object-oriented paradigm. The objective of the second description is to preserve the meaning of the first and to be readily translatable into a high-level computer language or database description language.

Chapter V investigates the map construction and revision procedures implied for a cadastral layer by an object-oriented structure for cadastral location. This is done by examining an example area of a cadastral map to see, in practice, how objects are related in locational terms, and how an object-oriented cadastral layer would capture those relationships.

Chapter VI summarizes progress to date and raises some questions for future investigation, including the specifics of implementation and possible extensions.
CHAPTER II

THREE VIEWS
OF THE ASSESSOR’S
LAND INFORMATION SYSTEM

A system can be described in terms of a series of views, at different levels, of the systems which include it, and those which it includes. The broadest view presents it as an element in a larger system. If the view at the next level down — the view that users have of their systems in the context of their day-to-day interactions — is called the system view, then this highest-level view may be called the metasystem view. In like manner, the view on the system at the level below the system level — the level at which we are considering the system in terms of its internal parts and processes — is called the subsystem view.

Figure 2 depicts the metasystem view of the assessor’s LIS and other land information systems in a local cadastral and land records data sharing environment. The figure is based on the system concept advanced in Dueker, Conrad and Kjerne [1985], and should be thought of as a view “from above.” No system user, including those in the “multipurpose cadastre” area, would have this perspective in the course of using their system. From the point of view of an observer at the metasystem level, this is not an “information system,” because it does not take questions as input and present answers as output. It is a system of land information systems. The observable characteristics at this level include the identities of the components and the interactions (primarily data and information flows) between them.

We can observe that while the multipurpose cadastre forms part of the system of land information systems, it is formed, in turn, from the land information systems of a
small set of county agencies closely involved with land records. Depending on the particular situation, these three land information systems might exist in one physically integrated installation or as several independent systems at different sites, with several possible means of data transfer. These are extreme points on a spectrum of integration: in the first case, the separation of the multipurpose cadastre into separate information systems seems more conceptual than real, while in the second case it is hard to say that three systems in different sites constitutes "one thing."

The arrows indicate some (by no means all) flows of data. Data flows to the assessor’s LIS from the county surveyor’s and county recorder’s LISs; from outside the multipurpose cadastre, title companies, planning agencies, and administrative agencies all provide data to the assessor’s LIS. In addition, private surveying, engineering, and photogrammetric mapping firms supply data indirectly to the assessor’s LIS by providing
base map information to the county surveyor's LIS, which then sends it on to the assessor's land information system.

The content of the data flows cannot be discerned at this level of observation; that will be possible at the next level down. But it can be said, at this point, that the most important components of the data flow from the assessor's land information system to other information systems include the location, size, and shape of land parcels — data from the cadastral layer of the assessor's LIS.

In Figure 2, data flows from the assessor's LIS to every other information system, pointing up the central importance of the cadastral layer component of the assessor's land information system. Data flows also occur between the land information systems of agencies and firms outside the multipurpose cadastre.

Figure 3 shows the characteristics of an information system observable at the system level — the level of users' interaction with the system. In this figure, the information system is perceived as a black box with certain inputs and outputs: data, resources, questions, and answers. For each user, each data supplier, and each resource supplier, the system will exhibit a different set of behaviors. Table I shows some of the input data, resources, questions, and answers specific to the assessor's land information

![Figure 3. Information system — system level view.](image)
system, including a good deal of things having to do with location of cadastral objects — the "cadastral layer" dataset.

**TABLE I**

**SYSTEM-LEVEL INPUTS AND OUTPUTS FOR THE ASSESSOR'S LAND INFORMATION SYSTEM**

<table>
<thead>
<tr>
<th><strong>input data</strong></th>
<th><strong>resources</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>location of survey objects</td>
<td>hardware:</td>
</tr>
<tr>
<td>parcel identifiers</td>
<td>plotter</td>
</tr>
<tr>
<td>date of survey</td>
<td>modems</td>
</tr>
<tr>
<td>degree of accuracy of survey</td>
<td>tape drives</td>
</tr>
<tr>
<td>parcel owner name</td>
<td>CPU</td>
</tr>
<tr>
<td>owner address</td>
<td>etc.</td>
</tr>
<tr>
<td>partitions</td>
<td>software:</td>
</tr>
<tr>
<td>improvements</td>
<td>communication</td>
</tr>
<tr>
<td>easements</td>
<td>coordinate geometry</td>
</tr>
<tr>
<td>zoning boundaries</td>
<td>etc.</td>
</tr>
<tr>
<td>administrative boundaries</td>
<td>personnel:</td>
</tr>
<tr>
<td>addresses of permitted improvements</td>
<td>system analyst</td>
</tr>
<tr>
<td>valuation</td>
<td>cartographer</td>
</tr>
<tr>
<td>etc.</td>
<td>etc.</td>
</tr>
</tbody>
</table>

**questions (commands)**

"Display a section map."

"Display a particular parcel."

"Find data about a particular parcel."

"Find data about a group of parcels."

"Summarize data about a group of parcels."

**(format of) answers**

graphic display (map)

graphic display (plat)

report

report

graphic display (chart)
We go down one more level (Figure 4) to gain the subsystem view — to "look inside" the assessor's LIS. Clarke [1986], with other writers [Tomlinson and others, 1976; Dangermond, 1984] has identified various numbers (ranging from four to six) of subsystem components of a GIS or LIS. While that level of detail may be appropriate for design of a complete system, the purpose here will be served by thinking of subsystem functions in the broadest terms. Thus the assessor's LIS can be thought of as containing three objects: a data capture subsystem, a data storage subsystem, and a subsystem to decipher user questions and provide answers. These three have been characterized [Lendaris, 1985] as the data selector, data store, and data massager.

The selector contains capabilities to convert graphic or written locational and attribute data to digital form. This subsystem serves as an interface between the input data and the data store.

The data massager serves as the interface between the user (in the role of question...
asker/answer receiver) and the data store.

The data store for the assessor's land information system contains, in addition to the files of spatial data constituting the cadastral layer, additional files relating to other aspects of the management of assessment data.

For the assessor's LIS, each of these three subsystems is specialized to handle the types of input data, questions, and answers (using the resources) indicated in Table I.

Development of either a data selector or data massager subsystem for an information system is a considerable task — as would be the complete specification of a data store. It amounts, for the selector, to specification of the hardware components and software routines to convert input data to the form required by the data store. Similarly, for the data massager, the task implied is that of design of a user interface — the hardware (e.g. display, keyboard, plotter) and software (e.g. query language, device drivers) to handle questions, pass them to the data store, and present the answers from the data store to the user. The complete specification of the data store involves defining the data items of concern to all the users of the information system throughout the organization. In the present case, some of the users of the assessor's LIS are assessment officers, system analysts, administrators, and the public — in addition to cadastral cartographers. The task of the present work is restricted to a focus on a single aspect of the data store from the point of view of one of the users — that is to say, the location of cadastral objects by the cadastral cartographer. The next chapter presents some of the tools used in that task.
CHAPTER III

PROCEDURE AND PARADIGM

The previous chapters have presented the context in which the problem of storage of cadastral object location occurs and a series of views of the system involved. It should be noted that the present work is restricted in scope to a small, albeit crucial, aspect of the design of an assessor's land information system, since it concentrates on cadastral object location as it is understood by one of the multiple users of the system—i.e., the cadastral cartographer.

At the same time, the design procedure for the information structure, and the paradigm for that design, may prove to be of more general use. This chapter presents some background on that procedure and paradigm.

The procedure undertaken in Chapter IV is to present knowledge of locational relationships of cadastral objects in two ways, as understood at two different levels: 1) as understood by humans versed in the problem area; and 2) as understood by humans designing (part of) a land information system to answer questions in the problem area. The first part of this chapter—Chapter III—presents a fuller exploration of the concept of information system design as a process of “translating” knowledge representations from a level of knowledge representation understandable only by humans to a level understandable only by the system—that is, by a particular machine hardware/software configuration.

The paradigm used for the second way of presenting knowledge of cadastral location is based on the structure of object-oriented computer languages. For many knowledge areas (including cadastral location), this structure is closer to that of the representation as understood by humans acquainted with the problem than a structure based
on other computer languages. Thus, the second presentation, or description, of cadastral location presented in Chapter IV is phrased in terms of objects that "know" their internal state and that "know" how to perform actions in response to messages directed to them. The second section of this chapter — Chapter III — gives some background on this type of computer language, explains some of the terminology used, and presents two methods (object class templates and class hierarchy diagrams) used in the following chapters to encapsulate knowledge about the classes of objects, and their relationships, in a knowledge representation.

LEVELS OF DESCRIPTION IN DATA STORE DESIGN

The design strategy for an information system may be thought of as involving the preparation of a series of descriptions of entities and their relationships — the subjects of questions directed to the system. The first description will, of necessity, be general in nature. It will only be implementable — that is, useful in producing answers to questions — with the aid of a highly sophisticated "knowledge base," hardware, and software configuration. So sophisticated a configuration, in fact, that it only exists now in the form of humans, who can take a verbal description of a problem, a set of data, and tools such as pen, ink and calculator, to produce answers to questions.

