ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 133

2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE2016)

Application of Design Patterns in Development of Reusable GIS Symbol Library

Xiaojian Li and Jianxun Chen

College of Computer Science and Technology, Wuhan University of Science and technology, Wuhan, 430065, China

Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan, 430065, China

***Abstract*—The design and development of complex 2D symbol library system based on GIS system is completed by combining design pattern technology. Firstly, it states that the design of the symbol library system has its unique difficulties. The diversity and complexity of symbol library lead to difficulties in the design of symbol. Secondly, using the abstraction to implement the classification and generalization of symbol objects. Using and Improving the classic composite pattern to extend the diversity of symbols with the method of combining primitive symbols. Combining factory method and template method pattern to reduce the coupling between modules and improve the extendibility of the system. Finally, the design of the system is very good to achieve the 2D symbol library's function under GIS. It's integrated in the existing GIS system well. It's powerful, reusable and good extendibility.**

Keywords-design pattern; GIS system; 2D symbol library; primitive symbol

I. INTRODUCTION

Map symbol is the language of geographic information visualization, which expresses the nature and content of geographical elements in the way of visual graphic and image[1].Symbol library system is the editing and management system of map symbols which uses computer to manage, edit, store, retrieve and update the data of map symbol[2]. Reference [3] pointed out that the map symbol design is one of the main functions of the geographic information system and the map drawing system. As a result, good design of symbol library is the important foundation of GIS.

At present, the design pattern has been introduced into the field of GIS software development, but the design of the symbol library system under GIS is rarely involved in the design pattern. In this paper, we explore how to use design patterns to build a reusable and extensible symbol library system. The comparatively popular component GIS symbol library is published in the form of component and embedded into the GIS to achieve reuse. The symbol library designed in this paper is also published in the form of DLL component, but it focuses on the design of complex symbol class. The reusable design can be applied no only to desktop GIS but also to other platform GIS, such as the web GIS system described as in [4].

Symbols in symbol library have the characteristics of diverse styles, complex structures, wide application fields, strong customizability and good interactive. If we do not use the appropriate design pattern in the design and development to eliminate the various dependencies between the modules, small changes can cause significant changes in the overall software architecture. As a result, it will greatly increase the software development cycle and affect the success of software development possibilities. Reference [5] described that design patterns abstract common and repetitive requirements, and define the object interaction pattern that can satisfy these requires, remove the strong coupling in different modules, reduce the complexity of the problem. Design patterns are considered to be widely accepted can be used to improve code reuse, build a stable and robust system technology. Therefore, it is reasonable to believe that it will be feasible and efficient to use the corresponding design pattern techniques to solve specific problems encountered in symbol library design.

Reference [6] said although the problem is similar, the time, the specific conditions are not the same, the solution must be given under the preconditions. Improving the classic patterns appropriately to meet the design requirements of the symbol library.

In this paper, several design patterns will be adjusted and combined to build our system. Firstly, analysis the complex and volatile symbol problem and use object-oriented approach to abstract the problem; Secondly, describe how to use composite pattern produce symbols based on primitive symbols; Finally, combining factory method to produce primitive symbols uniformity and template method to delay the rendering operation of different primitive symbols to sub-class implementation.

II. THE DIVERSITY OF SYMBOLS

A. Problem Description and Analysis

Regardless of ArcGIS, Supermap, MapGIS such GIS software or AutoCAD, CorelDraw such design software provides a rich symbol library. As shown in Figure I, it is intercepted symbols in ArcGIS, some symbols are regular shapes, some are irregular shapes, some are vector symbol, some are character symbols, some are raster image symbols, some are single structure symbols, some are complex structure symbols. Reference [7] also pointed out that the complexity and diversity of symbols and platform compatibility are the huge barriers for symbol library to share in GIS and other systems. Therefore, only a well- designed symbol model can solve the problems described above. The diversity of symbols is mainly reflected in the following aspects:

- The diversity of symbol's attribute parameters. For example, the diversity of symbol's size, color, rotation, style and so on;

- The complexity of symbol's structure. For example, a symbol may be produced by the composition of several primitive symbols in different ways;
- The variability of symbol's behavior. For example, different symbols use different rendering algorithms.

