Prakash and Karri, J Remote Sens GIS 2018, 7:3
DOI: 10.4172/2469-4134.1000248

Journal of Remote Sensing & GIS

Review Article Open Access

Application of Design Patterns for Designing GIS Map Display Component

Anand Prakash* **and Nidhi Arora Karri**

DRDO-Institute for Systems Studies and Analyses, Delhi, India

***Corresponding author:** Anand Prakash, DRDO-Institute for Systems Studies and Analyses, Delhi, India, Tel: 011-23902481; E-mail: adprakash2006@gmail.com

Rec date: August 31, 2018; **Acc date:** September 11, 2018; **Pub date:** September 14, 2018

Abstract

Design Patterns are recurring solutions to common design problems in a well-defined context and conflicting forces. These are documented best practices which can reduce the complexity of design process of a software system by introducing reusability and maintainability in the design. The aim of this paper is to describe the applicability of design patterns representing well- defined design problem in GIS domain. In this paper, we use GIS map display system to define context of various design problems and system of forces associated with different sub-systems.

Keywords: Design pattern; Abstraction; GIS; Open source

Introduction

A Design Pattern [1] is defined as a recurring solution to common problems in a specific context and system of forces. It is a documented best practice to solve a design problem. Design Pattern are not complete codes, hence, we need to specify a context of the problem.

A well-defined context works as a scene for a problem [2]. The system of forces are constraints that must be satisfied by our proposed solution. It can be thought as solution parameters and defines the applicability in which our proposed solution is valid.

Design Patterns reduces design complexity of the system by introducing certain amount of abstraction in it. Abstraction is defined as selective examination of certain aspect of the system under a given purpose. Abstraction is related with uncertainty which in turn is related with amount of information content in the system. Higher abstraction means higher uncertainty and thus higher the potential information in the system. This extra information allows the system to decouple its subsystems and components. Furthermore, abstraction is directly proportional to the reliability and inversely proportional to the complexity of the system. Using appropriate level of abstraction, a design pattern can introduce reusability and maintainability in the design.

GIS is an information system having data and procedures to help user to track events, activities and things by specifying its geospatial location on earth's surface [3]. It is a software system that transforms geospatial data into information. It facilitate users to visualize and analyze geospatial data efficiently [4].

We describe the design of a map display GIS component, commonly known as MapViewer, using Open Source GIS technologies. It can be used to visualize vector and raster data. The MapViewer accepts vector data in Shape File (.shp) and WKT formats. The system accepts raster data in GeoTIFF File (.tif), DTED (.dt0, .dt1, .dt2) formats. It also accepts world files (.jpw, .tfw, etc.). It gives facility to provide Layer Styling in two ways, first is by providing SLD File (.sld) format and the second is Style Chooser Dialog Box on the fly. It also gives basic navigation facilities such as, Zoom-in, Zoom-out, Pan, Reset, NoAction and Info. It can accept data source from PostGIS and Oracle Spatial databases. It provides facility to display maps from a map server using WMS services. It gives facility to save map display area in various image formats such as jpeg, tiff, pdf etc.

Related Work

Usage of design pattern is advantageous to software development but there is insufficient research that supports sound and complete mathematical foundation of design patterns in the area of software development. The term 'pattern' suggests that it's a design solution that repeats itself to solve a design problem in a specified context and in the presence of conflicting system forces.

The problem here is, how to design an application to display vector and raster maps in the context of Geographic Information System? The application should be opened for extension and closed for modification. This constraint creates a system of forces on various subsystems and components implicitly.

Object-oriented modeling and design is a phenomenal problem solving approach in computer science. Rumbaugh et al. [5] describes how to apply object-oriented modeling and design in real world scenarios to visualize both the problem and solution in terms of objects. Object oriented paradigm says that an object is a real-world entity, physical or conceptual, having properties and behavior. Thinking problem and solution in terms of objects allows loosely coupled and highly cohesive components in the system. Tryfona and Hadzilacos [6] describes models and tools for the conceptual level to develop geographic applications in detail.

