The Application of TRIZ Theory in Software Engineering

Yanqing Li, Jinghao Xu, Changdao Zheng,Yueyi Zhao, Xueyu Cheng, Zhen Zhao*

Qingdao University of Science and Technology, Qingdao 266061, China

*Corresponding Author: zzxm2000@126.com

Abstract

This paper analyzes the application of TRIZ theory in the field of software engineering from many aspects. This paper introduces the contents of the TRIZ theory system and the interaction with the software engineering field, and uses the Internet innovation of the TRIZ invention principle, and the TRIZ invention principle also has a reference role in the software design pattern.

Keywords

TRIZ; software engineering; principle of invention; mobile Internet; object-oriented.

1. Overview of TRIZ

TRIZ is the Russian acronym for the invention of problem solving theory. It was proposed by professor G.S.Altshlller of the Soviet Union and a group of researchers led by him on the basis of studying 2.5 million patents of various countries in the world since 1946. Altshlller believes that the basic principles of invention problems exist objectively. These principles can be confirmed and sorted out to form a theory, which can not only improve the success rate of invention, shorten the cycle of invention, but also make the invention problems predictable. After 60 years of development, TRIZ has become a powerful methodology for solving invention problems, which has been applied to enterprises in the Soviet Union, the United States, Japan and many European countries, and solved thousands of problems in the development of new products.

2. TRIZ tools, techniques and methods

TRIZ has established a series of tools and principles, which can be broadly divided into three categories: TRIZ's theoretical basis, analytical tools, and knowledge base. The evolution of technology system is the theoretical basis of TRIZ. After studying a large number of patents, innovation in the most popular sense is the process of finding and solving problems creatively. The powerful role of TRIZ theory is that it provides systematic theoretical and methodological tools for people to creatively discover and solve problems.

The modern TRIZ theoretical system mainly includes the following contents:

1. Innovative thinking methods and problem analysis methods

TRIZ theory provides scientific methods for systematic analysis of problems, such as multi-screen method. However, the analysis of complex problems includes the scientific modeling method of problem analysis -- matter-field analysis method, which can help quickly identify the core problems and find the fundamental contradictions.

2. The law of technological system evolution

Based on the analysis of a large number of patents, the TRIZ theory summarizes and extracts eight basic evolutionary principles. By using these evolution rules, we can analyze and confirm the technical status of current products, predict the future development trend, and develop competitive new products.

3. Principle of solving technical conflicts

Different inventions often follow the same rules. TRIZ theory summarizes these common laws into 40 innovation principles, and specific solutions can be sought based on these innovation principles and in combination with engineering practice for specific technical contradictions.

4. Innovate the standard solution to the problem

According to the different characteristics of the matter-field model for specific problems, there are corresponding standard model processing methods, including model dressing, transformation, material and field addition and so on.

5. Invent problem solving algorithm ARIZ

It mainly aims at the technical system with complex problem situation, contradiction and unclear related parts. It is a non-computational logic process that carries out a series of deformations and redefinitions of the initial problem, realizing the gradual in-depth analysis of the problem, the problem transformation, until the problem is solved.

6. Knowledge base based on engineering principles such as physics, chemistry and geometry

The knowledge base based on the analysis results of millions of invention patents in physics, chemistry, geometry and other fields can provide a rich source of solutions for technological innovation.

The 40 principles of invention in TRIZ theory are not detailed here. Many of them are closely related to engineering, but mainly focus on mechanical design. TRIZ also proposes 39 general parameters, which do not necessarily represent all the parameters, but are relatively general parameters in terms of design. We can also customize it.

3. TRIZ′s interaction with the field of software engineering

Software engineering is the study of how to develop and maintain software in a systematic, standardized, quantifiable, and procedural way, and how to combine time-proven sound management techniques with the best available technical methods. It involves programming language, database, software development tools, system platform, standards, design patterns and other aspects.

As a software engineering in the development of industry, are in close contact with the other industry, more and more products need software support, but without a good software design for the design of the model, and the design patterns, and some design thought is essentially in the use of TRIZ theory and practice, through the study of TRIZ course this semester and team concluded, we combined with their own professional knowledge, deep understanding about the application of TRIZ theory in software engineering, discussed the software in the field of design patterns, method of idea is from the theory of TRIZ, And how TRIZ theory is used to solve problems in the field of software. The following will illustrate the practice of TRIZ theory in the field of software engineering through several examples of problems in the field of software engineering.