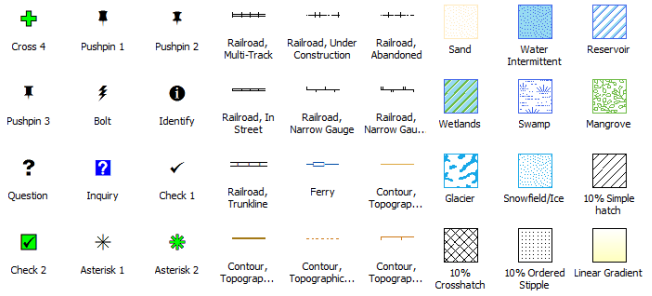

FIGURE I. SYMBOLS IN ARCGIS

B. *Abstract Primitive Symbol Class*

In order to achieve a wide range of different forms of symbols, we abstract graphic representation of the symbols. Reference [8] pointed out that abstraction can help classify and generalize objects. We can understand the characteristics of symbols better by classifying diversified symbols. At the same time, abstracting symbol objects make sure extract the common requires which helps improve reusability.

Symbols representing space features can be divided into point, line and fill symbols according to the distribution of objective things. Point symbols are symbols used to represent small area surface features (such as oil fields, etc.) and point features (such as control points) that can not be expressed in proportion. It has the following characteristics: 1) It keeps fixed pattern while its location changes in the map; 2) It has a fixed location and direction; 3) It's with regular graphics rules, mostly geometric composition. Line symbol is a symbol that expresses things that extend linearly or strip-like, such as a river, whose length and width is not generally represented in proportion. It has the following characteristics: 1) It has an invisible or visible positioning line; 2) Complex line symbol can be seen as the superposition of a number of line with different types (such as straight line, dotted line, dash dot line, etc.); 3) Graphics of line symbol can be considered as the basic unit of line symbol is periodically repeated in the advancing direction of the line. Fill Symbol is a symbol that show the distribution range of things according to the proportion. It has the following characteristics: 1) It has an invisible or visible closed outline; 2) It configures different patterns or colors in the range of outline in order to distinguish objects within the scope of the outline[2].

According to the principles above, diverse symbols are divided into point symbol, line symbol and fill symbol. The common attributes and operations of all kinds of symbols are abstracted into the high-level class.

C. *Design Interface of Primitive Symbols*

As shown in Figure II, public attributes (style name, tag, etc.) and operations (zoom, display, hide, rotate, etc.) of the primitive symbols are abstracted into the high-level interface class ISymbol. Three primitive symbol interfaces IPoint, ILine and IFill are drived from ISymbol according to the unique attributes of the symbol. As a result, the basic primitive class can be derived from them.

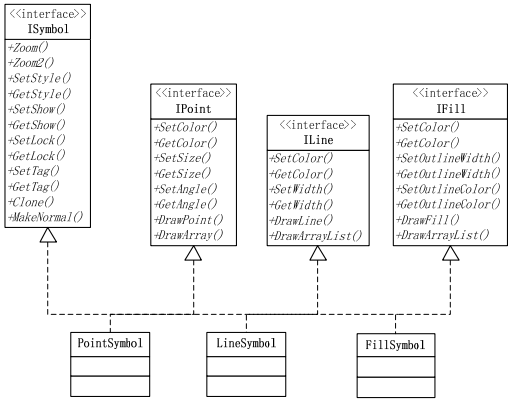

FIGURE II. DIAGRAM OF PRIMITIVE SYMBOL INTERFACE

Three kinds of abstract primitive base classes are abstracted by object-oriented method. Common attribute parameters of symbols are encapsulated into abstract primitive class. And each abstract primitive class can derive specific primitive classes which is extended by its unique attributes and operations. For example, simple point with regular shape, arrow point with direction, image point with raster image, character point using system Unicode and so on can be derived from class PointSymbol.