Once a description has been formulated at this level, it must be "translated" into a more specific description so that it can be implemented using a less sophisticated hardware/software configuration. The objective is to preserve the ability to answer questions, but to, in effect, move the expertise of the higher-level description from the minds of the human experts into the information system.

The number of levels necessary to traverse between the human verbal description and the machine code implementation varies with the type and complexity of the problem.
Nyerges [1981] discusses the design of a cartographic data base in terms of six levels of description (Table II).

**TABLE II**

<table>
<thead>
<tr>
<th>SIX LEVELS OF DESCRIPTION OF A CARTOGRAPHIC DATA BASE [NYERGES, 1981]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) information reality</td>
</tr>
<tr>
<td>infological models</td>
</tr>
<tr>
<td>2) information structure</td>
</tr>
<tr>
<td>3) canonical structure</td>
</tr>
<tr>
<td>4) data structure</td>
</tr>
<tr>
<td>5) storage structure</td>
</tr>
<tr>
<td>6) machine encoding</td>
</tr>
</tbody>
</table>

The highest level description, **information reality**, consists of “observations … about geographical entities and their relationships which knowledgable persons would communicate with each other using any medium ….” [Nyerges, 1981, p. 7] In the present case, this description consists of observations about cadastral objects and the spatial relationships cadastral objects have with each other, as they would be described by cadastral cartographers.

The information structure and canonical structure are both **infological models**. These are descriptions specified by the system user or subject area expert (as contrasted with those specified by a programmer).

The **information structure** is “a formal model … [acting] as an abstraction of reality … It includes entity sets plus the types of relationships which exist between those entity sets.” [Nyerges, 1981, p. 7] That is, its structure is explicit, while the structure of observations about reality at the higher level may be implicit.
An organization is likely to deal with a number of different “information realities” corresponding to the different concerns of its departments or offices. Each information reality will have its own information structure. The canonical structure is “a minimal structure … developed from all information structures” in an organization. The intent is to reduce redundancy (and updating and concurrency problems) by having any entity, though it may be referred to by more than one department within the organization, appear only once in the structure. If there is a single information structure for an organization, the canonical structure and the information structure are synonymous [Nyerges, 1981, p. 9] — which is what happens in the present case, since there is only one information reality under consideration.

Levels of description below the canonical level are datalogical. That is, they are specified by a programmer rather than a user. The data structure level is the highest one at which a description of the portion of reality under consideration can be entered to a machine. With conventional database management software, for instance, this activity takes the form of specifying formats for fields, defining the pointers from fields in one file to those in another, and so on. With relational database management software, much the same kinds of things happen: specifying the layout of forms and the relationships between items. (One of the advantages of relational database management software over conventional ones is that these relationships can be revised and new ones added after data has been entered into the system without the necessity to convert the database to a new structure.) For an object-oriented database management system, definition of the data structure would entail specifying the objects known to the system, including the types of values they store internally, and the actions they perform in response to messages.

The descriptions at levels below the data structure are primarily of interest to programmers, and will not be considered here in detail. It may be noted that these descriptions can be produced automatically by the machine in its compilation process,
although programmers or system designers may wish to intervene to optimize a description for speed or storage considerations.

The advantage of using an object-oriented paradigm in specifying the information structure should be apparent: once the specification is performed at a level of description understandable by humans, it can readily be transferred to the data structure level without having to conceptually break up entities and relationships into data types and procedures.

OBJECT-ORIENTED LANGUAGES

Most computer languages are procedural languages. These languages operate under the "data-procedure" paradigm, in which active procedures do something to the passive data presented to them. For instance, a square root function, presented with a number, returns its square root. To do this appropriately, it has to be presented with the right kind of number — integer, floating point, double precision or whatever. If a procedure designed to find the square root of an integer were presented with a floating point number, it would still operate on the data — but the returned result would be garbage. To be able to return the square root of any of the various kinds of numbers that might be used in a computer application, a series of square root functions must be defined, one for each data type, and some means must be set up to know the data type of a number before it is acted upon by the procedure. With procedural languages the programmer, rather than the system, assumes this responsibility.

Object-oriented languages operate under the "message-object" paradigm, in which data and procedures are combined into a single entity, an object. When an object receives a message, the object performs an operation on its own data. To use the same example as above: in an object-oriented system, an integer object can be defined which knows how to take its own square root; a floating point object can be defined which knows how to take its square root, using a different method. The same message could be sent to each of these
objects, interpreted differently by each, and appropriate responses returned.

As a programming style, object-oriented languages began in the early 1960's with the development of Simula by the Norwegian Computation Center in Oslo, Norway. Simula was the first language to implement the Class construct. In the early 1970's, the Learning Research Group at Xerox Palo Alto Research Center began implementation of the Smalltalk programming environment. Smalltalk was the first system to be designed completely around the Class/Object concept. Many other languages have since provided support for classes, objects, and subclassing, including CLU, Ada, C++ (an extension of C), and several versions of LISP. [Horn, 1985]

Only a few of these languages are truly object-oriented, though they may implement some of the features found in object-oriented languages. To be object-oriented, a language needs to possess four capabilities: 1) information hiding; 2) data abstraction; 3) dynamic binding; and 4) inheritance.

Most modern computer languages, including procedural languages, have an information-hiding capability. This is the capacity to define software modules with variables which are not accessible to procedures outside the modules, but only to ones inside them.

This capability is carried a step further in the capacity to define new, abstract data types using simple data types, such as integers or bytes. Queues, stacks, and arrays are examples of abstract data types: entities which keep track of a number of items, and allow access from "outside" in strictly defined ways (e.g., only the "top" item of a stack is available). Again, most modern languages, including procedural ones, allow definition of abstract data types.

Dynamic binding is the determination of the data type of an object during runtime, and hence permits the linking of procedures appropriate to the type of the object. This allows polymorphism, illustrated by the example above where the same message to
a variety of objects resulted in different, appropriate procedures being selected for each type. With the addition of the capability for dynamic binding to the capabilities of information hiding and the definition of abstract data types, languages become object-oriented. (It is possible to emulate polymorphism in procedural languages, but the connection — the binding — between data type and procedure is not truly dynamic: it must be accomplished during the compilation stage, not during runtime.)

**Inheritance** of object characteristics allows object classes to point to other classes where other internal variables or methods can be found. This permits the definition of object classes which have the characteristics of the pointed-to class, in addition to characteristics unique to the new object class. This capability allows a major reduction in the effort required to write new code. Presently, in most object-oriented languages, each class can point to only one other such class, called a **superclass**.

**Instance** and **factory** are alternative terms for object and class (or class definition) that may help to clarify the relationship of these important concepts. When a class definition ("factory") in an object-oriented program or application receives the appropriate message (e.g. "new") it creates in memory a uniquely identifiable instance ("object"). In order to do this, the class definition contains information about the methods, instance variables, and superclass for that particular class of objects. **Methods** are the procedures an object "knows" how to perform, while **instance variables** are the data items an object "knows" how to store and access. The objects (instances) of a class are individual "packages" storing particular values in their instance variables.

Appendix A consists of a reproduction of a text box from Pascoe [1986] giving a concise explanation of the basic terminology of object-oriented programming, and depicting the basic data structure of an object-oriented language. The following discussion makes use of a message expression syntax presented in the same source; the syntax is derived from, though simpler than, Smalltalk's.
In order for a message to be understood, there must be an indication of the recipient of the message, the action to be undertaken, and, sometimes, other values to be used in the action. In the message expressions used in the present work, the first element is the message's **receiver**. This is followed by the **selector**, denoted by a leading colon (:). Any further arguments follow, and the expression is terminated by a period (.). For example, in the message expression “x :sqrt.” the implication is that the object x is being asked to perform a square root operation on itself. In the expression “x :+ 5.122.” the object x is asked to add the value 5.122 to itself.

An object class definition can be summarized in an **object class template**, which gives the class name, superclass, internal variables, and methods for a class of objects. If x, in the examples above, were an object of the FloatingPoint class of objects, its class template could look like that in Figure 5.

In the figure, the first line of the indicated methods gives the method selector and the pattern to be expected for any arguments. It is possible to see how the example expressions given above fit the message patterns of their corresponding methods in the class definition template. The body of the method would, if we were writing actual class definitions, contain a set of expressions to be evaluated. Instead of expressions, the class templates as presented in these figures contain, between parentheses, general descriptions of what the methods do. Appendix B contains listings of object class definitions written in Neon, an object-oriented language for the Apple Macintosh computer, and

![Figure 5. Template for FloatingPoint object class.](image-url)
provides an idea of what method definitions would really look like.

