I.J. Information Technology and Computer Science, **2014, 05, 58-64**
Published Online April 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.05.08

Substitute and Communication Pattern for an Internet Banking System

A. Meiappane
Research Scholar, Pondicherry University, Puducherry– 605014, India
E-mail: auromei@yahoo.com

Dr. **V. Prasanna Venkataesan**
Associate Professor, Pondicherry University, Pududherry– 605 014, India

***Abstract*−** The design patterns are the reusable component used in the development of the software, which delivers enhanced quality software to the end users. The design patterns are available for user interface, mobile applications, text classification and so on. There are no design patterns for internet banking applications. This motivated to mine the design patterns for internet banking application from the document of Business Process Management (BPM) by using the qualitative research technique. The nonfunctional quality attribute of software architecture is enhanced by using the design patterns. In this paper the mined two patterns are presented namely substitute pattern and communication pattern for internet banking application.

***Index Terms*−** Design Patterns, BPM, Qualitative Research, Substitute Pattern and Communication Pattern

I. Introduction

The software architecture of a system is the set of structures needed to reason about them or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [1].

Architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.

Software architecture plays a major role in the development of any software systems. To build such systems the design of architecture should satisfy not only the functional requirements but also the non-functional requirements or quality attribute requirements such as reusability, performance, maintainability, modifiability, availability, scalability and so on [2]. Consideration of quality attributes adds strength to the software system and makes it adoptable and adaptive. Quality attributes are enhanced by using the design patterns in the software system [3].

Design pattern in architecture ameliorates the non-functional quality attributes. The pattern provides solution to the problem at a particular context [4]. The developer has to use pattern appropriately in the system, instead of starting from scratch that is reinventing the wheel again. Developing time of software is minimized by the use of pattern, which in turn reduces the cost of development of the software.

The software architecture is an innovative and interesting area in the software engineering research field. More paradigms have been identified and commonalities are operated in design of software architecture. By using design patterns, the software quality attributes are improved [3]. An architecture designed through pattern addresses the principle difficulty in the software development and manages the complexity, which increases the technological improvement.

The remainder of this paper is organized as follows: In section II, POSA has been discussed where it represents about the Patterns and its qualities and also the elements/representation and various types of patterns. In section III, Business Process Management (BPM) has been explained which reveals about the pattern identification and its various approach. In section IV, where the crosscutting concern has been identified by using the qualitative research method. In section V, the pattern mining has been narrated. The conclusion and future enhancement has been explained in the section 6.

II. Pattern Oriented Software Architecture (POSA)

When experts need to solve a problem, they invent a totally new solution. More often they will recall a similar problem they have solved previously and reuse the essential aspects of the old solution to solve the new problem. They tend to think in problem-solution pairs.

Identifying the essential aspects of specific problem - solution pairs leads to descriptions of problem - solving patterns that can be reused. The concept of a pattern as used in software architecture is borrowed from the field of (building) architecture.

The goal of patterns within the software community is to help software developers resolve recurring problems encountered throughout all of software development. Patterns help create a shared language for communicating insight and experience about these problems and their solutions.

One of the widely used definitions is: "A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts and presents a well proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate".

2.1 Patterns

Definitions:

1. Giving solution to the recurring problem at the particular context [5].

2. Pattern falls into the family of similar problems**.** It is the process of distilling common factors from the system.

3. Pattern is said to be as the relation between the context, problem and solution.

4. A pattern system provides a pool of proven solutions to many recurring design problems. On another it shows how to combine individual patterns into heterogeneous structures and as such it can be used to facilitate a constructive development of software systems.

5. Patterns are the proven solution and not just a solution.

6. Patterns are useful, useable and used

Good patterns do the following:

- **It solves a problem:** Patterns capture solutions, not just abstract principles or strategies.
- **It is a proven concept:** Patterns capture solutions with a track record, not theories or speculation.
- **It describes a relationship:** Patterns don't just describe modules, but describe deeper system structures and mechanisms**.**
- **The pattern has a significant human component:** All software serves human comfort or quality of life; the best patterns explicitly appeal to aesthetics and utility.

Qualities of Pattern

1) **Encapsulation and Abstraction:** Each pattern encapsulates well defined problem and its solution in a problem domain.

2) **Openness and variability:** Each pattern should be open for extension or parameterization by other patterns so that they may work together to solve a larger problem.