III. THE PROBLEM OF SYMBOL COMBINATION

A. *The Design of Symbol Layer*

Symbols in library not only have the diversity in attribute, but also are composed of several complex primitive symbols. As shown in Figure III below, a complex line symbol consists of a mark line above and a drawing line below.

FIGURE III. THE COMPOSITION OF A COMPLEX LINE SYMBOL

It can be concluded that a number of primitive symbol get together in a certain order to form a symbol layer. A symbol layer can be expressed as a symbol, and the symbol library provide the management of the collection of symbols. It's summarized as below:

Symbol library = {symbol};

Symbol = the presentation of symbol layer;

Symbol layer = {primitive symbol};

Primitive symbol = {attribute parameter} + {operation}.

The symbol layer(SymbolLayer) has the same attribute with the primitive symbol, so symbol layer objects is also classified into point symbol layer(PointSymbolLayer), line symbol layer(LineSymbolLayer) and fill symbol layer(FillSymbolLayer). The design of symbol layers have shown in Figure IV below:

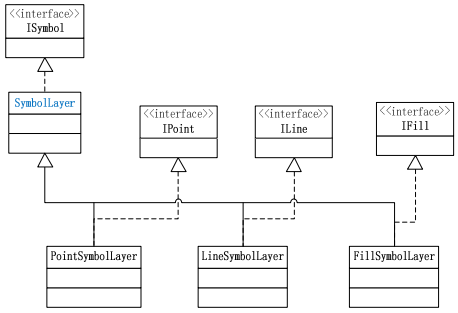

FIGURE IV. DIAGRAM OF SYMBOL LAYER

B. *Complex Symbol Generation Using Composite Pattern*

Symbol layer is a complex object which is assembled by basic primitive symbols in symbol library. So symbol layer should have operations to manage the collection of primitive symbols in it. What is more, both symbol layer and primitive symbol provide same services. For instance, primitive symbol and symbol layer both offer drawing service, primitive symbol draw itself, symbol layer call the set of primitive symbols to draw themselves. Reference [7] said that functional interface provide customer with its service. “Service” is used here because we think it’s appropriate and vivid. In addition, each primitive symbol is not limited to offer the common service, which can provide a service based on its own unique attributes and operations.

Overall, the symbol layer and the underlying primitive symbol are the relationship of “whole” and “part”. They provide the same service, while the primitive symbol can provide services polymorphic. Therefore, the composite pattern can be used to achieve the requirements. It’s defined as: Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly[9].

In addition, the drawing unit of some line and fill primitive symbol depends on other layer symbol. As shown in Figure V, Mark Line primitive depends on another point symbol layer. As shown in Figure VI, Mix Line primitive depends on another line symbol layer. That is to say primitive object is a part of layer object, at the same time, primitive object depends on layer object. It’s a typical dependency relationship, we can let primitive object depends on the layer object based on the classic composite pattern.

FIGURE V. MARK LINE

FIGURE VI. MIX LINE

According to the analysis above, as shown in Figure VII, it’s a diagram of using improved composite pattern with line primitive and symbol layer as an example.