4. Application example

1．Innovations related to Internet applications

Here we mainly introduce the innovation of Internet application in the last one or two years, including some interactive methods, design concepts... Etc.

WeChat Applet：

Applet is an application that can be used without downloading. It is also an innovation with a very high threshold. After nearly two years of development, a new Applet development environment and developer ecology have been constructed. Small program is also China's IT industry for many years in a truly able to affect the innovation of ordinary programmer, now has more than 1.5 million developers to join in the development of small programs, together with we can promote the development of small programs, small program applied to a population of over one million, covering more than 200 segment of the industry, the day live users achieve two million, small program implemented in several cities support the subway and bus service. Given all the benefits of small programs, how do small programs borrow from TRIZ theory? Let's start with the TRIZ theoretical method used by the Applet:

Extraction principle

Abstract the advantages of ordinary web application technology application, small program is based on the web specification, using HTML,CSS and JS to build a set of framework, WeChat official gave them a very awesome name: WXML,WXSS, but in essence is still under the construction of the whole web system.

Combination principle

The combination of web application and traditional mobile client greatly reduces the learning cost of users because the operation logic and ordinary APP are basically the same

Multi-function principle

Developers can implement different functions under the framework of WeChat. A lot can be done with the same technology. The nesting principleApplications nested within the application, unified entry, on demand, no separate download.

Pre-action

To provide a smooth user experience and reduce load time, WeChat pre-loads the basic framework into the user's phone. At the same time, the user will cache part of the data after the first use of a small program, reducing the time of the next load.

Dynamic principle

Because of web-based development, a large number of parameters of the application are loaded in real time, and the data in the cache will be loaded dynamically and used for the program every time the application starts. It saves users the trouble of constantly updating the APP.7. The principle of deficiency and excess. Ordinary developers can hardly expect the effect and popularity of the APP. The dynamic update principle of WeChat small program can be very convenient to quickly improve and remedy the deficiencies and bugs of the program.

Multi-dimensional principle

Internally nested patterns transform one-dimensional applications into two-dimensional ones, reducing application entry and adding rich functions.

Principle of mediation

WeChat is equivalent to an intermediary. Small programs do not need to activate registration, login and other complicated steps. Relying on a large number of user groups of WeChat, developers can quickly launch an application and obtain a large number of user groups. The mode of mediation can also prevent users from installing malicious applications and improve security.