Stability of a software is defined in terms of system's resistance to ripple effect i.e., cascading phenomena of changes. Ampatzoglou et al. [7] has analyze the effect of GoF design patterns on stability and conclude that usage of certain design patterns provides a shield to the system against any modification introduced in the system.

Effectiveness of software design patterns is a fundamental question in front of software developers to use them judicially. Zhang and Budgen [8] have performed a mapping study that supports that the usage of design patterns can make the development process qualitative. Their mapping study is mainly focused on object-oriented design patterns.

Ampatzoglou et al. [9] have also performed similar kind of mapping study on number of research papers on detection of design pattern and effect of design patterns on software quality attributes.

Dong et al. [10] propose how to use a UML profile for making out design patterns using UML diagrams by declaring new stereotypes, tagged values and constraints to explicitly indicate the roles of design elements in diagrams. Using this technique users can easily visualize design patterns in their applications and can communicate this information dynamically to others.

Hasheminejad and Saeed [11] propose an automatic two phase method to select a design pattern. Two phase method is based on text classification to choose a suitable design pattern. This approach has two advantages. First, no semi-formal specifications is required for design patterns and second, degree of similarity can be used to a design pattern to solve a design problem.

Zhu and Bayley [12] propose an algebra of design patterns by introducing set of operators for instantiating and composing design patterns. Their proposed work include a complete set of algebraic laws. These laws can be used to prove or disprove whether two pattern expressions are equivalent or not.

Kampffmeyer and Zschaler [13] talk about intent ontology for design patterns for finding a suitable design pattern for a specific problem in context. They have developed a tool to describe a design pattern formally that provides a wizard to suggest a user which design patterns are applicable within the problem context. User can submit its query about design pattern within a context.

Moha et al. [14] talk about various design problems. Authors propose a good method to smell bad designs using specification and detection techniques with empirical validation of the proposed method. The bad design can be detected for anti- patterns, functional decomposition, and spaghetti codes.

Cacho et al. [15] explain how AOP (Aspect Oriented Programming) can be blended with design patterns. The authors further investigate the pros and cons of composing design patterns using AOP. Composition of design patterns is, sometimes, challenging due to modularity preservation issue of design patterns; AOP can be a potential solution to preserve the modularity of design patterns while composition process.

Hegedüs et al. [16] discuss about Impact of design patterns on software maintainability is a matter of analysis because there is relatively little research that can confirm the fact that usage of design patterns is, indeed, helpful in software maintainability.

Eden et al. [17] suggest a mathematical basis for instantiating design patterns by introducing uniform sets of classes or functions, class hierarchies and regularities. The authors further introduce LePUS, a declarative, higher order language. LePUS can describe a design pattern in a general complete and accurate manner using predicate calculus.

Saluka and Bertok [18] propose mathematical approach to categories design patterns by providing more general descriptions of design elements. The authors point out that the proposed method captures static and dynamic properties of patterns and is capable for analyzing, comparing and detecting design patterns from existing software programs.

Conceptual Modeling of GIS Applications [19] is an important research area to cater for different types of uncertainties in both conceptual and logical levels. The authors propose a method to include imprecision and uncertainty in database models. Uncertainty is inversely proportional to the complexity of a system. It means that if we want to reduce the complexity of a system then we need to raise the level of uncertainty and abstraction. Thus, this can be thought as a basic concept to develop an Intelligent GIS.

Description of Mapviewer

The Objective of developing MapViewer is to design a software system to display vector and raster maps. The system uses Geotools [20] which is a Java GIS toolkit to develop GIS applications. The system uses Geospatial Data Abstraction Library [21] for reading and writing a raster spatial data formats like GeoTIFF.

The MapViewer provides support for two spatial databases namely PostGIS and Oracle-Spatial. It provides a simple interface to select spatial tables from a database to create feature layers on the map.

The MapViewer provides functionalities to display spatial data from various data sources and provides standard GIS navigation tools. Styled Layered Descriptor (SLD) is an open source XML based document that addresses the requirement of the visual portrayal of geospatial data on maps. The system provides an interface, known as Style Chooser, to allow the user to give style to different layers on the map. In addition, the system provides basic map navigation toolset like pan, locate, reset, zoom-in, zoom-out etc.