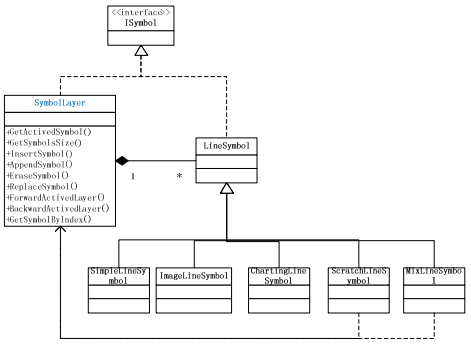

FIGURE VII. COMPOSITE PATTERN USED ON LINE PRIMITIVE AND SYMBOL LAYER

C. *The Result of Design*

By improving traditional composite pattern, we design a symbol editor which is shown in figure below, archiving the following effect:

- As shown in Figure IX part 1, the layer class SymbolLayer as the abstract base class provides management operations on the set of primitive symbols. A symbol layer is composed of several primitive symbols;
- SymbolLayer and LineSymbol inherit from common interface ISymbol, so they can provide the same service. At the same time, each primitive symbol can provide services alone. For example, line circled part in Figure VIII, layer object call the operation change color, each primitive symbol in it call the operation change color. What is more, in Figure IX part 2, each primitive symbol call their own services separately, so it only changes the state of their own.
- Primitive symbols (SimpleLineSymbol, ImageLineSymbol, ChartingLineSymbol, ScratchLineSymbol, MixLineSymbol) inherit from abstract LineSymbol to achieve polymorphic, so they can provide their own unique services;
- Maintaining a SymbolLayer object internally, ScratchLineSymbol and MixLineSymbol can provide services depends on other symbol layers’ function. In Figure X, Mix Line choose another layer object(part 1). It uses the drawing function provided by layer object selected to offer services(part 2).

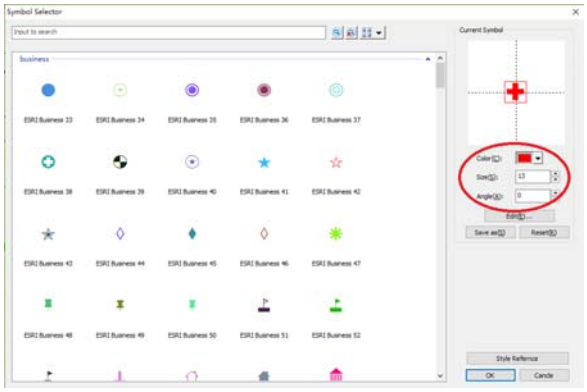

FIGURE VIII. SYMBOL SELECTOR

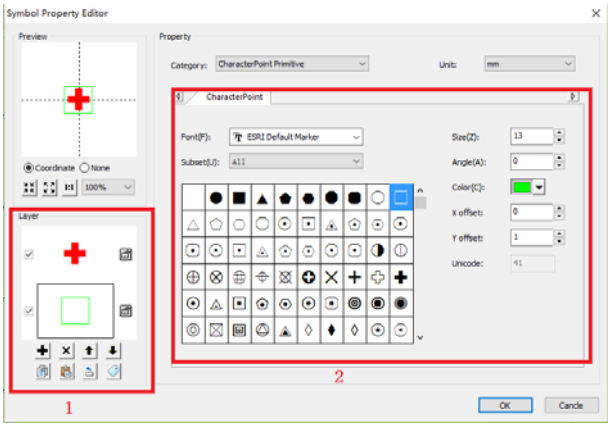

FIGURE IX. SYMBOL EDITOR

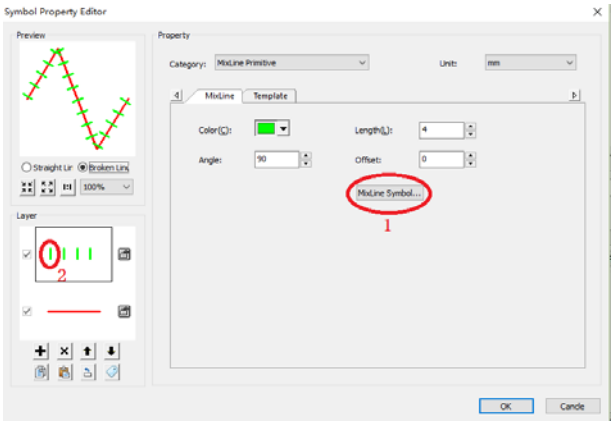

FIGURE X. MIX LINE EDIT

The design of the other two kinds of primitive symbols are similar. It has two improvements compared to the traditional composite pattern. One is the "whole" references to a base class in traditional one, the base class can be polymorphic called to offer correct operation. While the "whole" has the "part" references in improved one, but the "part" is abstract base class which can be used as parent class to derive entity class, so it also can be polymorphic called. Another is the "part" object reference to an abstract "whole" object in order to provide services depends on the "whole".

