# STRUCTURING THE KNOWLEDGE OF CARTOGRAPHIC SYMBOLIZATION - AN OBJECT-ORIENTED APPROACH

**Feibing Zhan**
Center for Computer Graphics and Mapping
Faculty of Geodesy, Delft University of Technology
Thijsseweg 11, 2629 JA Delft, The Netherlands

ABSTRACT

Knowledge-based systems for cartographic symbolization concerned with GIS's output have been suggested by a number of researchers. The structuring of the knowledge with a proper knowledge representation scheme is one of the key issues for the development of such a system. The specific requirements for the knowledge representation scheme are specified. It is argued that the conventional knowledge representation schemes such as rules, semantic networks, conceptual graph, object-attribute-value and frames are not rich and powerful enough to meet the requirements. Then it is suggested to use an object-oriented knowledge representation (OOKR) scheme to construct the knowledge. It meets the requirements and overcomes major problems of the conventional knowledge representation schemes. Further, examples are given to demonstrate the power and flexibility of the object-oriented knowledge representation scheme.

## 1. Introduction

The analysis results of a GIS are usually represented by maps which are, at present, generated through the relevant facilities of a GIS automatically or interactively. The maps are subsequently used as a major tool for decision making and communication. Currently, none of the GIS systems includes mechanisms to ensure the correct use of graphic functions. This may lead to poor use of graphics as GIS systems are widespread, and many of the users of GIS's are not professional cartographers. Indeed, many poorly designed maps can be observed (Muller and Wang, 1990). To solve this problem, considerable investigations on using knowledge-based system technology have been conducted and some achievements have been made (Mark and Buttenfield, 1988; Muller and Wang, 1990; Weibel and Buttenfield, 1988). However, no comprehensive and truly intelligent system has been constructed up to now.

Many issues should be addressed for developing a full-scale map design knowledge-based system (Mackaness and Fisher, 1986; Weibel and Buttenfield, 1988). Among these issues, a proper knowledge representation scheme that can be used to organize the relevant knowledge and facilitate the relevant issues concerned is fundamental for the development of such a system. In the Artificial Intelligence (AI) community, commonly used knowledge representation schemes are rules, semantic networks, conceptual graph, object-attribute-value(OAV) and frames. Each of them has certain advantages and disadvantages. Muller and Wang (1990) used a frame-based knowledge representation scheme for cartographic symbol design. Wang (1990) proposed a conceptual graph based representation scheme for cartographic information representation.

In this paper, it is suggested to use object-oriented knowledge representation (OOKR) scheme for a knowledge-based system for cartographic symbolization concerned with GIS's output (hereafter we will only call it cartographic symbolization). In next section, the specific requirements for the knowledge representation scheme are specified. In Section 3, it is argued why the conventional schemes are not rich and powerful enough to meet the requirements, and why object-oriented knowledge representation scheme is suitable. The representation of the knowledge of cartographic symbolization by the object-oriented representation scheme is illustrated by examples in Section 4. Discussions and future work are given in Section 5.

## 2. The Requirements for the Knowledge Representation Scheme

A knowledge representation scheme is the way in which the facts and relationships of the domain knowledge are organized. It is an issue of key importance for developing a knowledge-based system. General requirements of a knowledge representation scheme can be found, for example, in Luger and Stubblefield (1989). Up to now, there has been no comprehensive knowledge representation scheme which can be used to organize every kind of knowledge. The choice of the knowledge representation scheme depends on the characteristics of the domain knowledge under consideration. The first question then is: what are the requirements of the knowledge representation scheme for cartographic symbolization?

First, let us have a look at an example. Suppose a geographic information system contains information about the buildings of a municipality. A user of the system wants to have the statistical information of each district about area of buildings used for residential and industrial purposes respectively, and the statistic information must be represented on a map. The common procedure for the generation of the map, at present, is: First, one groups the two kinds of information on each district (e.g. by SQL) to produce a data file. Second, one designs the map type and relevant symbols for representing the information based on cartographic symbolization principles. In this step, besides the rules used for decision making, some calculation is often necessary, for instance, to determine the value and size of a symbol. Then the designed map parameters are passed to a package (e.g. GIMMS) to generate the map. If a cartographic symbolization knowledge-based system is attached to the GIS, it is natural and desirable that the knowledge representation scheme could facilitate the issues concerned with the three steps mentioned above.