The MapViewer has five subsystems namely, Map Display Manager, Data Source Manager, Feature Layer Manager, User Interface Manager and finally Database Connection Manager. These five subsystems represent five different common problems in any GIS application to visualize a map in raster and vector formats.

The MapViewer application framework has been designed using various relevant design patterns. The architectural pattern followed is MVC (Model-View-Controller) and the software design patterns currently being used are Template Method, Abstract Factory, Flyweight and Prototype.

Model-View-Controller architectural pattern is described in Figure 1. MVC is a good choice while designing an interactive application like GIS. Model implements business logic, View implements GUI (graphical user interface) and Controller implements the way the user interface reacts to user input.

MVC introduces reusability and flexibility in the design of a software component by decoupling business logic, GUI and user interactions.

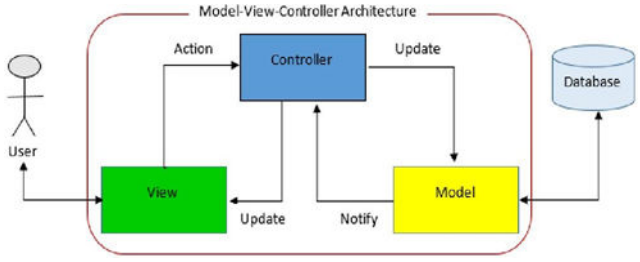

Figure 1: Model-View-Controller Architecture.

The MapViewer uses GeoTools as the basic GIS library to implement GIS functionalities. GeoTools is java based open source library that provides comprehensive functionalities to develop a GIS application. GDAL raster library is used to read raster images then GeoTools is used to display these raster images. To gain better look and feel MapViewer uses JTattoo UI library. Figure 2 shows the block diagram of the technologies used to develop the software.

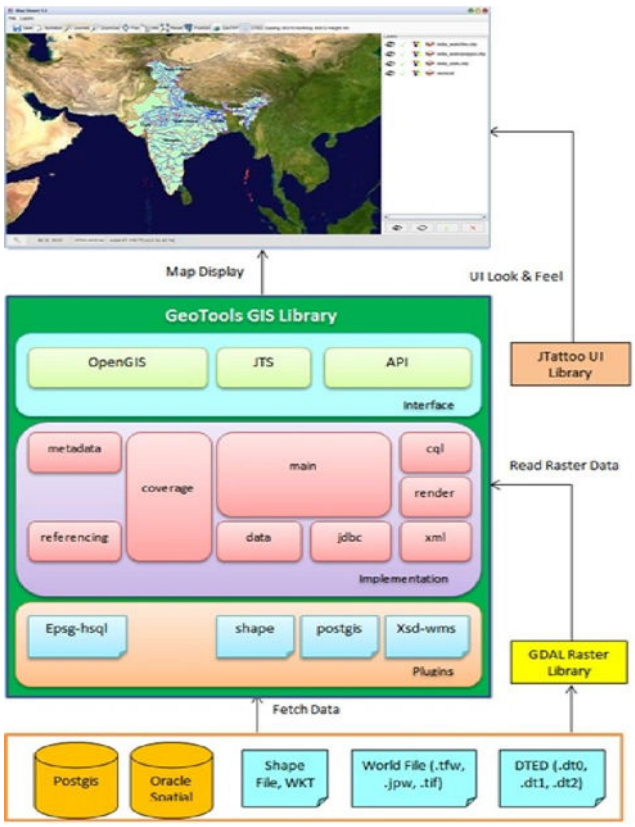

Figure 2: MapViewer block diagram.

The MapViewer uses modular design whereby each module (sub-system) is shown using Package diagram in Figure 3. The diagram shows how different sub-systems interact with each other according to MVC architectural pattern. Packages in yellow color belong to Model, package in green color belongs to View and Package in Blue color belongs to Controller parts.

Map Display Manager, is responsible to display vector and raster maps in different formats. This subsystem is mainly responsible to implement view classes for displaying a map in GIS application. The subsystem is also responsible to provide user interfaces to interact with maps. These interfaces mainly include data source selection wizards, navigational tools and map area saving in different formats. The package keeps the outline of map display algorithm intact.