Figure 6 shows a template for the Point object class. The "instance variables" area is of particular interest here: x and y are where the object stores the value of the x-coordinate and y-coordinate, respectively. These objects perform the indicated operations in response to the messages ":put" and ":+" followed by their respective arguments, and return the value of either x- or y-coordinate in response to the "getx" or "gety" messages.

Figure 7 illustrates the inheritance characteristic of object-oriented languages. Figure 7a is a Rectangle class template, showing three methods to which this object can respond. Note that the instance variables are of the Point object class.

Figure 7b shows the class template for a subclass of the Rectangle class. The RndCorRect is a rounded-corner rectangle, able to respond to messages for its class as well as those for its superclass which have not been superseded. The search for any message selector begins at the class level; only if it is not found there does the search go to the superclass. Thus, for instance, if a RndRect object z is sent a message "z :put a b c." the Point values (a and b) are put into z's TopLeft and BotRight internal variables. (Note that these are not defined in the RndRect class, but in the Rectangle class.) The floating point value c is put into the Radius internal variable, which is defined in the RndRect class.

If a RndRect object is sent a "fill" message, the search fails at the class level,
Figure 7. Templates for Rectangle and RoundCorRectangle object classes.

and goes to the superclass. There it succeeds; the area is filled with a pattern.

Figure 8 shows an object inheritance diagram, which depicts the relationships among a set of object classes. The class hierarchy in the figure is the one defined among the object classes described in the templates above.
CHAPTER IV

TWO DESCRIPTIONS

This chapter has two parts: the first presents the "information reality" of a cadastral cartographer constructing a parcel map; the second presents an "information structure" of the entities and relationships discussed in the first section, using an object-oriented framework — class inheritance diagrams and object class templates. The first description is based on three main sources: ODOR, 1979; Brown and Eldridge, 1967; and, finally, the author's approximately ten-year experience in the fields of property surveying and cadastral cartography. The second description illustrates the facility with which such a framework can be used to mirror the structure of a knowledge domain as understood by its experts and, at the same time, be translatable to a datalogical system description.

THE INFORMATION REALITY
OF THE CADASTRAL CARTOGRAPHER

To find the location of objects on a cadastral map, the cadastral cartographer researches deed descriptions and property surveys. Deed descriptions are analyzed to understand which boundaries have legal precedence — for instance, whether a parcel was created at an earlier time than its neighbor. Survey records are inspected to determine the steps the property surveyor took to determine the physical location of evidence on the ground, and what the surveyor found to be the relation between the physical evidence and the location of legal objects such as property corners and boundaries.

Table III summarizes the characteristics of the objects which the cadastral cartographer is concerned with locating. (There are other objects, having location as a characteristic, on a cadastral map — parcels, grid ticks, annotation, borders, and the like —
but their locational behavior is not a concern of the present work, since it is adequately specified by global coordinate values or simple relationships to other objects.) These objects can exhibit a combination of four dichotomous characteristics.

First, the object may be physical or abstract. That is to say, it may or may not have
Monument points and lines are physical objects. Monuments may be natural or artificial. Natural monuments are objects such as trees, boulders, streams, ridges, and so on. Certain artificial structures, such as buildings or curbs, are considered durable and stable enough that they are classed as natural monuments. Artificial monuments are usually objects such as iron bars or rods driven in the ground or brass disks set into concrete piers [Brown and Eldridge, 1962, pp. 11-66].

Control points are also physical objects. These, like monuments, may be artificial or natural, and look much like monument points. But they are used by cadastral cartographers to relate a number of monuments to a common reference framework. They may or may not be associated with property survey monuments; it is not usual procedure for a property surveyor to measure to them unless they are so associated. A property surveyor's goal, in surveying a parcel, is to locate evidence in the immediate vicinity, not to relate a number of parcels in a wide area. This is, however, precisely the goal of a cadastral cartographer, who thus needs to determine relationships between control points and monuments.

Survey points are physical objects. They may be marked temporarily in the field by nails or stakes, or they may be marked by something as permanent as the iron rods used for monument points. The difference between survey points and monument points is simply that monument points are specified in a deed, while a survey point — as a point — is not.

Survey lines are not physical objects. This forms an interesting asymmetry with survey points compared to monument points and lines and property points and lines.

Monuments, whether linear or punctual, are physical objects — iron rods, building walls, etc. Property points and lines are both abstract. But survey points are physical objects, while survey lines are abstract. The reason for this has to do with the fundamentals
of geometry and their expression in surveying instruments and practice.

The axioms of plane geometry are symmetrical for lines and points. That is, each axiom can be formulated either in terms of points or lines. Either geometry is equivalent, but the expression of each would be quite different. We can get an idea of the difference by imagining a surveying practice that used physical lines to establish location and position. These lines might be imagined as thin straight or curved beams which could be laid out and secured on the ground. Establishing a position would entail locating intersections of beams, measuring distances along beams, and placing new beams at specified places on and angles to the beams. Notice that with this "reversed" surveying practice that it is now the points — the intersections of the beams — which are abstract, non-physical objects.

Property points are abstract objects with no tangible existence. Commonly, they are called for in a parcel description as being located at a monument; so wherever the surveyor finds the monument in the field (unless, for example, there is evidence the monument has been disturbed) that is where the property point is. But a property point can also be located where there is nothing physical to mark the spot. For instance, a property point may be located a certain distance along an unmonumented line which is defined to be located a certain distance from, and parallel to, the line between two specified monuments.

Property lines are bounded (at either end) by property corners, and are themselves boundaries between parcels. They are usually straight lines between property corners, but may also be simple curves (arcs of circles), complex curves (segments of spirals), or irregular lines corresponding to or parallel with natural linear monuments such as ridge lines or stream centers or banks. (Right-of-way lines and easement lines are special types of property lines. Right-of-way lines divide publicly owned land used for passage from other ownership, while easement lines separate areas of "total" ownership rights from areas of restricted rights.)

Cadastral objects may or may not be called for in the property description. As
Table III shows, monument points, monument lines, property points, and property lines are called for. Survey points are not called for, but they are used, by both property surveyors and cadastral cartographers, to locate objects called for in a property description. Control points are used by cadastral cartographers to locate a group of parcels on a common framework. (Measurements used to locate survey points may be specified in a description, or a survey which is referenced in a description, as evidence to locate monuments or property boundaries. If the points themselves — the physical objects — are referenced, they are not survey points, but monuments.)

Objects may be capable of having their position measured by an observation in the field, or they may only be capable of having their position or shape inferred from such an observation. This characteristic is closely tied to whether an object is physical or abstract — no abstract object can be directly observed.

In addition, the shape cannot be directly observed for one type of physical object: linear monuments. As with the survey line, this characteristic is a result of the fundamental structure of geometry. Surveying (and photogrammetric) instruments cannot measure lines. They locate points, and their operators infer the shape of the lines between the points. Thus, while a contour line or stream bank has physical existence and can be found in the field, its shape is always "captured" as a set of points, more or less widely spaced, along with some rule, express or implied, for describing the shape between the points.

Finally, an object's location may be "given" or derived. Only control points have a "given" location; all other cadastral objects in a cadastral map derive their location from those of control points.

It is probably not possible to give an exhaustive list of the possible ways in which the location of a cadastral object can be derived in a property description. A general pattern will be suggested, and a number of configurations will be given. But the present writer's acquaintance with deed descriptions suggests that scriveners are capable at any time of
coming up with a completely novel relationship.

All such relationships must, however, fit into a general pattern: there must be a reference object or objects; there must be a rule or procedure to apply; and there may be values to use in applying the rule. Control points, with their "given" locations, form the only exception to this pattern.

It was noted above, for instance, that property points are frequently located "at" monument points. That is, for such a property point, the reference object is a particular monument point; the rule is one of identity of location; and there are, for this rule, no parameters — no values to be supplied when the rule is invoked.

Property points are also frequently located at a certain distance along a survey, monument, or property line. In this case, the reference objects are the particular line and a starting point on the line (often an endpoint of the line); the rule is one describing the operation of measurement from the starting point along the line; and the value would be the distance along the line.

A point may be located by what is known as an angle and distance tie. In this case, there are again two reference objects: a point, from which the distance is measured, and a directed straight line, relative to which an angle is measured (usually the reference point is at the origin of the directed line). The rule describes the angle-and-distance operation. And there are two values to be supplied, the angle and the distance.

Other location possibilities for points include: at the intersection of two lines (which may be straight, simple curves, complex curves, or irregular streamlines); at the intersection of two or more radii; at the intersection of two angles; and so on.

It should be noted that the points involved here could be physical or abstract objects. It is probably more common, in fact, for a survey monument point or line (a physical object) to be located by a fairly involved chain of relationships, while a property point or line is simply located "at the monument."
Lines may be thought of as deriving their location from their boundary endpoints, but they often possess shape characteristics which need to be derived from other values and/or related to the location of other objects. To know the shape of a simple curve (an arc of a circle), for instance, requires the value of the radius of the curve and the direction in which the curve trends. A line depicting the shape of a river bank will probably know what its shape is by referring to the locations of points along the edge as measured by a surveyor or, perhaps, by a photogrammetrist.