IV. FACTORY AND TEMPLATE METHOD PATTERN HELP ACHIEVE ENTITY CLASSES

A. *Motivation*

As mentioned before, there are many symbols in the symbol library. Every time we need a primitive symbol, we need new it according to the specific class. As a result, interface classes and symbol layer will depend on symbol classes. If primitive symbols produced by a unified symbol factory, pass a identified parameter to indicate primitive symbol's production, the way to produce primitive symbol only depend on the symbol factory. It becomes unified and greatly reduces the dependencies between modules.

Drawing primitive symbol in symbol library is very complex, some are based on vector graphic drawing, some are based on raster image rendering, some can call system library to draw directly, some need design drawing algorithm by ourselves. What is more, there are many complex public operations in the process of drawing, such as logical and equipment coordinate conversion, unit conversion, translation, zoom, rotation of the coordinate system and so on. All these complex operation are needed while drawing, but the algorithm of rendering for each primitive symbol is greatly different. Therefore, we use the template method pattern to define a skeleton of rendering process. The common operations are defined in the abstract superclass, drawing operation is delayed to subclasses, so we only need realize the unique drawing operation of each entity primitive symbol.

B. *Design and Effect*

1) *The design and effect of factory method pattern*

Factory method: Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory method lets a class defer instantiation to subclasses[9].

As shown in Figure XI, we make some improvement which is different from the traditional factory method pattern: a) Abstract parent class is replaced with generic factory[10] ObjectFactory<IdType, ObjectType> which draws lessons from generic factory in boost library. IdType is the identifier of primitive symbol, and ObjectType is the class of concrete primitive symbol. All we have to do is to register the primitive symbol class in the generic factory. b) We change the entity factory in traditional factory method to singleton factory, in order to have a single factory to produce primitive symbols to avoid too many factory to manage.

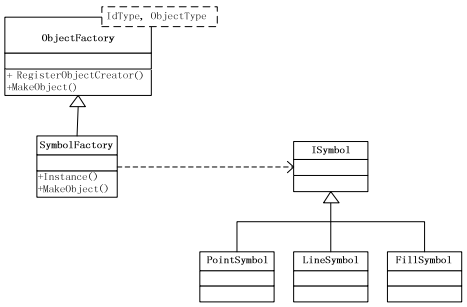

FIGURE XI. PRIMITIVE SYMBOL FACTORY

2) *The design and effect of template method pattern*

Template method: Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template method lets subclasses redefine certain steps of an algorithm whihout changing the algorithm's structure[9].

As shown in Figure XII, we use fill primitive symbol as an example, method DrawFill for single and method DrawArrayList for mass is defined in interface IFill. The virtual method DrawFill call DrawArrayList to realize its function, so method DrawArrayList will be delayed to subclasses' implementation. Abstract class FillSymbol inherits from IFill and realizes all of the common operations needed to draw the fill primitive symbol. The skeleton of drawing template has been established, the only thing we need do is to realize the method DrawArrayList in the subclasses.