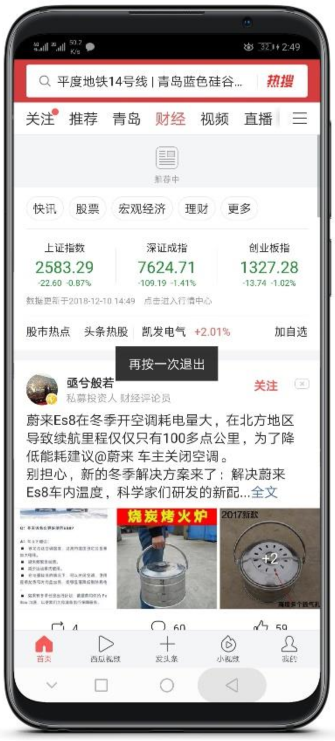
2. How apps interact

Principle 2 of invention: Taking out-separation

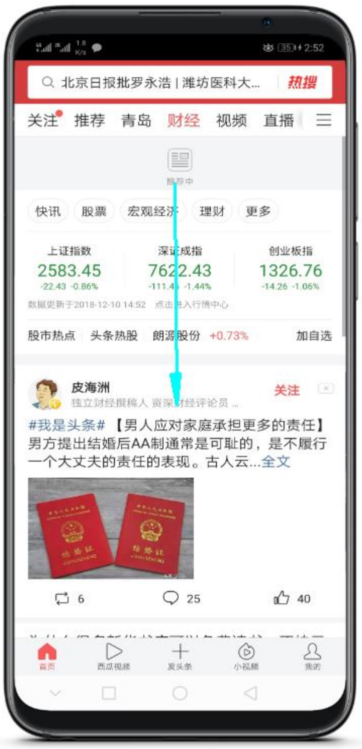
Pull-down refresh: drag the screen with your finger and pull down, the App will slide out a specific content and refresh the current page. In 2012, Twitter invented the new way of interaction, and before that such applications to refresh the way most used specific refresh menu buttons, drop-down refresh (Pull to refresh) cause product interaction of a small wave after the development of mobile client applications to emulate the realization of the function of the separation, make the interaction more convenient and humanization, see Fig. 1 and Fig.2.

Principle 6 of invention: Universality

Back to refresh: another innovation to refresh mode, to toutiao (v7.0.1 Build 40bf5ff; 20181204) as an example, the refresh mode integrated into the user's return to action, when users click the back button for the first time the page refresh, at the same time giving prompt: "then quit once" perhaps it is designed to keep think about leave application users, but it also brought an innovation to application refreshes the way, the return key to perform a variety of functions, and the location of the return key in the mobile side lower part, more convenient for the operation of the fingers, see Fig.3 and Fig.4.

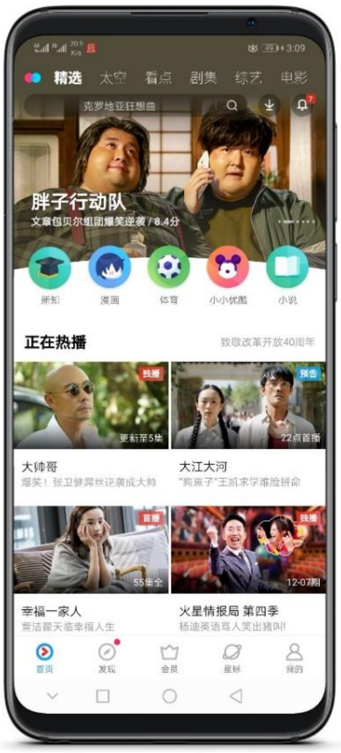

平度地铁14号线 | 青岛蓝色硅谷...
热搜
关注 推荐 青岛 财经 视频 直播
推荐中
快讯 股票 宏观经济 理财 更多
上证指数 2583.29 -22.60 -0.87%
深证成指 7624.71 -109.19 -1.41%
创业板指 1327.28 -13.74 -1.02%
数据更新于2018-12-10 14:49 点击进入行情中心
股市热点 头条热股 凯发电气 +2.01% 加自选
再按一次退出
亟兮般若
私募投资人 财经评论员
关注
烧炭烤火炉
2017新款

蔚来Es8在冬季开空调耗电量大，在北方地区导致续航里程仅仅只有100多点公里，为了降低能耗建议@蔚来 车主关闭空调。
别担心，新的冬季解决方案来了：解决蔚来Es8车内温度，科学家们研发的新配...全文


北京日报批罗永浩 | 潍坊医科大...
热搜
关注 推荐 青岛 财经 视频 直播
推荐中
快讯 股票 宏观经济 理财 更多
上证指数 2583.45 -22.43 -0.86%
深证成指 7622.43 -111.45 -1.44%
创业板指 1326.76 -14.26 -1.06%
数据更新于2018-12-10 14:52 点击进入行情中心
股市热点 头条热股 朗源股份 +0.73% 加自选
皮海洲
独立财经撰稿人 资深财经评论员 ...
关注
6 25 40