In summary, then, the cadastral cartographer locates objects on a cadastral map by using knowledge of how property surveyors locate property boundaries in the field and how property descriptions are formulated in the office. Fitting together numbers of surveys and descriptions of individual parcels into a coherent whole is often based on the measurement of relationships between control points and property survey monuments.

AN OBJECT-ORIENTED INFORMATION STRUCTURE OF CADAstral LOCATION

It was noted in the discussion of the “information reality” of the cadastral cartographer that it is probably not possible to completely specify the list of operations by which the location of cadastral objects can be determined. Thus, the information structure presented here also cannot have a closed set of object classes. This is indicated in the object class inheritance hierarchy (Figure 9) by the boxes containing ellipses (...). The possibility must exist, in the information structure and in the data structures based on it, for definition of new types of object location methods as the need arises.

Seven basic object classes are portrayed at the top of the diagram. At the data structure level, these classes would contain methods and internal variables needed to display instances on the screen or plotter; an information structure of the entire cadastral layer would show methods and internal variables to access attribute data. For example, the Monumentpoint object class as completely defined would probably store the date the
Figure 9. Object class inheritance for cadastral location
monument was set, the surveyor who filed the survey report, and the type of material the monument was made of, as well as other data items. The cadastral layer stores the land parcel geometry. This includes the topological relationships of punctual, linear, and areal cadastral objects as well as the metrical (coordinate and shape) descriptions focused on here [White, 1984]. The topological relationships are stored in superclasses of the cadastral object classes — ZeroCell for MonumentPoint, PropertyPoint, ControlPoint, and SurveyPoint; OneCell for MonumentLine, SurveyLine, and PropertyLine.

The subclasses of all these classes (except the ControlPoint object class, which has no subclasses) are all various kinds of locationally-determined monument, survey, and property points and lines.

The first part of Figure 10 shows some of the subclasses for the point-type cadastral objects; the second part of the figure shows the subclasses for linear ones. As was mentioned above, the possible number of object classes is left open.

Linear objects are much easier to limit to a closed set, at least conceptually. There are only a few linear objects commonly used as property monuments: fences, building walls, pavement edges, stream edges, and the like, and there are only a few common shapes for these objects: straight lines, arcs of circles, and streamlines.

Survey lines, being non-physical objects, can have more complicated mathematical shapes. Commonly, for instance, transitional curves between straight segments and circular arcs along rights of way are defined as segments of spirals. A "SurveylineSpiral" is indicated in the figure for occasions when points defining such a line have been measured in the field but the line itself is neither marked physically nor referenced as a monument. A MonumentLineSpiral could occur in the case of a wall along a freeway spiral curve.

Property lines are also non-physical objects. The main function of the PropertyLine subclasses which include shape methods is to allow the definition and location of lines called for in descriptions which have not, in fact, been monumented in the field. (More
Figure 10. Template for Controlpoint object class.

precisely, the bounding points have not been monumented and a surveyed line has not been established between points.) A property line which had been surveyed would be stored as a PropertylineAtLine, referencing the Surveyline or Monumentline subclass for its shape and location.

Finally, object class templates are presented for some of the cadastral objects that have been mentioned.

Figure 10 shows the class template for control point object class. Since the present work concentrates on locational relationships, the template doesn't show additional methods and instance variables that would actually be defined for this object class. — e.g. the order of accuracy of the survey establishing coordinates of the point. For present purposes, it is important to note that Controlpoint objects are the only objects which directly store their coordinates. All other cadastral objects derive them.

For example, each of the subclasses of Monumentpoint shown in Figures 11, 12, and 13 stores the different variables used to compute its position upon receipt of the "GetLocation" message. The instance variables store values of measurements of references
<table>
<thead>
<tr>
<th>Class Name</th>
<th>MonumentAngleDistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superclass</td>
<td>MonumentPoint</td>
</tr>
<tr>
<td>Instance Variables</td>
<td>ReferencePoint Angle ReferenceLine Distance</td>
</tr>
<tr>
<td>Methods</td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
:PutLocation theReferencePoint theReferenceLine theAngle theDistance
  (put arguments into instance variables)

:GetLocation
  (gets location of theReferencePoint, direction of theReferenceLine, calculates location using value of theAngle and theDistance)
```

**Figure 11.** Template for MonumentAngleDistance object class.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>MonumentOffsetIntersect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superclass</td>
<td>MonumentPoint</td>
</tr>
<tr>
<td>Instance Variables</td>
<td>Line1 Line2 Offset1 Offset2</td>
</tr>
<tr>
<td>Methods</td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
:PutLocation theLine1 theLine2 theOffset1 theOffset2
  (put arguments into instance variables)

:GetLocation
  (gets directions of theLine1 and theLine2, value stored in Offset1 and Offset2, computes intersection)
```

**Figure 12.** Template for MonumentOffsetIntersect object class.
Figure 13. Template for MonumentOnline object class.

Figure 14. Template for PropertycomerAtPoint object class.
to objects. Thus, the "ReferencePoint" variable in the "PropertycornerAtPoint" object (Figure 14) contains, not the location of the reference object, but the identifier of or pointer to the reference object. When an instance of this class receives a ":GetLocation" message, it in turn sends a ":GetLocation" message to its reference object. The reference object (an instance of a subclass of Monumentpoint, Surveypoint, or Controlpoint) determines its location using means appropriate to its subclass and (eventually) returns the location to the requesting object.

Linear objects need to store both location and shape data; they interact with other objects to determine location. When a MonumentpointAngleDistance object receives a ":GetLocation" message, it sends a ":Direction" message to its ReferenceLine. The
Referenceline (for example, a SurveylineStraight object, the class template for which appears in Figure 15) in turn obtains the location of its start and end points, computes the azimuth, and returns the result to the MonumentAngleDistance object, which then computes its own location. It should be noted that curved-line and streamline types of objects (Figures 16, 17) do not have :Direction methods, and hence cannot serve as reference lines for this type of monument.

Stream-type linear objects are used to model the location behavior of a natural, irregular linear physical object such as the edge of a stream, a ridgeline, contour line, edge of pavement, or break in elevation. Consistent with the idea that none except Controlpoints directly store their coordinates, the points identified in the object store their locations
Figure 17. Template for MonumentlineTopoStream object class.

According to the way they were derived. Usually, this would be by digitization from a photogrammetric model, by measurements from a topographic survey, or by digitization from a topographic map (where the original survey notes were unavailable), or aerial photograph. The object storing locations for a line digitized from a photogrammetric model would identify the control points used to orient the model and store the model transformation parameters and the list of photo coordinates of the line. Similar objects would store orientation and offset information for lines derived from maps or topographic survey notes.

Figure 17 shows the object class template for a MonumentlineTopoStream. For this object, the points sampled along the linear monument (CD in Figure 18) were measured as offsets at right angles to a surveyed line (AB in Figure 18) — the ReferenceLine in the
Figure 18. Survey location method modeled by MonumentlineTopoStream object

class template. The OffsetList identifies the points (SurveypointOffsetDistance objects).

While ambiguities can exist in deed descriptions, and conflicts can exist between deed descriptions, it is impossible for a real-world object (e.g., a monument or survey point) to have more than one location at a time. And since all legal objects must be located in the real world by reference to physical objects, it is at least theoretically possible to determine one most likely or reasonable location for a legal object, even if it is not constrained, as a physical object would be, to one location. The task of a cadastral cartographer is to resolve the ambiguities in descriptions and surveys in order to develop a reasonable representation of the real-world situation. This task requires sifting through the locational constraints for each object to find the one that has priority. For example, is the point "on a right of way"? Then if the right of way is shifted, the point moves also. Or is the point under consideration a monument, the location having been measured in the field?
Then the method by which the location was determined is (usually) the controlling constraint.

This "information reality" assumption — that one and only one location will be true for an object at any specific time — is reflected in the information structure, which allows one and only one locational rule (with its combination of reference object(s) and parameter values) for each object class.

Such a restriction in the information structure makes it impossible to build circular location dependencies. In order for that to happen, some object class would need to be capable of being located by more than one rule.

Being restricted to a single locational rule does not, however, mean that an object must be related to a fixed number of objects. The clearest example of this occurs with closed surveys. Points within a closed survey have a fixed location relative to each other. Absent a change in the measurements between points, they move as a unit if a new location (relative to objects outside the survey) is found for one of the points.

A ClosedSurvey object would have much in common with the StreamLine object discussed above. That is, its internal variables would include a list of points, and identifiers of objects from which to derive the values used to transform the points' relative locations into global ones — placing the surveyed points into the common reference framework. (The details of this transformation method would differ, depending on whether the closed survey was related to the global coordinate system by one, two, or more reference points.)