More generally, the following issues are essential requirements for a knowledge representation scheme when developing a knowledge-based system for cartographic symbolization.

a. Like any knowledge representation scheme, the scheme must have the capabilities to describe the objects and model the relationships between the objects concerned with cartographic symbolization. The objects in cartographic symbolization are those concerning the interpretation of the spatial information to be mapped, the cartographic symbolization principles, and the relevant cartographic semiology.

b. An important feature of spatial information, and the relevant cartographic symbolization principles is their organization into class hierarchies (e.g. Egenhofer and Frank, 1990; Muller and Wang, 1990). Thus the ability of the knowledge representation scheme to represent the inheritance between a class and its instance objects, and between a class and its superclass is essential.

c. As it is believed that the development of a map design knowledge-based system should be started from a limited domain (Muller and Wang, 1990), and thus knowledge may then be gradually acquired in an "amplified intelligence" strategy (Weibel and Buttenfield, 1988), it is desirable that the knowledge representation scheme should be well structured and be able to support modularity and reusability. Hence, when the size of the knowledge base increase significantly, the knowledge is still manageable, and can be extended and reused.

d. When the knowledge base grows and changes, consistency checking becomes important. Moreover, judging from the issues concerned with map design knowledge-based systems (e.g. Muller and Wang, 1990), one can see that map design and generation are problems that mix logical deduction, rule-based inference, and procedure execution (e.g. graphics generation). The solution of these problems demands a knowledge representation scheme that effectively combines rules and procedures, and provides a vehicle for implementing graphics I/O, consistency checking, and interactions between objects.

e. When using GIS, spatial information to be mapped is usually from the database of a GIS, this information is then used for deduction, reasoning and map generation. Therefore the knowledge representation scheme should not only be able to support data input through consultation, but also be able to facilitate automatic feeding of data from a spatial database. This should be considered as an essential feature of the knowledge representation scheme.

These issues may be partially addressed by combining existing technologies such as database, conventional knowledge representation scheme and mapping packages (e.g. Muller and Wang, 1990). However, what is desirable is that the issues could be accommodated by a knowledge representation scheme in a uniform way. We will see how object-oriented approaches can be used to facilitate the issues.

## 3. Why an OOKR Scheme is Suitable for Structuring the Knowledge

We will see, in this section, why the conventional knowledge representation schemes can not meet the requirements discussed in Section 2, and describe the promises of the object-oriented knowledge representation scheme.

### 3.1 Object-oriented knowledge representation

Follow Luger and Stubblefield (1989), and Meyer (1988), an object-oriented knowledge representation scheme may be defined as the organization of knowledge as structured collections of abstract data type implementations. In this scheme, everything is defined as an object or system of objects. An object can be defined as an independent entity represented by some declarative data and a set of methods (such as routines and rules) that operate on the object. Relationships between objects and the overall problem specification are implemented as messages between objects. In addition, objects are abstracted into a hierarchy of classes, allowing the inheritance of properties and methods.

For other basic concepts concerned with object-oriented knowledge representation such as classes, inheritance, attributes, methods, controls, message passing, encapsulation, redefinition, polymorphism, dynamic binding, modularity and reusability, we refer to Leung and Wong (1990) and Meyer (1988).

It should be noted that object-oriented knowledge representation scheme is different from conventional knowledge representation schemes (except frames) in that knowledge is abstracted to classes which are instantiated by objects. It differs from commonly-called object-oriented approach for software construction in that rules are included in methods.

To adequately model a complex system in reality, abstraction mechanisms are necessary. The fundamental abstraction mechanisms from the database paradigm can be used. These abstract mechanisms are classification, generalization and aggregation (Smith and Smith, 1977). Classification is the abstraction from individuals with common properties and behavior to a class, by which 'instance-of' relation is modeled. Generalization is the combination of several classes to a more general superclass, by which 'is-a' relation is modeled. A class that references one or more other classes is called an aggregation of those other classes. By using aggregation, a 'has-a' relation between classes is modeled. Using types in the various relations and message passing, any kind of specific relations can be modeled (Meyer, 1988).

### 3.2 Conventional versus object-oriented knowledge representation

Conventional knowledge representation schemes, such as rules, semantic networks, conceptual graph, object-attribute-value triples and frames, are commonly used in traditional knowledge-based systems (Luger and Stubblefield, 1989; Townsend, 1986). Each of them has its own advantages and disadvantages (Leung and Wong, 1990).