#我是头条# 【男人应对家庭承担更多的责任】男方提出结婚后AA制通常是可耻的，是不履行一个大丈夫的责任的表现。古人云...全文

Fig. 1 Return key refresh Fig. 2 The drop-down refresh



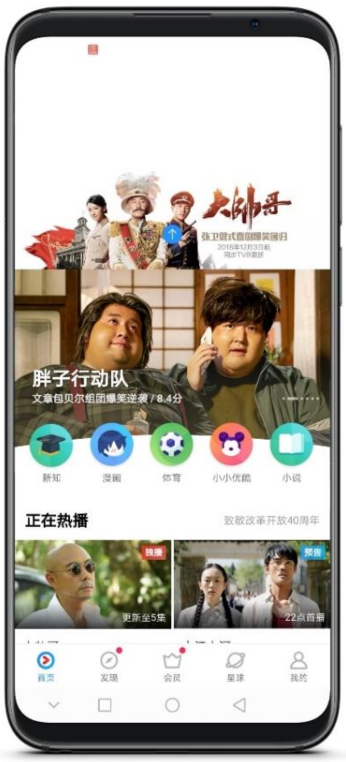


Fig. 3 Pull-down operation Fig. 4 Pull-down operation

Other application drop-down pages are no longer limited to "updates",but are given a new feature: thematic events. After entering the App, JDcom's drop-down page presents games for 618 promotion activities, while Youku's drop-down page is the promotion of some TV series, which have been innovated on the basis of drop-down updates. In fact, this method of play Taobao, Tmall has been successfully practiced before, in this way before the promotion of small games implanted, issued vouchers, red envelopes, etc. This can make the software more interesting, enhance the user's desire to buy, the drop-down page has a more rich application scenarios, to achieve a variety of functions.

3. Design patterns

The first thing that comes to mind is the strategy pattern in the software design pattern. I will introduce the strategy pattern. Of course, other design patterns are also implementing and utilizing TRIZ principle theory.

This situation is often encountered in software development. There are many algorithms to realize a certain function. We can choose different algorithms to complete the function according to different application scenarios. Extract the parts of A class (A) that change frequently or may change in the future, as an interface (B), and include the interface (B) in class (A) so that instances of class (A) can invoke the behavior of the class (C) that implements the interface at runtime. For example, define a series of algorithms, encapsulate each algorithm, and make them interchangeable, so that the algorithm can be independent of the customer using it. This is the strategy pattern. In the book Head First design patterns, a vivid example is used to explain:

The company is going to make a duck game. There will be a variety of ducks in the game, while swimming in the water, while quack, and some can fly. How to design this project?

Scenario 1 Inheritance

A superclass Duck is designed, including methods quack (), swim () and fly () respectively to simulate the quacking, swimming, flying and other behaviors of ducks, and an abstract class display () to display the different appearances of each Duck. Each duck subclass implements display () when it inherits its parent class. The advantage of this is that each duck subclass inherits a parent class's methods at the same time, which can achieve the purpose of code reuse, but this will make some subclasses not suitable for this behavior also have this behavior. If some subclass of duck, such as a "rubber duck", does not have certain functions (such as flying), it should not have this flying function. Of course, you can Override this method with @override in subclasses. But when different subclasses need to override different methods, it can be cumbersome and error-prone, and these new overridden methods cannot be reused. For example, "rubber duck" can only quack but can't fly, "wood duck" can't quack and can't fly. Every time a new duck subclass comes along, you're going to have to check and possibly override those methods, which is maddening.

Scenario 2 Interface

In the superclass Duck, variable methods such as quack (), fly () are replaced by interfaces Quackable () and Flyable (). In each Duck subclass, this interface is implemented if there is a "fly" or "quack" function. The downside is that the code can't be reused. If you have 100 subclasses that all behave like flying, you have to repeat the code 100 times.

Design principle 1: identify areas of the program that may or may not need to change, and isolate them. Changing one part of the system does not affect the other parts.

This is also the principle of TRIZ theory -- the principle of separation, the selection or separation of key parts of an object. In this case, it refers to separating the changing parts and abstracting them into interfaces for implementation. Since fly () and quack () change from Duck to Duck, separate the two behaviors from the Duck class and create a new set of classes to represent each behavior.

Design principle 2: program to an interface, not an implementation.

Combining with the dynamic characteristics of TRIZ theory, the structure of the object is divided into several combined parts which can be changed and matched with each other. Automatically adjusts an object to make it movable or adaptive. In this example, we use polymorphism to program the superclass, and execute the actual behavior according to the actual situation, so as not to be tied to the behavior of the superclass. This makes the properties of the object adaptive. It used to be that the behavior came from a concrete implementation of the superclass or inherited an interface and was implemented by the subclass itself. Both methods are tied to "implementations" and cannot easily change behavior. This also reflects the separation principle of TRIZ principle. Now we use interface for each behavior, such as FlyBehavior and QuackBehavior, and let each class implements the interface behavior, then defined in Duck class as long as the interface of the instance variables, so in each Duck subclasses if want to have a particular behavior, as long as using this interface class instance variables to refer to a specific behavior.