Finally, the information structure presented in this chapter may be compared to one version of a data-structure level description in Appendix B. This Appendix contains, in addition to listings written in a particular object-oriented language, output from a working program in which instances of some of the object classes discussed here "know" their location by reference to other objects.
This chapter examines the procedures that could be used to construct and upgrade the cadastral layer of an assessor's land information system with an object-oriented information structure. First, some background information is presented to make more understandable the differences between procedures now used to construct (and reconstruct) manual or hardcopy cadastral maps (and the data stores designed with these maps as a model), and the procedures which would be used with an object-oriented information structure. Following this background, an example area of land is considered in terms of how its cadastral locational data would be entered into such a data store — how it would be built up using available surveys, plats, and deed descriptions.

BACKGROUND: PRESENT PRACTICE

In discussing map construction, the author of the Manual of Cadastral Map Standards, Concepts, and Cartographic Procedures [ODOR, 1981] begins with a series of definitions and then outlines a step-by-step process to be followed in developing a base control map of a section-wide or larger area (italics in the original):

*Base Control Map.* A graphic-geometric network of primary survey quadrilaterals that serve as a base: for secondary surveys, for the lines and points to be plotted from instruments of conveyance and for the natural and cultural features to be supplied by aerial photography. Generally synonymous with basic map control, control map base or control.

*Map Life.* The effective life of any map. In cadastral cartography it is that period of time when a map accurately represents the latest surveys, aerial photos and deeds. It is measured from the time that the original map is completed to that point in time when it will no longer accommodate new property segregations, new surveys, new photogrammetric detail or new parcel
numbers. . . . It is based on the principle that the permanent map does not exist.

**Maverick survey.** A survey that is not in harmony with abutting surveys. The term does not mean that the survey is in error. In fact, new surveys are often labeled as maverick surveys because they are executed at a greater degree of precision than were the abutting surveys; hence, the measurements of the older surveys do not conform to the later [sic].

**Primary Survey.** A survey that provides controlling data for bridging the section and township corners and the points, monuments, and lines of other primary surveys.

**Secondary survey.** A survey that does not provide measurements that could be used in map control. A survey that is subordinate to—or based on—a larger survey. A good example would be a survey of a lot in a subdivision, the subdivision survey would be the primary survey; the survey of the lot would be a secondary survey. [ODOR, 1981, p. 334-345]

All control points should be converted to the N, E state plate coordinate values (or the x and y values) so that they may be plotted from the coordinate grid you have constructed on the control sheet. All plotting should begin with what you consider as the most reliable primary survey data. Plot only by measuring the x and y distances from the coordinate tics. **Do not plot by bearing and dimension at this time.** Plotting should follow this order:

1. Locate coordinated section corners....
2. Locate the stations of the national control net.
3. Plot the transmission line points and important ties to 1/4 corners, 1/16 corners, Donation Land Claim corners, subdivision corners, railroads, road intersections, etc.
4. Plot the traverse of the state highways and secondary highways; locating important ties, intersections, etc. Also plot the surveys of roadside parks and stockpile sites.
5. Plot the traverse of railroads; including sidetracks and station site survey data.
6. Plot county road centerlines (beginning with the latest road surveyed).
7. Plot any major pipeline or ditch traverses and ties to important points.
8. Plot subdivision points and lines; beginning with the largest and most recent subdivisions.
9. Plot the governmental (federal, state, county and city) and private surveys; beginning with the largest and most recent. [ODOR, 1981, pp. 345-349]

At this state of the map construction operation it is essential to complete the section line grid; including meander lines of rivers and lakes, . . . Donation Land Claim corners and lines, and patented mining claims. . . .

Completion of the control map involves using photogrammetric methods to plot unsurveyed rivers, creeks, lakes, ponds, ditches, major logging roads and other major roads. [ODOR, 1981, p. 349]

After the control sheet is developed, a lift-off or layout sheet is made by tracing a portion of the control information. This layout sheet becomes the basis for the final individual section, quarter-section, or sixteenth-section map:
To complete the cadastral map it is now necessary to plot the supplemental lines, points and data of the right-of-ways [sic], subdivisions and primary surveys. Next, plot the secondary surveys. ...

At this point in the map finalization operation the deed boundaries should be plotted. In this step we are developing the inventory of real property. ...

[ODOR, 1981, pp. 349-350]

It is apparent in the process described above that the order in which the location of objects is defined is extremely important. The first objects defined for the map provide the framework into which subsequent objects are fitted. This is as true for a land information system with a data store design based on this model as it is for a paper map. The order of locational definition builds up a hierarchy which reflects the degree of trust the cartographer has in the reliability of location for each object. It builds the map or data store in a sequential process, but instead of proceeding from the past to the present, as the data would come to the system in the normal course of events, it must proceed from the present to the past. The less accurate (generally older) information must be fitted to the more accurate (generally newer) information, rather than the reverse. And since the objects on the map sheet (or in the data store) are “dumb” about any locational information other than their global coordinates, it is necessary to look elsewhere to find which objects should be “held” and which may be relocated during the location-updating process.

After the cadastral map is constructed, new information will continue to come in: new partitions, resurveys, corrections to control point coordinates, etc. New surveys, especially, fall under the classification of “maverick surveys” mentioned above. In a manual map system, these changes are placed on the existing map as far as possible, but at some point the new measurements may have to be so distorted to fit into the older ones that the map (at least as far as the cartographer is concerned) has lost its credibility. The “map life” of this particular document is over, and it is time for reconstruction.

With an object-oriented information structure, the hierarchy of spatial relationships is preserved in the data “known” to the objects in the data store rather than in the order in which the objects were drawn on the map sheet or entered into the data store. Obviously,
this means that it is possible to be much more flexible about the order in which such data are entered. It may be advantageous in map construction, for instance, to follow the historical record, "following in the surveyors' footsteps" to get an idea of what they were thinking as they located monuments, rather than always using the newest large surveys first. In any case, when upgrading a map, we have no choice if we wish to avoid "reconstruction." In updating locational data we are always going from older to newer, rather than from newer to older, from a less complete to a more complete state of knowledge.

Unfortunately, even the newer state of knowledge is still fragmentary. For instance, it is extremely uncommon for the cadastral cartographer to have access to private surveyors' notes. Subdivision plats, which simply give bearings and distances between boundary points, do not give any indication of how the measurements were actually made or, usually, which monuments were found, rather than set by the surveyor for the plat.

This points up how much of the locational data stored by the cadastral cartographer actually results from a series of judgement calls, relying on a knowledge of surveying practice in the area. In the course of making these judgements, a number of assumptions inevitably are made — and, it is to be hoped, refined in the light of further information. Fortunately, an object-oriented information structure should allow revision and redefinition more easily than one based on the model of a paper map.

Taking A Map Apart
For Storage

Sample Area Description

The example area (map in map pocket) is in the southwest quarter section of Section 25, Township 1 North, Range 2 East of the Willamette Meridian. It is a suburban residential area of Northeast Portland, Oregon, with several large-acreage unrecorded subdivisions (one of which dates back to the turn of the century); a dozen or so
single-family residential subdivisions dating from the post-World War II period; and a
small-tract retirement-style subdivision of the 1980's. (In fact, the sequence forms a
veritable archaeological layering of land uses worthy of its own thesis on historical urban
cartography.) The remainder of the quarter-section is filled in with a number of acreage lots
with individual metes-and-bounds descriptions, a number of which are based on or
reference recorded property surveys. In addition, there are a number of roads; most, as
county roads, have their own descriptions and surveys. There are two control points on
this quarter-section (the southwest section corner and the quarter-corner halfway between
this point and the southeast corner), and one that, although not on the map, fixes the
direction of the west boundary of the section.

Definition of Objects

The Northwest Sector. As mentioned above, an object-oriented information
structure would allow a large amount of flexibility in the order of entry of data into the data
store. Thus it would be possible to follow the same order as is called for in the Oregon
Department of Revenue's Cartographic Manual guidelines — but in order to illustrate how
map maintenance would work, let us imagine putting in data from the oldest subdivision
first (the unrecorded Rose Park Acres), especially noting the location of the southeast
corner of the Wilkes D.L.C. (Donation Land Claim).

The first three objects defined are the Controlpoints at the southwest and northwest
section corners and the quarter-corner on the south line of Section 25.

The next three objects are MonumentPointOnLine objects, the reference line being
defined between the southwest and northwest section corners, the distances being given on
the plat. They are: 1) the quarter-corner on the west line of the section; 2) the southwest
corner of Rose Park Acres; and 3) the intersection of the south line of the Wilkes D.L.C.
with the west line of the section.

Finally, the southeast corner of the Wilkes D.L.C. is located with a
MonumentPointAngleDistance object from the intersection of the south line of the D.L.C. with the section line.