As pointed out by Leung and Wong (1990), a common shortcoming in rules, semantic networks, conceptual graph and OAV representations is that they are not structured enough. Because the knowledge cannot be modularized, the interactions among rules and objects become too complex when the number of objects or rules in the system increases significantly. Thus the system becomes very difficult to manage. When the value of an attribute is modified, it is difficult to pinpoint the effects on the whole system. Therefore, such knowledge representations are difficult to develop and maintain, especially for a large knowledge base like cartographic symbolization.

Frames are more structured than rules, semantic networks, conceptual graph and OAV knowledge representations, since related attributes and rules can be grouped into frames hierarchically. However, modularity of knowledge represented in frames can not be clearly defined, and frame representation lacks flexibility. In a frame system, relationships between frames may be member or subclass links and thus are not unique. Moreover, in some systems, a rule is represented by a frame linked to another frame with special relationship. These factors greatly reduce the structure in a frame system (Leung and Wong, 1990).

Another shortcoming of the conventional knowledge representation schemes is that the objects represented in the schemes are not active. Thus operations through message passing between objects are not possible. Although frames allow the creation of complex objects and the integration of procedural and declarative representations, they are passive data structures that must be acted on by external procedures. The execution of attached procedures requires that the procedure definition be retrieved and evaluated by some external agent (Luger and Stubblefield, 1989).

Object-oriented knowledge representation scheme has the following advantages over the conventional schemes.

Firstly, like semantic networks and conceptual graph, it is flexible. In object-oriented knowledge representation, by storing the names of other objects as the attributes of an instance object, relations between instance objects can be established dynamically (Leung and Wong, 1990). These relationships have the same power as links in semantic networks, and relationships in conceptual graph. In fact, the object-oriented construct can be viewed as dynamic semantic network. The 'is-a' links of semantic network can be implemented in object-oriented representations by relationships between classes and subclasses or between classes and instances. The 'has-a' links can be implemented by the relationships between classes and attributes.

Secondly, object-oriented knowledge representation supports classes and inheritance. In a pure object-oriented system, everything is an object; all objects are abstracted to a certain number of classes. This allows inheritance of attribute names, values, and methods. In addition, each class defines instance variables, which must be instantiated when an individual member of that class is created. Instance objects bind these variables to all the particular information, such as size and location, that distinguishes individuals from each other. The behavior of the members of the class, or the set of all messages to which the class responds, is called the protocol of the class (Luger and Stubblefield, 1989).

Thirdly, it supports modularity and reusability. Modularity and reusability are of prime importance for any truly flexible system. A true modularized system should facilitate modular decomposability, modular composability, modular understandability, modular continuity and modular protection. To achieve these modular capabilities, modules must correspond to syntactic units in the language used, every module should communicate with as few others as possible, exchange of information between modules should be as little as possible, interfaces between modules must be explicit and all information about a module should be private to the module unless it is specifically declared public. Five issues must be solved before we can hope to produce practically reusable modules. These issues are: variation in types, variation in data structure and algorithms, related routines, representation independence and commonality within subgroups (Meyer, 1988). Object-oriented approach satisfies the criteria and principles of modularity, and provides a remarkable set of answers to the set of reusability issues (Meyer, 1988).

Finally, declarative and procedural knowledge can be integrated, and the objects are active. Objects in a object-oriented knowledge representation scheme are active in the sense that the methods are bound to the object itself, rather than existing as separate procedures for the manipulation of a data structure. Objects thus have characteristics of both data and programs in that they retain state variables as well as react procedurally in response to appropriate messages. Objects execute their methods directly in response to a received message. It is the active nature of objects that makes the message passing, execution of methods (rules, routines, etc.) possible. Such methods provide the vehicle for consistency checking, implementing graphics I/O, and combining rules and procedures.

## 3.3 How OOKR scheme facilitates the requirements

Based on the observations in the last two subsection, we then discuss how the OOKR scheme facilitates the specific requirements which are specified in Section 2.

a. Objects and their relationships can be represented in both passive form, and active form by a mixture of attributes, rules, routines, 'is-a' relations, 'has-a' relations and messages. Therefore declarative and procedural knowledge can be integrated in a uniform way, and complex knowledge can be adequately organized.

b. Inheritance exists between classes and subclasses. Thus, knowledge can be represented in an abstracted form with common features generalized in a superclass. Existing classes can be extended and reused by using relevant techniques in object- oriented approaches.

c. As object-oriented approach facilitates modularity, related rules can be well grouped in a class or a module that is independent of other classes or modules. This enhances manageability, understandability and maintainability.

d. Rules and procedure executions can be defined in methods, thus rules and procedures are naturally combined. Routines can be defined by any language which produces routines in an executive form, and then bound to the objects, hence routines such as graphics generation and parameter calculation can be conveniently performed.

e. Data can not only be input through consultation but also be automatically feed from a spatial database through the execution of relevant methods, therefore a knowledge-based system based on this scheme can be naturally attached to a GIS.