Data Source Manger is responsible to implement model classes for displaying maps using different data sources like raster images, PostGIS and Oracle Spatial databases. This package contains main business logic to create and display map data sources without exposing its implantation details to the client.

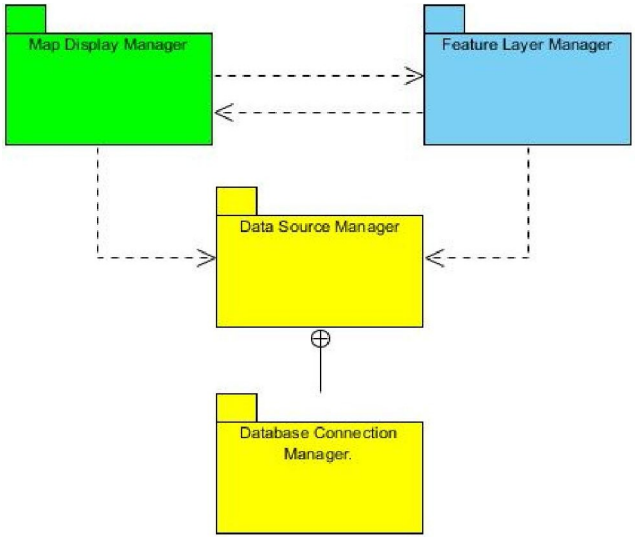

Figure 3: MapViewer Package diagram.

Data connection Manager is responsible to implement model classes for creating and managing connections for different users. The package is contained in the Data Source Manger package. The package has responsibility to effectively manage connections.

Feature Layer Manager is responsible to create feature layers. The package implements controller classes.

Design Patterns in Mapviewer

Map display management using template method pattern

Problem is, how to keep the outline of map display algorithm intact in terms of abstract operations among subclasses belonging to the subsystem?

Context of the problem is Map Display Management subsystem which is responsible to keep the map display algorithm intact and common. Map display algorithm has variant and invariant portions. The variant portion is creation of style, creation of layer and creation of map content. The invariant portion is outline of map display algorithm which is common and intact for subclasses.

System of Forces for the given subsystem is to display a map using code reuse and implementing the invariant part of the map display algorithm by specific subclasses.

Solution to this problem is Template Method pattern. It is used in those situations where there is an algorithm having two portions. One portion of the algorithm is called invariant and the other portion is called the variant part. Invariant portion of the algorithm is supposed to be intact during the execution of the algorithm such as outline of the algorithm. This invariant portion is implementing inside a method known as template method defined inside a class known as template class. The variant portion of the algorithm is left for specific implementation by different subclasses of this class. These subclasses implement this variant portion in many different ways. There are subclasses, described in Figure 4, responsible to give specific implementations to the variant portion of the map display algorithm.

Here, Map is an interface having a template method known as displayMap() method, that keeps the outline of the map display algorithm intact. Subclasses namely MapGeoTIFF, MapDTED, MapShapeFile, MapPostGIS and MapOracleSpatial implement those variant portion which is specific to map display using different data sources.

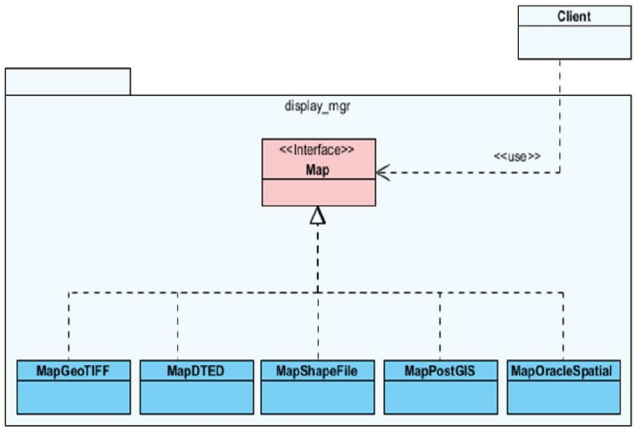

Figure 4: Template Method Pattern.