Now the plat of Clifgate, blocks 19-23, is put in. This is treated as a closed survey, so the first step is to compute and adjust the boundary points relative to each other. (Although the choice of an origin point is arbitrary, it is simplest to start and close the boundary traverse at the intersection of the south line of the Wilkes D.L.C. and the section line.) After a series of MonumentPointOffset objects (with their coordinates offset relative to the origin of the survey) has been defined, the next step is to define a ClosedSurvey object with the capability to transform this particular series of points to the proper global location. It needs the (in this case, single) reference point to determine translation values and a reference line to determine the rotation values (since there is only one global reference point, the survey is not scaled.) The point reference is the intersection of the south line of the Wilkes D.L.C and the section line; the reference line is the section line (another, corresponding, line is defined between two of the points on the survey).

It should be noted that property surveyors do not commonly use astronomic or geodetic north as a basis of bearing. It was more usual in earlier times to use magnetic north — which, of course, changes over time — or a line between found monuments with a recorded or, perhaps, an assumed bearing. When direction is carried to the other side of a survey, errors will accumulate and be adjusted out in the final survey report. It is thus more likely than not that adjacent surveys will have different bearings for the same line. In any case, all relative bearings must be converted to a common basis of direction ("grid north"), just as all relative locations must be converted to a common coordinate system.

We note that the surveyed point on the new subdivision for the southeast corner of the Wilkes D.L.C. is in a slightly different position than the monument located by the unrecorded subdivision survey. The easiest way to handle this is to delete the old MonumentPointAngleDistance object and give the new MonumentPointOffset object the old
object's identifier. Thus any locational references to the southeast corner of the Wilkes D.L.C. will find the location as most recently — and accurately — defined.

After defining the location of boundary monuments for subdivisions, the lines depicting the actual boundary may be defined between them. These are usually straight lines or arcs of circles. The interior points of subdivisions are probably most efficiently stored as additional “closed survey” points — that is, points with a known offset relative to the origin of the subdivision survey, allowing the coordinates to be transformed wholesale when the area is displayed, plotted, or used in computations. While it would be possible to define each internal point by some survey procedure, it does not, in general, seem useful (or even possible) to guess at exactly how the surveyor proceeded in laying out the subdivision — and in most cases, there are a number of possible ways it might have been done.

The Southwest Corner. Let us now look at the southwest corner of the area, again imagining what would happen if we had to upgrade older data with newer.

The unrecorded plat for Rozella Acres does not contain enough data to treat it as a closed survey — two crucial angles are missing from interior corners — but the dimensions given fit plausibly enough for it to be considered (as the title assures us) as being the “West 1/2 of the SW 1/4 of the SW 1/4 of Section 25.” We assume the southwest corner of Rose Park Acres (already defined) to be the northwest corner of Rozella Acres and locate the northeast corner of Rozella as a MonumentPointOnLine using that point and the south line of Clifgate as reference point and line, respectively. Similarly, the southeast corner is located along the south section line.

However, according to the next large survey made in this area (the plat for “Norma”), the monuments for Rozella were actually put in two feet west of the west section line — so the definition of location for those two objects changes (and the northeast and southeast corners move as a result, even though their definitions do not change).
After this correction is made, the MonumentPointOffset objects for "Norma" are defined and related to the global coordinate system through a new ClosedSurvey object citing the section corner and the south line of the section as reference objects.

**The Middle of the Map.** Plats or closed surveys adjacent to each other will often describe the location of the same monuments. In these cases, the most recent or accurate measurements should be "held" (that is, used as reference objects) and the earlier or less accurate points transformed to fit. A Helmert, or linear conformal transformation, can be used with two or more reference points (although other adjustment methods may be preferred for the case of two reference points).

In the middle region of the example area, Larthel, blocks 6 and 7, is fitted to the later Elmgate. The only point in common is at the northeast corner of the older subdivision; the common line is used to determine the rotation value. Larthel, blocks 4 and 5, fits precisely to blocks 6 and 7 (considering that the two plats were surveyed by the same person within one year, it would be a matter of concern if they did not fit exactly). However, the oldest plat of this group, Larthel, blocks 1-3, has one distance about 0.25 foot too long to fit without adjustment. It should use the points defined in the later surveys, basically along its west line, as reference points. (If we were developing this area of the map in historical sequence, the already-defined points in Larthel, blocks 1-3 along its west line would be replaced by points measured in the newer surveys.)

**Roads and Streets.** Roads outside of subdivisions are frequently described in surveys as beginning at a fixed location and ending at another. The segment of Halsey Street in the example area, for instance, has a set of "road notes" describing the distances and angles between centerline points from the section corner to the quarter corner. Treating the road notes as a closed survey and adjusting the measurements between the fixed end points gives reference points for the right-of-way lines as well as for surveys like the plat for "Becky Lynn."
The segment of 148th Avenue south from its intersection with Sacramento Street to Halsey Street is similarly described in road notes. In this case the large, recent, and accurate survey on which the "Summerplace" plat is based locates a number of monuments on the east right-of-way line, so it is best to reference the road survey from these points. That is, the west line of the right-of-way would be defined as located forty feet from the centerline points, which would be defined as located according to the distance measured in the survey from the monuments found (but the east line would be located forty feet from the centerline points; it is a property line, not a monument).

Acreage tracts. The remaining parts of the mapped area are filled in using deed descriptions or, where available, secondary recorded surveys. Since a description is an attempt to articulate the intent of the parties conveying ownership of a parcel of land, they are generally reliable in describing the locational relationship of property boundary objects, even if they may not close mathematically. Property surveys of such parcels, or within subdivisions, also frequently do not close mathematically (commonly omitting a crucial linear or angular measurement) but, as with deed descriptions, generally provide enough information to map the parcels involved. Thus, the objects defined for these parcels will usually be property points, monument points, and property lines deriving their location from one or two other objects using a single rule, rather than a series of points mutually related in a closed survey.

Numbers. Table IV shows the number of objects of each general type in the example area.

There are twelve subdivisions or closed surveys in this quarter section, with 83% of the monuments or property corners. This is encouraging news in terms of considerations of storage amount or complexity of computation required to handle densely-developed areas. For closed survey points, location-finding methods are not complex and the amount of storage required for each point is relatively small.
### TABLE IV

**NUMBER OF CADAstral OBJECTS IN THE EXAMPLE AREA**

<table>
<thead>
<tr>
<th></th>
<th>monuments and prop. pts.</th>
<th>property lines</th>
<th>right-of-way lines</th>
<th>subdivision lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>subdivisions and closed surveys</td>
<td>833</td>
<td>1063</td>
<td>235</td>
<td>116</td>
</tr>
<tr>
<td>acreage tracts</td>
<td>174</td>
<td>201</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>totals</td>
<td>1007</td>
<td>1264</td>
<td>283</td>
<td>116</td>
</tr>
</tbody>
</table>
CHAPTER VI

SUMMARY AND FUTURE DIRECTIONS

Following a summary statement, this chapter presents a short examination of two possibilities opened by an object-oriented approach to design of a land information system. The first is that of implementation of the information structure described here in an operating land information system; the second is possible further development of the functions available to such an LIS through extensions to the capabilities of object-oriented languages.

SUMMARY

This thesis has presented an information structure for a portion of the data store subsystem of an assessor's land information system — the cadastral layer, the portion concerned with the location of objects in parcel descriptions and recorded surveys. The information structure is based on the object-oriented paradigm of recently-developed computer languages, and as such is relatively easy to translate, in one direction, into a description (as understood by the makers of cadastral maps) of the behavior of these entities, and, in the other direction (as understood by the hardware and software of a land information system) into a data structure using objects with their methods and internal variables.

The information structure allows the capture of locational relationships of objects referred to in deed descriptions and property survey documents so that those objects can be updated for location in the data store without the necessity to refer to the original source documents.
Two different descriptions were given of the behavior of cadastral objects: the first at the "information reality" level of the cadastral cartographer, the second at the "information structure" level of data store design. Finally, an example area of a cadastral map was examined to give some clues as to the procedures that might be used in the production of such maps using such a system.

IMPLEMENTATION

Once a description, however sketchy, exists of an information structure that will reflect the cadastral cartographer's information reality, actualization of such a structure has moved a step closer. If the basic premise is correct that a better representation of cadastral location will lead to more efficient production and maintenance of cadastral mapping data, such actualization should be a positive occurrence for a number of actors at the local government level, not restricted to the cartographer or assessor's office. But exactly how does one move from a description such as the one presented to one that "works"?

The most straightforward way would use an existing geographic information system that allows the definition of object classes for its data store: an object-oriented GIS. Given (or loaned) such a system, translation of the information structure described here into the data-structure level of description would proceed in the same fashion as any other specification of a data structure.

If a GIS that allowed definition of object classes in the data store were not available, one could start farther back in the development process. It would involve starting with an object-oriented database handler to define a data store, and linking it to already defined (or defining from scratch) the data selector and data massager subsystems. Even more formidable would be the task of starting with an object-oriented language and defining the database handler. It could be done, but (since one of the great virtues of object-oriented design is the ability to use and modify already-defined applications with little concern for
strange side effects) it seems more expeditious to use available applications that can be adapted to the purposes in mind.