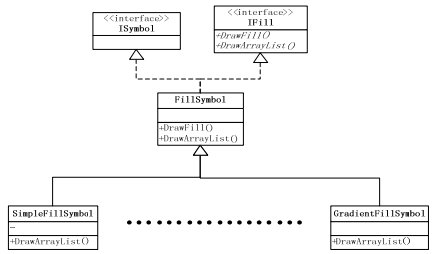

FIGURE XII. TEMPLATE METHOD FOR FILL PRIMITIVE SYMBOL'S DRAWING

3) The Effect of the Combination Use Factory Method and Template Method

The two design patterns are put together to discuss in order to show that the combination of the two patterns can greatly enhance the extendibility and maintainability of the system. Using fill primitive symbol as an example, to achieve a new primitive symbol class only requires two steps: a) Inheriting from the abstract parent class(FillSymbol) and realizing its unique drawing algorithm in virtual method DrawArrayList; b) Registering the new primitive symbol in the generic factory. It achieves the greatest degree of reuse code, hides the realization of the drawing of primitive symbols, so that any drawing algorithm changes will not affect the other parts.

V. CONCLUSION

According to the ideas of this paper, we analyze the diversity of symbols in GIS, and complete the design and development of the symbol library system. Firstly, the symbols are abstracted to implement classification and generalization. Then, Using composite pattern to build the class structure of primitive symbol and symbol layer, realize the expansion of the symbols. Finally, combination use factory method and template method to decouple dependence between modules, ensure system's robustness and extendibility.

It's great correctly and high efficiency to develop symbol library within using and improving classic design pattern. The symbol library can offer all the features of 2D symbol library, and has been integrated into a GIS System successfully with the style of DLL component. As shown in Figure XII, the system has good encapsulation, and is easy to maintain and extend.

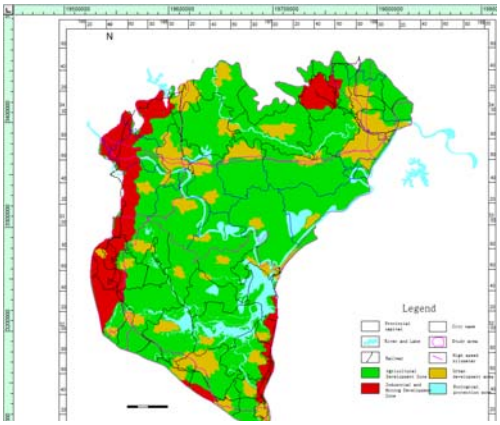

FIGURE XIII. SYMBOL LIBRARY INTEGRATED INTO A GIS SYSTEM

REFERENCES

[1] Desheng Tian. The theory of modern cartography[M].Beijing:Surveying and Mapping Publishing House,1991

[2] Bing Li, Haijian Ye, Jinyun Fang. The Study of Symbol Database Based on Graphical Element Method[J]. Computer Engineering and Applications,2005,41(17):36-38,45

[3] Penggen Cheng, Jianya Gong, Haigang Sui. Design and Implement of Map Symbol Design System in GIS[J]. Journal of Image and Graphics, 2000:5(12):1006-1011

[4] Jinqu Zhang, Yuqiang Zhu. A method based on graphic entity for visualizing complex map symbols on the web[J]. Cartography and Geographic Information Science.2015,42(1):44-53

[5] D Gay,P Levis,D Culler. Software design patterns for TinyOS[J]. ACM. 2005,40(4):40-49

[6] Shuping Lin, Jian Luo. Research on applying design patterns[J].Computer Engineering and Design,2005,16(11):2980

[7] Mingguang Wu, Axing Zhu. An improved map-symbol model to facilitate sharing of heterogeneous qualitative map symbols[J]. Cartography and Geographic Information Science. 2015:1-14

[8] Lawrence W.Barsalou. Abstraction in perceptual symbol systems[J]. THE ROYAL SOCIETY. 2003,358(1435):1177-87

[9] Erich Gamma, Richard Helm, Ralph Johnson. Design Patterns:Elements of Reusable Object-Oriented Software[M]. Beijing:Machinery Industry Press,2000.9

[10] mpforwd. Implement object factory design pattern with boost factory boost function. http:// blog. csdn.net/mpforwd/article/details/5791699.