Data source management

Problem is, how to manage different data sources for displaying feature layers on a map?

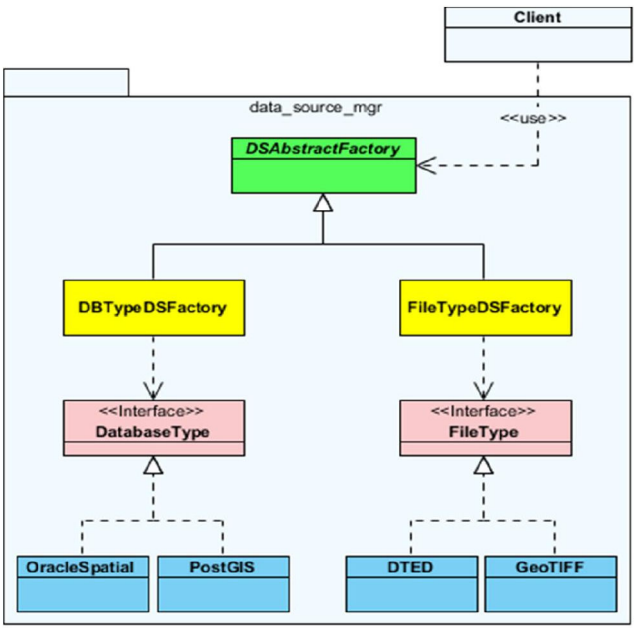

Figure 5: Abstract Factory Pattern.

Context of the problem is Data Source Management subsystem which is responsible to manage two types of the data sources namely file type and database type. File Type data source has two types, DTED and GeoTIFF. Database Type data source has two types, PostGIS and Oracle Spatial.

System of forces for the given subsystem is to create a data source object without knowing its concrete class, when there are family of data sources available in the subsystem.

Solution to this problem is Abstract Factory pattern. It is a creational pattern which can be used by client to define a common interface to create family of related objects. The instantiation process is transparent to the client. It allows client to decide when to create without knowing how to create these objects. Abstract factory can be considered as a factory to manage other related factories. Client uses a concrete factory producer class to produce a desired factory by passing factory type information to abstract factory method which is defined inside it. The proposed solution to the above mentioned problem is explained using a UML model described in Figure 5.

Feature layer management using flyweight pattern

Problem is, how to reduce the number of objects in need while ensuring the performance both in terms of memory and time.

Context of this problem is to display feature layers created from different data sources on a map. The layer management should be effective in terms of performance and memory utilization. The subsystem, Feature Layer Manager, is mainly responsible to decrease the feature layer objects' memory requirement for displaying on a map.

System of forces for the given subsystem are available system memory and number of map feature layers respectively. MapViewer allows thousands of feature layers to load in the memory.

Solution to this problem is Flyweight pattern that comes under structural pattern. Flyweight pattern can, significantly, reduce the number of objects to be created. The pattern is used when an object has intrinsic and extrinsic information in it.

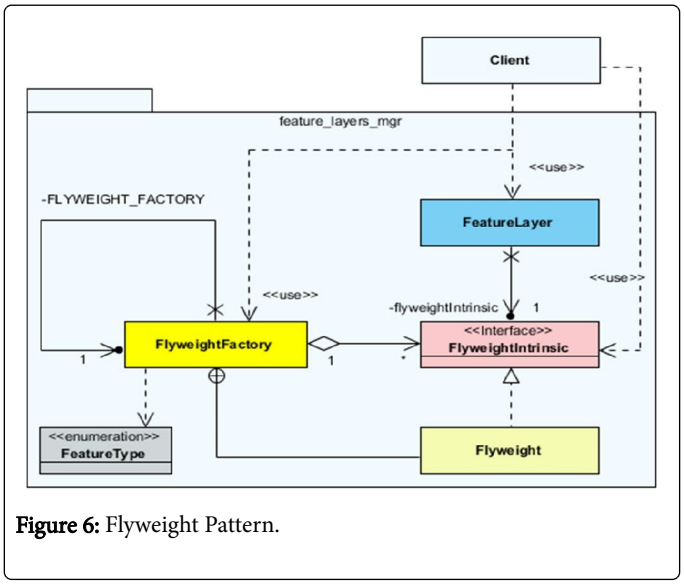

Figure 6: Flyweight Pattern.