FURTHER RESEARCH

It should be apparent from the description of the procedures used in Chapter V that an LIS using the information structure described here relies on the ability of the cadastral cartographer to abstract cadastral spatial relationships from deeds and surveys. Work on expert systems suggests that this ability can be emulated with a set of rules or constraints to be applied to particular situations [Michaelson and others, 1985]. Future cadastral mapping systems could be designed to suggest solutions to locational problems from text input of parcel descriptions and digitization of survey records. Such a system might be envisaged as operating in a fashion similar to that of medical-diagnosis or geological-exploration expert systems, which present an interactive dialog with the operator to present and refine possible solutions. For solving cadastral location problems, this dialog should undoubtedly include graphic, as well as textual, communication.

Programming by example would be another development beyond the model presented here that would greatly facilitate definition of new object classes. Using this capability, a user constructs a “type object” from which the system compiles code describing its behavior.

It will be noted that a number of the object subclasses in the information structure description use the same locational methods as those of other subclasses of different superclass objects. For instance, there is a “MonumentpointOnLine,” a “SurveypointOnLine,” and a “PropertypointOnLine” object class. Each object’s superclass displays differently and stores different attribute information, but the method used to determine location in each of the subclasses is identical. This actually amounts to a duplication of some of the code used to specify the data definition (in practice, this does not
cause much additional effort in writing the code, since it can be copied from one object class definition to another; but it does use additional storage space to hold duplicate location methods).

**Multiple inheritance.** where the target data structure allows it, permits development of a more elegant information structure. At the present writing, only one language for the Macintosh computer implements multiple inheritance, although this feature is not uncommon in versions of these languages running on other computers [Schmucker, 1986]. That is why the information structure presented here uses single inheritance. Simply put, multiple inheritance allows a subclass to point to more than one superclass. In Figure 19 the white blocks correspond to the superclass “point” object class definitions (Monumentpoint, Surveypoint, Propertypoint), while the black blocks correspond to the locational methods (for example, a generic PointOnLine object class). Each of the subclass objects points to one superclass defining its appearance and attribute data characteristics, and to one superclass where its locational method may be found.

![Figure 19. Multiple inheritance of object characteristics.](image-url)
All these capabilities are implemented in ThingLab, an extension to Smalltalk [Borning, 1981]. This graphic simulation laboratory adds to Smalltalk the construct of "constraints," which specify relations that must be maintained, and the ability to keep track of part-whole relationships. These present the possibility of alternative ways to formulate an information structure from a given information reality, as object-oriented languages allow alternatives in structure not available to procedural languages.
SOURCES CONSULTED


APPENDIX A

OBJECT-ORIENTED LANGUAGE DATA
STRUCTURES AND NOMENCLATURE
[PASCOE, 1986]
The class data structure shown in Figure 20 contains information needed to construct and use object instances. The first and second fields indicate the number of instance variables and their names, respectively. The names are needed so that when methods are compiled, the instance variable names can be identified as referring to instance variables.

The third field of the class data structure is a flag that indicates whether an instance
contains a variable number of indexable instance variables. Arrays are examples of objects that may have a variable number of instance variables. The instance variables of an array are accessed by indexing instead of by names. Since arrays of different sizes will respond to the same protocol (the set of messages to which an object responds), it does not make sense to have different classes for each array size.

The class variable field contains storage for variables shared by all instances. Class variable names are also stored here. Class variables can be convenient for storing information that should be common to all instances of the class. Not only does this save storage, but it greatly simplifies changing shared information. Otherwise, all instances of a class would have to be located to change an instance variable.

The method dictionary contains pairs of selectors and methods. When a message is sent to an object, the class’s method dictionary is searched for a method to execute.

The superclass field points to the superclass of this class. The superclass is another class data structure that has all the same fields as this class. Other instance variables, class variables, and methods may be defined in the superclass.

If during a message-send a method is not found in the instance’s immediate class, the superclass method dictionary is searched. If it is not found there, the superclass’s superclass method dictionary is searched, and so on, until the class at the root of the hierarchy (most often called Object) is searched and found not to have a matching selector. In this case an error is returned.

Note that each instance contains a pointer to its class and a field indicating the length of the instance. The pointer to the class is needed so that the class, and therefore an object’s methods, may be easily located when the instance is sent a message. Storing length information in the instance aids the storage manager and is useful when dealing with indexable instances.
APPENDIX B

DEMONSTRATION PROGRAM
LISTINGS AND OUTPUT
This Appendix is intended to provide a proof of concept, using a demonstration program, of the ideas discussed in the present work. In addition, it illustrates what object-oriented code (in one particular language) looks like, as compared to the object-oriented information structure description using object class templates and inheritance diagrams. The listings are written in Neon™, an object-oriented language for the Apple Macintosh™ computer. The output consists of screen dumps (“snapshots” of the monitor display) taken at various stages of the running of the demonstration program; they are unaltered except for having been cropped to fit the margins for this document format.

Listings 1-4 are written in what might be called “Neon pseudocode.” They illustrate some of the general features of this Forth- and Smalltalk-flavored language, with comments (delimited by parentheses or, alternatively, by backslash (\) and carriage return) explaining some of its peculiarities of syntax. Foremost of these results from the stack implementation, derived from Forth, in which arguments for messages precede the selector, and are processed within a method in reverse order to that in which they appear in the message. Thus, the message \texttt{a \ b \ c \ do: theThing} would cause \texttt{theThing} to perform the method \texttt{do:}, taking \texttt{c} (the top object in the stack) first, then \texttt{b}, and finally \texttt{a}.

Listings 5-16 are the working code for the program illustrated in the screen dumps presented in this Appendix. Since they are working code, these listings will necessarily appear more arcane. The author regrets this circumstance, but wishes to assure readers that it’s not that bad once you get used to it.
---Listing 1---

:CLASS Object <Super Meta

( This object class has general methods, like GET: and PUT:, which allow the storage of data into internal variables. All objects in an object-oriented language inherit, directly or indirectly, from the Object class )

;CLASS

---Listing 2---

:CLASS ZeroCell <Super Object

Point Location

---Listing 3---

:CLASS Control <Super ZeroCell

( theX theY -- )

:DEFINE:

Put: Location

;M ;CLASS

---Listing 4---

:CLASS Meta <Super Object

( The Meta object is the superclass for all objects that include, in addition to the methods of Object, a "moo:" method.

;CLASS

---Listing 5---

:CLASS Thing <Super Meta

( The Thing object is a subclass of Meta. It inherits all the methods of Meta, as well as the "moo:" method.

;CLASS

---Listing 6---

:CLASS Windows <Super Thing

( The Windows object is a subclass of Thing. It inherits all the methods of Thing, as well as the "moo:" method.

;CLASS
---Listing 4---

```small
CLASS MonAzDist <Super ZeroCell

(This type of monument locates itself by azimuth and distance from another ZeroCell object.)

Angle Real

Azimuth Distance

Ptr Ret'Pt

(theRetPt theDistance theAzimuth --)

:M DEFINE: Put: Azimuth Put: Distance Put: Ret'Pt ;M

("Define:", like the method of the same name in the Control object class definition, takes input values and stores them in internal variables. Instead of storing x,y coordinates, this method stores the values used to compute the coordinates.)

(-- theX theY)

:M LOCATE:

(computes the sine of Azimuth times Distance, adds to x-coordinate of Ret'Pt, puts on stack; computes the cosine of Azimuth times Distance, adds to y-coordinate of Ret'Pt, puts on stack; swaps top two items on stack (otherwise the stack-condition comment above would be "theY theX"));M

;CLASS

---Listing 5---

(The following messages set up a full-screen window for drawing objects. Issue the message select: fWind to get the system window back to use for displaying text)

rect MapRect
0 20 512 342 Put: MapRect
window MapWind
MapRect " Map" 2 true false new: MapWind

(This "colon definition" sets up a word, "clear" as an alias for a series of messages allowing you to clear the map window. The "set:" message makes the recipient window the active window; the "cls" word clears the active window and places the cursor in the upper left corner.)

: clear set: MapWind cls set: fWind ;

(The first of the following messages makes a list that can hold fifteen objects to be drawn in MapWind. The colon definition of the word "drawmap" will cause Map to send a "draw:" message to its objects.)

15 Ordered-Col Map
: drawmap size: map 0 DO draw: [ i at: Map ] LOOP ;
```
---------Listing 6---------

(This is the superclass object for punctual objects. The method accesses a Macintosh Quickdraw command.)

:CLASS ZeroCell <Super Object
    Point Location
    Var DestBits

:M GETPORT: Abs: DestBits call GetPort 2 +: DestBits ;M

;CLASS

---------Listing 7---------

(The class definition allows the instantiation of Quickdraw bitmap objects, and gives them two methods to perform. The first of the colon definitions allows addition of point coordinates, while the second prints out a text string. There's no real reason for them to be in this particular file except that that's where they wound up.)