3) **Generativity and Composability:** Each pattern once applied generates a resulting context which matches the initial context of one or more other patterns in a pattern language.

4) **Equilibrium:** Each pattern must realize some kind of balance among its forces and constraints.

2.2 Elements of Pattern:

Each pattern should contain the following elements [6]

Pattern Name

It is used to describe a design problem, its solutions and also its consequences. It increases design vocabulary. Pattern name is used to enhance the communication. Naming a pattern increases our design vocabulary. It should be a meaningful word or a single phrase.

Context

It is the situation giving rise to a problem. It is the description of the initial state before the patterns are applied. This should be a recurring situation.

Problem

A recurring problem arising in the situation i.e., at a particular context describes when to apply the pattern. It explains the problem and its context. It might describe specific design problems such as how to represent algorithms as objects. It refers to the goal we are trying to achieve in this context and also refers to any constraint in the context. Sometimes the problem will include a list of conditions that must be met before it makes sense to apply the pattern

Solution

A proven solution to the problem, describes elements that make up the design, their relationships and their responsibilities. It does not describe specific concrete implementation. It tells about Abstract description of design problems and how the pattern solves it. It is a general design that anyone can apply which resolves the goal and set of constraints.

Forces

A description of the relevant forces and constraints and tells how they interact and conflict with one another and the goals we wish to achieve.

Resulting Context

The state of the system after the pattern has been applied. It tells about the consequences (both good and bad) of applying the pattern and also other problems and patterns that may arise from the new concept. This element describes which forces have been resolved and how and which remains unresolved.

Examples

One or more sample applications of the pattern which tells a special initial context, how the pattern is applied to and transforms, that context, and the resulting context.

Rationale

A justifying explanation of steps or rules in the pattern and also of the pattern as a whole in terms of how and why it resolves its forces in a particular way to be in alignment with desired goals, principles. Rationale provides insight into its internal workings.

Related Patterns

It is the static and dynamic relationships between this pattern and others within this same pattern language or system. Related patterns often share some common forces.

Known Uses

Describes known occurrences of the pattern and its application within the existing system.

2.3 Types of Pattern

- Architectural patterns
- Design Patterns
- Idioms

Architectural Pattern

It is said to be as the high level of patterns. Architectural pattern tells about the fundamental structural organization of the system. It provides the set of pre defined subsystems and specify their responsibilities and also includes rules and guidelines. In general Architecture pattern tells how to build the system.

Design Patterns

It is said to be as the medium level of patterns. Design Pattern is a general repeatable solution to a commonly occurring problem in software design. It refines the subsystem or the components. It gives the template of how to solve the problem. The design we obtain from the design pattern is directly transformed to the code [6].

Idioms

It is the low level of patterns. It describes how to implement the particular aspect of components. It addresses both the design and the implementation. It eases communication among the developers and speed up software development and maintenance. A collection of such related idioms defines the programming style.

2.4 Motivation of Design Patterns

Design pattern is applied to many domains like e-commerce applications, online share trading and internet shops. The contribution of design pattern towards the financial domain is limited. The usage of banking is 46%, mobile banking is 30% whereas the internet banking is 77%. This inference reveals that the usage of internet banking is high [7], but design pattern to internet banking is less.

These made to concentrate on the internet banking. The importance to design pattern brings out the good architecture. The customer satisfaction and retention are turning to be the crucial factors for success in internet banking [8].

The design pattern for internet banking is not available so far for the developer. So the design pattern has to be implemented to improve the non-functional quality attributes of the architecture. In order to achieve this, requirements or issues of mining the design pattern have to be identified and this motivates to formulate the research problem.

Patterns are extremely valuable tools for capturing and communicating acquired knowledge and experience to improve software quality and productivity by addressing fundamental issues in the development of software.

The following are the various factor that motivates for design patterns.

- Designing reusable code is difficult. Patterns provide reusable frameworks and successful reusable design exists.
- It tells about the detailed design of the framework.

III. Business Process Modeling (BPM)

BPM is the process of representing processes of an enterprise, so that the current process may be analyzed and improved. BPM is typically performed by business analysts and managers who are seeking to improve process efficiency and quality. In this process we are going to improve the internet banking application so that we can able to get a reusable application from which lot of users feel ease to use the internet banking application [9].