The intrinsic information is inherently independent of a problem context and can be shared among number of instances of the same class and can be encapsulated in a single common Flyweight object. The extrinsic information is context dependent and is unique for every instance of the given class. The proposed solution to the above

mentioned problem is explained using a UML model described in Figure 6.

The Flyweight class is an inner class defined inside a Singleton Flyweight Factory class that uses a collection like HashMap to store created objects. If a request comes for object creation then, its looked into the HashMap first, if it is already stored in HashMap then, a reference to the object is returned back, otherwise a new object is created, stored in the HashMap and a reference to the object is returned back.

Database connection management using prototype pattern

Problem is, how to reuse already existing objects to create new database connection objects while ensuing the performance in terms of time?

Context of this problem is Database Connection Manager, which is a subsystem of the MapViewer. The requirement of the system is to create various database connections with multiple states and privileges to the same database server keeping the performance of the creation process in mind. Different users are required to be connected server based on their profile.

System of forces here is number of available database connections. The subsystem supports the PostGIS and Oracle Spatial databases with ten connections each.

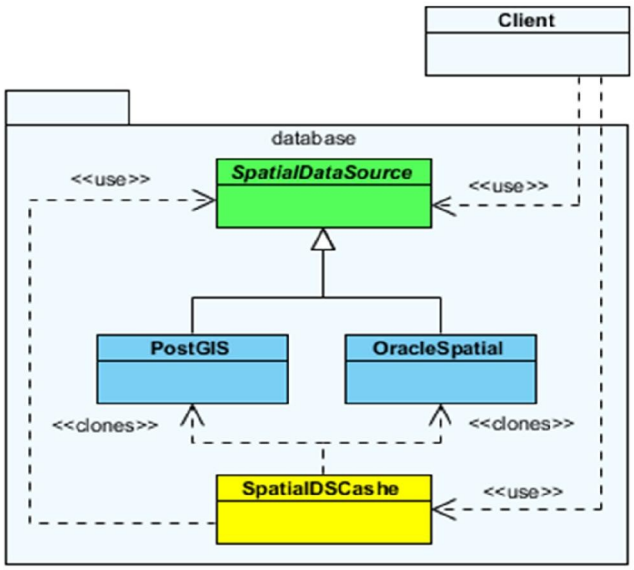

Figure 7: Prototype Pattern.

Solution to this problem is Prototype pattern which is a type of creational pattern. This pattern is applicable in those situations whereby we need to create a number of objects differing only in terms of their state. Creation of individual connection object is both time and resource consuming. The prototype pattern allows designating one object as a prototype object and other objects are created by copying the prototype object. These objects are then modified as per the requirement of the application. The proposed solution to the above mentioned problem is explained using a UML model described in Figure 7.

The prototype object is defined by the abstract class SpatialDSCache which is extended by the concrete classes PostGIS and Oracle Spatial. The abstract class defines the clone function which is implemented by the concrete classes. The client requests for the type of DataSource from SpatialDSCache.

The respective concrete class creates the clone of the connection and returns it to the client. The stateful object so returned can be further modified as per requirement.

Conclusion and Future Work

A typical GIS application is the amalgamation of various subsystems and hence the use of effective software design methodology is imperative here. This paper aptly describes the problem and the solution to the problem in the form of the identified design patterns. For example, Template Method pattern is used for map display wherein the map display part is kept invariant and feature style management is kept variant. Different data sources, namely File Data Type and Database Type, are maintained in the form of individual Abstract Factory and implemented by Abstract Factory design pattern. The paper also gives solution to one of the central problem of the GIS system which is the display of the large number of vector features. The problem is resolved using compound design pattern wherein design patterns namely Flyweight and Singleton work together. The combined application of these patterns markedly improve the performance of the feature display in terms of time and memory. A typical map consists of thousands of feature types features. Flyweight pattern manages the features effectively by maintaining the objects during run time restricted to number of feature type thereby improving performance. Database connection management is handled using Prototype pattern. Thus, this paper mainly focuses on solution to design problems in GIS domain.