:CLASS qdBitMap <Super Object
    Var BaseAddr
    Int RowBytes
    Rect BndsRect

(addr n l t r b ---)
:M PUT: Put: bndsRect Put: RowBytes Put: BaseAddr ;M

(-- addr)
:M BNDGET: Addr: BndsRect +base ;M

;CLASS

:+pair { x1 y1 x2 y2 -- x1+x2 y1+y2 } x1 x2 + y1 y2 +;
: CoordMsg ." The coordinates are ";

---------Listing 8---------

(Array and bitmap sources for control point, monument, and property corner templates.)

4 Array Ctlimage
    : xx to: Ctlimage ;
    hex
    18002400 0 xx
    42008100 1 xx
    81004200 2 xx
    24001800 3 xx
decimal
    forget xx
QDBitmap CtlptSource
    abs: Ctlimage 4+ 2 0 0 8 8 Put: CtlptSource
    \instantiates a four-element array called "Ctlimage" and stores the pattern for a control point

\instantiates a bitmap
\puts the pattern in the bitmap
(Each of the cadastral location subclasses has the same four names for its methods, although the internal definition is different in each subclass.)

:CLASS Ctlpt <Super ZeroCell
  Rect DestRect

:M DEFINE: { xloc yloc -- } 322 yloc - > yloc xloc yloc Put: Location
  xloc yloc -4 -4 +pair putTop: DestRect
  xloc yloc 4 4 +pair putBot: DestRect ;M

  ( -- x y )
:M LOCATE: Get: Location ;M

:M DRAW: set: MapWind getport: self Abs: CtlptSource Get: DestBits
  Bndget: CtlptSource Abs: DestRect 2 makeInt 0
  call CopyBits set: fWind ;M

:M REF: CoordMsg Get: Location 322 swap - swap . . cr ;M

;CLASS
-----------Listing 10-----------

( This class contains the display method for all monuments. )

:CLASS Mon <Super ZeroCell
  Rect  DestRect

:M DISPLAY: set: MapWind GetPort: self Abs: MonSource
  Get: DestBits Bndget: MonSource
  Abs: DestRect 2 makeln 0 call CopyBits set: fWind ;M

;CLASS

-----------Listing 11-----------

( The Monazdist object class computes its location by getting the location of its reference
object and using internally stored distance and direction. Note that the "Draw:" method
calls on the "Display:" method of the superclass. )

:Azmsg ." Azimuth ";
:DistMsg ." Distance from reference object ";
:RefMsg ." Reference point is ";

:CLASS MonAzDist <Super Mon
  Var  RefPt
  Float Az
  Float Dist

  ( theRefPt theAz theDist -- )
  :M DEFINE: Put: Dist dms>r Put: Az Put: RefPt ;M

  :M LOCATE: { \theObj xloc yloc %theDist -- x y } localfloat
    Get: RefPt -> theObj Locate: theObj -> yloc -> xloc
    Get: Dist -> %theDist
    sin: Az %theDist f* float> xloc + -> xloc
    cos: Az %theDist f* float> negate yloc + -> yloc
    xloc yloc
  ;M

  :M DRAW: { \xloc yloc -- } Locate: Self -> yloc -> xloc
    xloc yloc -3 -3 +pair putTop: DestRect
    xloc yloc 3 3 +pair PutBot: DestRect Display: Super
  ;M

  :M REF:  AzMsg Get: Az r>dms ... cr
    DistMsg cr Get: Dist 10 6 f.r cr
    RefMsg Get: RefPt 3 ->name id. cr
  ;M

;CLASS
---Listing 12---

(contains display method for all property corners)

:CLASS PropCor <Super ZeroCell
  Int Mode
  Rect DestRect

:M DISPLAY: set: MapWind GetPort: self Abs: CorSource
  Get: DestBits Bndget: CorSource
  Abs: DestRect Int: Mode 0 call CopyBits set: fWind ;M

:CLASS

---Listing 13---

(this is a class of property corner referenced from a ZeroCell-class object by azimuth and distance)

:CLASS PropCorAzDist <Super PropCor
  Var RefPt
  Float Az
  Float Dist

  (theRefPt theAz theDist --)

:M DEFINE: Put: Dist dms>r Put: Az Put: RefPt ;M

:M LOCATE: \ theObj xloc yloc %theDist -- x y \ localfloat
  Get: RefPt -> theObj Locate: theObj -> yloc -> xloc
  Get: Dist -> %theDist
  sin: Az %theDist f* float> yloc + -> yloc
  cos: Az %theDist f* float> negate yloc + -> yloc
  xloc yloc

:M

:M DRAW: \ xloc yloc -- \ Locate: Self -> yloc -> xloc
  xloc yloc -2 -2 +pair putTop: DestRect
  xloc yloc 2 2 +pair PutBot: DestRect Display: Super

:M

:M REF: AzMsg Get: Az r>dms . . cr
  DistMsg cr Get: Dist 10 6 . f.r cr
  RefMsg Get: RefPt 3 . >name id. cr

:M

:CLASS
(This property corner object is located at another Zero-cell-class object.)

:RefMsg ." Reference point is" ;

:CLASS PropCorIdent <Super PropCor
  Var RefPt
    ( theRefPt -- )
  :M DEFINE: Put: RefPt ;M

  :M LOCATE: { \theObj -- x y } Get: RefPt -> theObj Locate: theObj ;M

  :M DRAW: { \xloc \yloc -- } Locate: Self -> yloc -> xloc
    xloc yloc -2 -2 +pair putTop: DestRect
    xloc yloc 2 2 +pair PutBot: DestRect Display. Super ;M

  :M REF: RefMsg Get: RefPt 3 ->name id. cr ;M

:CLASS

---------Listing 15---------

(This is the superclass for linear objects. The methods access Toolbox commands and data structures.)

:CLASS SurLine <Super Object
  ( These Ivars comprise a PenState structure )
  Point PnLoc ( location of pen )
  Point PnSize ( width, height )
  Int PnMode
  Var PnPatLo
  Var PnPatHi

  :M GET: (ABS) call GetPenSt ;M ( save state here )

  :M SET: (ABS) call SetPenSt ;M ( restore from here )

  :M CLASSINIT: call PenNormal Get: self ;M

:CLASS
Listing 16

(This class definition sets up a straight line, defined as to location by its start and end points. Note how the "Draw:" message calls on the "Locate:" message in this subclass and the "Get:" and "Set:" messages of the superclass.)

:CLASS StLine <Super SurLine
  Var Start
  Var End

  ( theStart theEnd -- )
  :M DEFINE: Put: End  Put: Start ;M

  :M LOCATE: { \theObj -- x y } Get: End -> theObj Locate: theObj
            Get: Start -> theObj Locate: theObj
  ;M

            Set: Super Pack call LineTo
            Get: Super set: fWind
  ;M

  :M REF:     Get: End 3 ->name id. cr
            Get: Start 3 ->name id. cr
  ;M

  ;CLASS

-------------------------
After loading files containing the source code in Listings 5-16, it is possible to instantiate objects of the classes defined. Here, two control-point objects, amos and bette, have been created and their locations defined. The origin for the map window is at the lower left. amos is the diamond-shaped figure at lower left; bette is at the lower center of the map window.

Figure 21. Screen dump of demonstration (1: control points)
Figure 22. Screen dump of demonstration (2: azimuth-distance monuments)

Now three azimuth-distance monument objects have been created, their positions defined, and they have been asked to draw themselves. \texttt{m1} knows that it is on an azimuth of $12^\circ15'30''$ and 230.54 feet from \texttt{amos}; it is located at the upper left in the map window. Similarly, \texttt{m2} knows that it is $89^\circ00'00''$ and 150.35 feet from \texttt{m1} and \texttt{m3} knows that it is located on an azimuth of $310^\circ30'30''$ and 123.00 feet from \texttt{bette}. When these objects are sent a \texttt{draw:} message, they send \texttt{locate:} messages to their respective reference objects and compute their own coordinates relative to the values sent in response.
Figure 23. Screen dump of demonstration (3: property points)

Three property corner objects have been added to the map: two are located by azimuth and distance, like the monuments; the third is "on top of" its reference point. $p_1$ is located relative to $m_2$, $p_2$ is located relative to $m_3$, and $p_3$ is at the same location as $m_1$. 
Finally, two straight lines have been created, and their location defined between \( p_1 \) and \( p_2 \), and between \( p_2 \) and \( p_3 \).
For the grand finale, we ask amos (using the ref: message) for its coordinates, and change them using define:. Having previously added all the objects to the map object’s list, we can type the word drawmap (an alias for a series of messages to map causing it to send a series of draw: messages to its list elements) after clearing the map window. The dependent objects draw themselves in their new positions without having any of their internal variables modified.