Hence it can be concluded that the object-oriented knowledge representation scheme provides a set of answers to the specific requirements, and gives the promises to fully address the issues concerned with cartographic symbolization in a uniform way.

## 4. Examples of Knowledge Structuring for Cartographic Symbolization

In this section, first an example is used to demonstrate how object-oriented knowledge representation scheme can be used to address the issues concerned with cartographic symbolization. Then the abstraction of the knowledge, the capability of the scheme to support reusability and extendibility are discussed.

### 4.1 Knowledge structuring of cartographic symbolization for representing statistical building information from GIS - the example

Let us see how object-oriented knowledge representation scheme can be used to address the issues concerned with the example mentioned in section 2. To solve the problem, classes 'building', 'statistical_map', 'graphics_map' are defined. The definition of each class is illustrated as below. For the convenience of illustration, the definition of the classes is condensed. The notation used is those from Luger and Stubblefield (1989), except that rules are also included in the methods.

```
Class name: building
Superclass: ....
Instance variables: district_identifier, building_identifier,
    building_type, area, ...
Instance methods: ...
    group(): begin
             message(district_identifier, building_type, total_area)
        end
  ...
  end
Class methods: ...


Class name: statistical_map
Superclass: thematic map
Instance variables: map_type, info_property, title, legend, ...
Instance methods:
     info_input(): begin
         for i:= 1 to N do begin
            message(info_property(i))
        end
    end
  end
    map_type():    begin
        rule:  IF info_property = <quantitative> & <absolute> & <multiple>
               THEN map_type = <graphics_map>
               end
        ...
        end
        ...
Class methods:
   begin
    message(map_type, symbol)
      map_generation(title, map_type, symbol, legend)
  end


Class name: graphics_map
Superclass: ...
Instance variables: symbol_type, no_of_variables, variable_color,
    variable_size, ...
Instance methods:
        symbol_type():  begin
                rule:  IF map_type = <graphics_map>
                       THEN symbol_type = <bar_graphs> or <pie_charts>
                       end
                ...
        end
        no_of_variables():  begin
                       message(no_of_variables)
                       end
        end
        variable_color():  begin
                for i:=1 to <no_of_variables> do begin
                message(variable_color(i), color)
                       end
                end
        end
        size(): begin
                for i:=1 to <no_of_variables> do begin
            message(variable_size(i), size_calculation)
                end
            end
    end
    ...
Class methods: ...
```

After these classes have been defined, the cartographic symbolization can be effected by message passing. Through assembling the classes together by message passing, building information can be grouped from relevant database; consultation can be conducted; the map type can be inferred (graphic maps); the symbol can be determined (bar graphs) and the visual variables can be calculated (two elements, color and size), and finally the statistical map can be generated.

It is easy to see from the example that all issues such as procedures, rules, data (described by attributes) can be addressed in a uniform way with the scheme. But, this simple example can not completely show the power of the scheme. The power of the scheme is the abstraction of complex knowledge, the capability to support extendibility and reusability for constructing a system in the large, in our case, the construction of knowledge-based system for cartographic symbolization. We discuss these issues in the following sub-section.

### 4.2 *Abstraction, extendibility and reusability*

Although the above example has shown the flexibility and power of the object-oriented knowledge representation scheme, one can not see from it how the complexity of the knowledge of cartographic symbolization can be modeled. In this section, we will first discuss how the knowledge of cartographic symbolization can be abstracted by classification, generalization and aggregation. We will then illustrate the extendibility and reusability of the abstracted knowledge.

Aspects concerned with cartographic symbolization are generally the interpretation of spatial information concerned, the choice of map type and the design of symbols. These aspects can be sketched as in Figure 1.