The future work in the extension of the current work includes the validation of the effectiveness of design patterns. Currently there is no mathematical model to validate and verify the advantages of design patterns. The authors would make an attempt in this field. Further, the authors would also like to further study the relation between language and design patterns and identify the language dependent and language independent design patterns. The authors further plan to carry out work on development of GIS application on mobile devices and study the impact of various constraints of mobile environment on various design patterns to be used for the development.

References

1. Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: elements of reusable object- oriented software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
2. Kuchana P (2004) Software Architecture Design Patterns in Java. Auerbach Publications, Boston, MA, USA.
3. Shekhar S, Xiong H (2008) Encyclopedia of GIS. Springer Science Business Media, Berlin, Germany.
4. Demers NM (2008) Fundamentals of geographical information systems. John Wiley & Sons, New Jersey, USA.
5. Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W, et al. (1991) Object-oriented modeling and design. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
6. Tryfona N, Hadzilacos T (1995) Geographic Applications Development: Models and Tools for the Conceptual Level. Proceedings of ACM-GIS'95. Baltimore, Maryland, USA, pp: 19-28.

7. Ampatzoglou A, Chatzigeorgiou A, Charalampidou S, Avgeriou P (2015) The Effect of GoF Design Patterns on Stability: A Case Study. IEEE Transactions on Software Engineering 41: 781-802.
8. Zhang C, Budgen D (2012) What do we know about the effectiveness of software design patterns? IEEE Transactions on Software Engineering 38: 1213-1231.
9. Ampatzoglou A, Charalampidou S, Stamelos I (2013) Research state of the art on GoF design patterns: A mapping study. Journal of Systems and Software 86: 1945-1964.
10. Dong J, Yang S, Zhang K (2007) Visualizing design patterns in their applications and compositions. IEEE Transactions on Software Engineering 33: 433-453.
11. Hasheminejad SMH, Jalili S (2012) Design patterns selection: An automatic two-phase method. Journal of Systems and Software 85: 408-424.
12. Zhu H, Bayley I (2013) An algebra of design patterns. ACM Transactions on Software Engineering and Methodology (TOSEM) 22: 23.
13. Kampffmeyer H, Zschaler S (2007) Finding the pattern you need: The design pattern intent ontology. In International Conference on Model Driven Engineering Languages and Systems. Springer, Berlin, Heidelberg, Germany 4735: 211-225.
14. Moha N, Gueheneuc YG, Duchien AF (2010) Decor: A method for the specification and detection of code and design smells. IEEE Transactions on Software Engineering (TSE) 36: 20-36.
15. Cacho N, Sant'anna C, Figueiredo E, Dantas F, Garcia A, et al. (2014) Blending design patterns with aspects: A quantitative study. Journal of Systems and Software 98: 117-139.
16. Hegedűs P, Bán D, Ferenc R, Gyimóthy T (2012) Myth or reality? analyzing the effect of design patterns on software maintainability. In Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity. Springer, Berlin, Heidelberg, Germany, pp: 138-145.
17. Eden AH, Gil JY, Hirshfeld Y, Yehudai A (1999) Towards a mathematical foundation for design patterns. Technical Report, Department of Information Technology, Uppsala University.
18. Kodituwakku SR, Bertok P (2003) Pattern categories: a mathematical approach for organizing design patterns. In Proceedings of the 2002 conference on Pattern languages of programs. Australian Computer Society, Inc. 13: 63-73.
19. Yazici A, Akkaya K (2000) Conceptual Modeling of Geographic Information System Applications. In: Recent Issues on Fuzzy Databases. Studies in Fuzziness and Soft Computing Physica. Springer, Heidelberg, Germany.
20. Turton I (2008) Geo Tools. In: Open Source Approaches in Spatial Data Handling. Advances in Geographic Information Science, Springer, Berlin, Heidelberg, Germany.
21. Warmerdam F (2008) The Geospatial Data Abstraction Library. In: Open Source Approaches in Spatial Data Handling. Advances in Geographic Information Science. Springer, Berlin, Heidelberg, Germany.