3.1 Pattern Identification

Architecture and design patterns, as demonstrated solutions to recurring problems, have proved practically important and useful in the process of software development. They have been extensively applied in industry. Discovering the instances of architecture and design patterns from the source code of software systems can assist the understanding of the systems and the process of re-engineering. More importantly it also helps to trace back to the original architecture and design decisions, which are typically missing for legacy systems [10].

Software architecture is the key artifact in software design, describing the main elements of a software system and their interrelationships. We present a method for automatically analyzing the quality of architecture by searching for architectural and design patterns from it.

In addition to approximating the quality of the design, the extracted patterns can also be used for predicting the quality of the actual system.

In this paper we are going to identify patterns in the internet banking application where the quality of the architecture is improved [11]. We are going to analyze the overall process in the internet banking and apply any pattern identification approach and we identify various patterns so that we can able to achieve reusability in our system.

3.2 Pattern Identification Approach

In order to identify patterns some of the methods are available. Pattern identification is done by using the methods like similarity scoring, using multilayered approach (DeMIMA), through random walks [12].

The researchers confirmed that the interaction design patterns can be identified using qualitative research approach [13] [14] [15] [16] [17].

The Tesch [18] and Patton [19] literature approach distinguishes between the inductive and deductive process of analysis. The inductive analysis performs in the data while the deductive analysis works in a theoretically informed analysis framework i.e., to support the articulation of patterns.

The following reveals the researchers description of qualitative research:

1) The researcher acts as the human instrument for data collection.

2) Qualitative researchers predominantly use inductive data analysis.

3) The researcher Eisner [20] brings out qualitative research reports are descriptive and it results as the presence of voice in the text.

4) The trustworthiness is evaluated and judged using qualitative research.

Among all these approaches the efficient method is the qualitative research method to identify patterns [21]

IV. Crosscutting Concern

Understanding of concerns plays an important role in successful software development. Modularization of concerns is important for software development. Object oriented programming paradigm provides an ease of modularization of basic concerns. There are some concerns whose implementation cannot be modularized using object oriented paradigm like profiling, logging etc. The implementation of such concerns remains scattered throughout the source code. Such concerns are called crosscutting concerns [10]. Identification of crosscutting concerns plays an important role in aspect mining, defect detection and software maintenance.

Cross-cutting concerns are aspects of a program which affect other concerns and results in either scattering or tangling [22].

Every concern might not get modularized into a separate module. Such concerns whose implementation is scattered over more than one module are called crosscutting concerns. Such concerns lead to tangling of code.

Poor modularization of crosscutting concerns results in source code which has more defects and is difficult to maintain. For development of good quality software, it is essential to identify crosscutting concerns

Identification of crosscutting concerns at different stages of software development is essential for 3 reasons.

1) Refactoring legacy system to aspect oriented system.

2) For modularized implementation of the concern.

3) For appropriate distribution of testing effort error prone crosscutting code need to be identified.

In the internet banking application analysis on various processes [23] should be done on the internet banking in order to find the crosscutting approaches. Considering all these features the pattern should be identified.

To identify the crosscutting concern the Theme/doc technique is selected, from the available various techniques. This technique is well suited at requirement and design level to identify the cross cutting concerns [24]. Mahemoff and Johnston (2001) [25] argue after reviewing the scope of a set of pattern languages that the limiting the scope of design patterns to one domain application results well.

The crosscutting concerns of the internet banking system are identified with the theme/doc technique, which has been elaborated in the literature review. The design patterns are mined for internet banking application from identified crosscutting concerns by using the qualitative research approach.

V. Pattern Mining

Pattern mining is that mining of various patterns in the internet banking system by using BPM. It is used to represent the process involved in the internet banking and is used to analyze the process and then we apply crosscutting approach in this model.

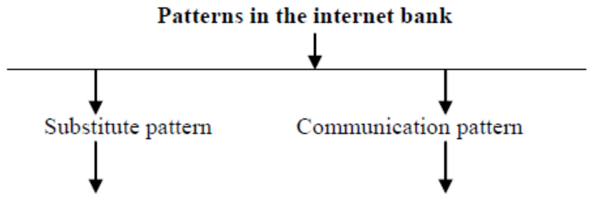
The Crosscutting Concerns are the aspect of program which affects the other concerns of the program. These concerns cannot be decomposed from the rest of the system in both the design and the implementation and can result in either scattering, tangling. This crosscutting concern is used to improve the modularity of the system.