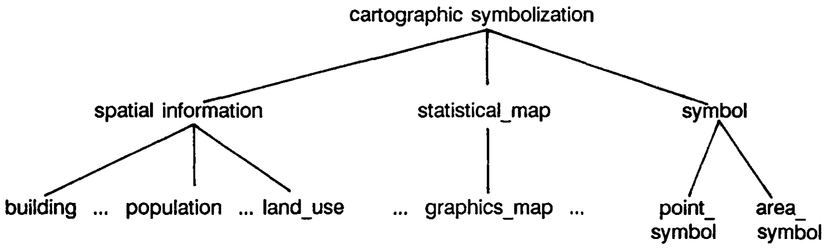

Figure 1 Sketched aspects of cartographic symbolization

To model these aspects in an abstract form, abstraction mechanisms are often used. One can see from Figure 1, information contained in a GIS may be classified as 'building', 'population', 'land use' and so on; these classes can then be generalized to a superclass- 'spatial information'. Likewise, map symbols are often classified as 'point symbol' and 'area symbol', the common property of these classes can be generalized in an abstract form in a superclass - 'symbol'.

Let us take the symbol module and go into depth. All point symbols can be considered as instance objects of class 'point_symbol' which is defined as follows:

```
Class name: point_symbol
Superclass: symbol
Instance variables: form, orientation, color, texture, value, size
Instance methods:
        form():  begin
                message(form, circle)
                ...
                end
        end
        orientation():  begin
                message(orientation, 45)
                ...
                end
        end
        color():  begin
                message(color, green)
                ...
                end
        end
        texture():  begin
                message(texture, 1/5)
                ...
                end
        end
        value():  begin
                message(value, value_calculation)
                ...
                end
        end
        size():  begin
                message(size, size_calculation)
                ...
                end
        end
Class methods: ...
```

Class 'point_symbol' can be regarded as subclass of class 'symbol'. By generalization, the above definition can be revised as follows:

```
Class name: symbol
Superclass: ...
Instance variables: orientation, color, texture, value
Instance methods:
        orientation(): begin
                message(orientation, 45)
                ...
                end
        end
        color(): begin
                message(color, green)
                ...
                end
        end
        texture(): begin
                message(texture, 1/5)
                ...
                end
        end
        value(): begin
                message(value, value_calculation)
                ...
                end
        end
Class methods: ...


Class name: point_symbol
Superclass: symbol
Instance variables: form, size
Instance methods:
        form(): begin
                message(form, circle)
                ...
                end
        end
        size(): begin
                message(size, size_calculation)
                ...
                end
        end
Class methods: ..
```

The above two definitions are abstractive in the sense that any point symbol can be generated through the definitions. They are generalized because common visual variables such as orientation, color, texture and value are defined in the superclass 'symbol'. This ensures that common behaviors across several subclasses (point and area symbols) will indeed have common definition, and instance variables and methods in class 'symbol' can be inherited by its subclass 'point_symbol'.

We then discuss how the definitions can be extended and reused. Suppose that a knowledge base only contain the above two classes about symbol, and now one wants to add class 'area_symbol' into the knowledge base. The question becomes how the definitions can be reused and extended without modifying the existing two classes. In this case, the answer is very simple:

use inheritance and redefinition to define a new class 'area_symbol' as illustrated below. In the class 'area_symbol', 'color', 'texture' and 'value' are redefined. Only orientation is inherited from the superclass 'symbol'.

```
Class name: area_symbol
Superclass: symbol
Instance variables: color, value, texture
Instance methods:
    color(): begin
        for i:=1 to N do begin
        message(color(i), color)
        end
        end
    end
    texture(): begin
        for i:=1 to N do begin
        message(texture(i), texture)
        end
        end
    end
    value(): begin
        for i:=1 to N begin
        message(value(i), value_calculation)
        end
        end
    end
Class methods: ...
```

After the examples, one can immediately see that knowledge concerned with the interpretation of spatial information and determination of map type can be abstracted in a similar way. And then can be reused and extended by using inheritance, redefinition, polymorphism and dynamic binding (Meyer, 1988).

## 5. Discussion and Further Work

Based on the specification of the requirements of the knowledge representation scheme for representing the knowledge of cartographic symbolization concerned with GIS's output, it is argued that the conventional knowledge representation schemes such as rules, semantic networks, conceptual graph, object-attribute-value and frames are not rich and powerful enough to meet the requirements. Then it is suggested to use the object-oriented knowledge representation scheme to represent the spatial knowledge concerned with cartographic symbolization. Discussions show that the object-oriented approach meets the specific requirements and overcomes the major problems of the conventional schemes.