Design principle 3: use combination more than inheritance.

Flying and calling these two different behaviors, we set up two different sets of behavior classes for them respectively, and then combine them through interface instance variables in the Duck class, which is called "combination". This makes the system highly resilient and allows for "dynamic behavior changes at runtime." Embodies the dynamic principle of TRIZ.

Model of the key is to get the variable in the system behavior of abstracting, encapsulate alone, with a set of behavior class (algorithm) to implement the specific interface, so that any class (such as Duck) if you want to have one of these algorithms clan algorithm, through interface instance variables can have the whole family of algorithms, in subclasses of the variable assignment again, and this also happens to be use of the dynamic characteristics of the theory of TRIZ, and separation principle.

5. High cohesion and low coupling in software engineering

The first thing to know is the meaning of cohesion and coupling: in software design, coupling degree and cohesion degree are commonly used as the standard to measure the degree of module independence. One of the criteria for dividing modules is high cohesion and low coupling.

Coupling is a measure of the degree of correlation between modules. The strength of coupling depends on the complexity of the module's indirect interface, the way the module is called, and how much data is transmitted through the interface. The coupling degree between modules refers to the dependencies between modules, including control relationship, call relationship and data transfer relationship. The more connections between modules, the stronger the coupling and the worse the independence. Reducing the coupling degree between modules can reduce the influence between modules, prevent the water wave effect caused by the modification of a module, and ensure the smooth progress of the system design. Cohesion and coupling are closely related. Modules with strong coupling relationship with other modules often mean weak cohesion, and strong cohesion usually means weak coupling.

The impact of a high degree of coupling: high degree of coupling, the maintenance of the code to modify a place will be involved in many places. If these coupling relationships are not clarified when modifying, the consequences can be catastrophic. Especially for projects with many requirements changes and multi-person collaborative development and maintenance, modification of one place will cause module errors that have already been running stably, which will lead to a vicious circle in serious cases and problems will never be solved. Development and testing are always laboring between various problems, which eventually leads to project delay, lower user satisfaction and higher cost. This has a bad impact on users and developers, and all kinds of risks are self-evident. In order to prevent these problems, one of the important means is to reduce the code coupling degree. But you can't have absolute zero coupling

Cohesion: indicates the length of internal aggregation and association. High cohesion means high degree of aggregation and association. Cohesion is determined by the relationship between classes, high, meaning the relationship between them should be simple, clear, not strong, or it will run into problems. The operation of one class affects other classes. High cohesion is recommended for module design due to its robustness, reliability, reusability, readability and other advantages.

This is the concept of software engineering, is the standard to judge the quality of the design, mainly for OO design, mainly to see whether the class cohesion is high, low coupling

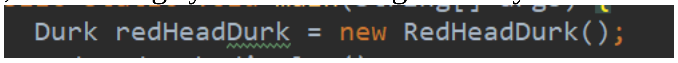
Combination with TRIZ theory:

"High cohesion, low coupling" is mainly described that each class needs the idea of responsibility separation in the object-oriented system. This is the idea of separation in TRIZ theory, where each class performs a specific, independent function, and this is called high cohesion. Coupling is the mutual invocation relationship between classes. If the coupling is strong and there are many mutual calls, it will affect the whole system, which is not conducive to maintenance and extension. The Settings between classes should be low coupled, but each class should have a high cohesion. Coupling is the scale of the interdependence between classes. If every object has a reference to all other objects, then a high coupling, it is not appropriate, because between two objects, potential flow too much information. Low coupling is required: it means that the object more independent of each other. To modify a class can be loosely coupled to minimize will modify other classes "chain reaction". Cohesion is a class of variables and method of the scale of the connection strength. High cohesion is desirable, because it means that I can perform a better job. Low cohesion is bad, because it shows that the elements of a class are rarely related. Related module between components is required. Each method should be highly cohesive. Most of the method only perform a function. Don't 'extra' instruction is added in the method, which can lead to method performs more functions.