The flexibility and the power of the OOKR scheme are only partially illustrated with examples. The work reported in this paper is still far from fully structuring the comprehensive knowledge of cartographic symbolization. However, as an approach, the object-oriented knowledge representation scheme offers greater potentials for capturing, organizing, processing the knowledge and applying it in the digital domain.

Once a reasonable amount of knowledge is specified and structured with the scheme, the knowledge base and inference engine can be implemented by a suitable media (e.g. a suitable knowledge-based system shell supporting object-oriented knowledge representation). Our opinion is that the object-oriented approach in general is a whole paradigm, in which object-oriented analysis, object-oriented design and object-oriented programming can be distinguished (see e.g. Coad and Yourdon, 1990). As far as only analysis and design (in this case high level knowledge structuring) are concerned, the object-oriented knowledge representation scheme is regarded as a high level construct.

To fully structure the knowledge of cartographic symbolization, a number of issues are still subject to further investigation.

Firstly, further investigation on the object-oriented knowledge representation scheme itself is still necessary, for example, multiple inheritance, and semantics to ensure correctness and robustness. These are the particular interests of the author and will be investigated in the near future.

Secondly, the interpretation of spatial information from a GIS should be addressed in detail as it is the fundamental step for the subsequent symbolization. Several issues are concerned with this aspect, for example, the encapsulation of knowledge in spatial database (this is effected through methods in the OOKR scheme), the rules for the interpretation of the spatial information, and the relationships between the two. These issues are currently under investigation.

Thirdly, much work needs to be done to use the scheme to structure the comprehensive knowledge of cartographic symbolization. To address this, the comprehensive knowledge concerned should be specified first. After sufficient knowledge is specified and captured, the knowledge then can be abstracted into classes by using the object-oriented knowledge representation scheme. The complicated relationships can be represented by 'is-a' relations, 'has-a' relations and messages. New knowledge can be captured gradually, and added to the knowledge base by defining new classes and/or subclasses of existing classes. A full scale knowledge-based system for cartographic symbolization then can be eventually achieved.

## 6. Acknowledgements

I sincerely thank Prof. Dr. Ir. M.J.M. Bogaerts for his stimulating discussions and guidance, and for his arrangement which made this investigation possible. Almost all of our colleagues in the Group of Land Information and Cartography, Faculty of Geodesy, Delft University of Technology, read an earlier version of this paper and gave their comments; I wish to thank all of them. Valuable comments from the anonymous referees are gratefully acknowledged.

## 7. References

**Coad, P. and E. Yourdon**, 1990, Object-Oriented Analysis, Yourdon Press Computing Series.

**Egenhofer, M.J. and A.U. Frank**, 1990, LOBSTER, Combining AI and Database Techniques for GIS. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 6, pp. 919-926.

**Leung, K.S. and M.H. Wong**, 1990, An Expert System Shell Using Structured Knowledge - An Object-Oriented Approach, IEEE Transactions on Computer, Vol. 23, No. 3, pp.38-47.

**Luger, G.F. and W.A. Stubblefield**, 1989, Artificial Intelligence and the Design of Expert Systems, Benjemmin/Cummings Publishing Company, Inc.

**Mackaness, W.A. and P.F. Fisher**, 1986, Towards a Cartographic Expert System, In Proceedings of Auto Carto London, pp.578-587.

**Mark, D.M. and B.P. Buttenfield**, 1988, Design Criteria for Cartographic Expert System, In Proceedings of the 8th International Workshop on Expert Systems, Avignon, France, Vol.2, pp.413-425.

**Meyer, B.**, 1988, Object-Oriented Software Construction, Prentice-Hall.

**Muller, J.-C. and Wang Zeshen**, 1990, A Knowledge Based System for Cartographic Symbol Design, The Cartographic Journal, Vol. 27, No. 2, pp. 24-30.

**Smith, J.M. and D.C.P. Smith**, 1977, Database Abstractions: Aggregation and Generalization, ACM Transactions on Database Systems, Vol. 2, No.2, pp.105-133.

**Townsend, C.**, 1986, Mastering Expert Systems with Turbo Prolog, Howard W. Same & Company.

**Wang, Z.S.**, 1990, A Representation Scheme for Cartographic Information, In Proceedings of the 4th International Symposium on Spatial Data Handling, Zurich, pp. 782-791.

**Weibel, R. and B.P. Buttenfield**, 1988, Map Design for Geographic Information Systems, In Proceedings of GIS/LIS'88, San Antonio, Texas.