6. The object-oriented nature of Java

The Java language is an object-oriented language. Its object-oriented main performance is inheritance, encapsulation and polymorphism, and these features are also the embodiment of TRIZ theory: encapsulation features reflect the idea of separation. Encapsulation can be thought of as a protection barrier that prevents the code and data of a class from being randomly accessed by code defined by an external class. To access the code and data for this class, strict interface control is required. The main function of encapsulation is that we can modify our implementation code without modifying the program fragments that call our code. Proper encapsulation makes code easier to understand and maintain, and enhances code security.

Polymorphism reflects the asymmetrical approach of TRIZ theory, and the overloading is also the use of asymmetric ideas, introducing symmetrical things into asymmetrical things, see Fig.5.

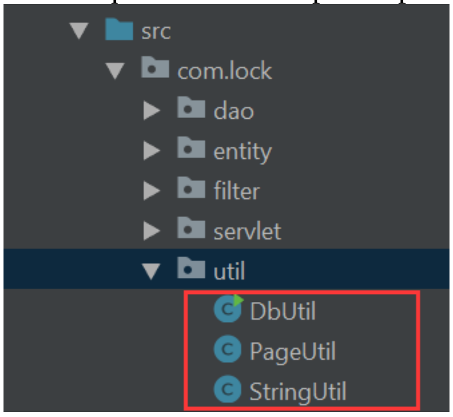


```
Durk redHeadDurk = new RedHeadDurk();
```

Fig. 5 A new object

For some scheduling problems of the main thread and the secondary thread, it also reflects the separation idea of the TRIZ invention principle. Concurrent processing is done through threads.

The basic encapsulation of basic public classes, strings, and database connection classes is also a reflection of the TRIZ theory, which is equivalent to a separate package, see Fig.6.



```
▼ src
  ▼ com.lock
    ▶ dao
    ▶ entity
    ▶ filter
    ▶ servlet
    ▼ util
        DbUtil
        PageUtil
        StringUtil
```

Fig. 6 Util class

7. The MVC pattern

The full name of MVC is Model View Controller, which is an abbreviation of model-view-controller. It is a software design paradigm. It organizes code with a business logic, data and interface display method. Business logic is aggregated into a single component, and there is no need to rewrite business logic while improving and personalizing the interface and user interaction. MVC has been uniquely developed to map traditional input, processing and output functions in the structure of a logical graphical user interface.

In the web page, the V(view) refers to the interface that the user sees and interacts with. For example, a web interface composed of html elements, or a client interface of software. One of the benefits of MVC is that it handles many different views for your application. There is actually no real processing happening in the view, it is just a way of outputting data and allowing the user to manipulate it.

M is the model, which means that the model represents business rules. Among the three components of MVC, the model has the most processing tasks. The data returned by the model is neutral, and the model is independent of the data format. Such a model can provide data for multiple views. Since the code applied to the model can be reused by multiple views only once, it reduces the duplication of code.

C is the controller that accepts the user's input and calls the model and view to complete the user's needs. The controller itself does not output anything and does any processing. It simply receives the request and decides which model component to call to process the request, and then determines which view to use to display the returned data.

This is also the practice of TRIZ theory, reflecting the separation of innovative methods, combined innovation methods. Through the revelation of TRIZ, the MVC model has been used a lot. It also brings a lot of benefits: reduced coupling, improved reusability, faster deployment, and improved maintainability.

8. Conclusion

Although the TRIZ theory focuses on mechanical engineering, it has an inspiring effect in many fields and a great contribution in the direction of software engineering.

References

[1] X.D.Wang,Wei Xiong: A trusted software technology conflict resolution method based on QFD and TRIZ[J], Journal of aviation, Vol.32(2011) No.1,p.36-38.

[2] F.J.Guan,H.T.Zhu: Application research of TRIZ theory in software development[J], Sme management and technology,Vol.3(2015),p. 311-312.

[3] G.Z.Cao: Effect research and software implementation based on TRIZ[D], Hebei university of technology,Vol2003,p.24-26.