In order to mine the pattern, it is so essential to identify the cross-cutting concern in Business Process Management (BPM) document. Here the identification of patterns is discussed in this section. Based on the identified patterns, the pattern architecture has been proposed for internet banking.

The definition for pattern from the various studies states that it is a relation between a certain context, a problem and a solution. The literature review concludes that the pattern provides a proven solution to a problem at a particular context. The identification of pattern for internet banking application deals with knowing the flow of BPM. The cross-cutting concern has to be identified in the flow of BPM. With the help of cross-cutting concern, the pattern has to be mined to the problem to provide a solution. The literature survey has enabled to mine the design pattern to the specific problem. The literature review concludes that the patterns are of the following two types, namely architecture pattern and design pattern. The architecture pattern deals with the overall design of the system and the design pattern deals with the component and sub-component of the system.

The internet banking system provides lot of services available to its end users. All these services resemble the same. Hence we need to use the crosscutting concern approach in order to find different patterns available in the internet banking system.

There are various services involved in the internet banking like account, Third party transfer credit card, debit card, mutual fund, loan, insurance etc[26]. Based on the analysis a model is drawn for business process modeling in the internet banking.



Credit card payment	loan communication
Debit card payment	insurance communication
Cheque Payment	Payment communication
DD payment	

Fig. 1: Patterns Mined from the Internet Banking System

- **Substitute pattern Mined from proxy pattern**

The payment pattern code is scattered in credit card, debit card, ATM because these acts as a proxy for ha amount. We can do the payments by using all these methods we can use the same code with the slight modification. This Substitute pattern is similar to that of the proxy pattern.

- **Communication pattern mined from the mediator pattern**

Here the communication is centralized by using a mediator. In the internet banking application mediator is needed for loan, insurance, payment processes. The coding is similar for all these cases with slight modification. This communication pattern is same as that of the Mediator pattern.

These are the two patterns mined from that of the internet banking system. These two patterns are identified and it can be reused whenever needed. These two patterns can be reused in all the applications. The above mined pattern mainly achieves the reusability and also the quality attributes like extensibility, maintainability, modifiability etc. These two patterns are catalogued by using the Eric Gamma method [2] and they are presented as follows

SUBSTITUTE (PROXY) PATTERN:

Context:

Whenever there is a need for the substitute to access the real object we go for the substitute pattern. We use this if we need to access the object by using the substitute instead of calling it directly.

Problem

The user whenever performs certain essential things there should also be a substitute which is equally valid or should have a equal value.

Solution:

The substitute pattern is mainly based on that of the proxy pattern. The proxy is that it acts as a surrogate or a placeholder in order to access the other object. We use substitutes for Voter id as driving licensee, Ration card as smart card.

In case of the internet banking system in order to access the amount the credit card or debit card has been used as a substitute. The account is retrieved by using the proxies.

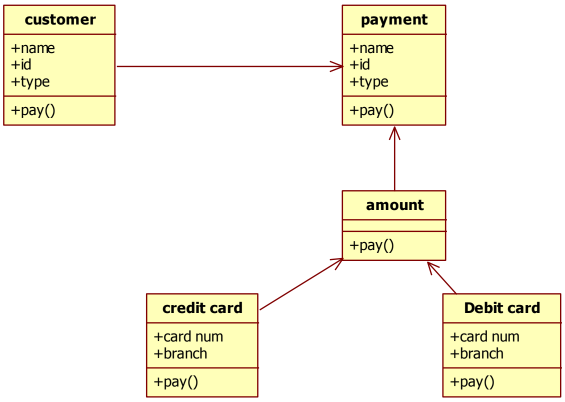


Fig. 2: Substitute pattern class diagram

COMMUNICATION (MEDIATOR) PATTERN:

Context:

Whenever we need the communication which clients and the system communicates through the server (admin) we use the mediator pattern. Through mediator we can do the communication.

Problem:

The discussion room is essential for all type of applications for that there is a need of mediator pattern, which results in direct and immediate communication.

Solution:

Here the mediator pattern is used in order to implement the communication pattern where all the clients communicate with the Server (admin) if they are registered with the server. In the internet banking system we use the mediator pattern for implementing the chat applications where all the clients with the admin in order to clarify any doubts.

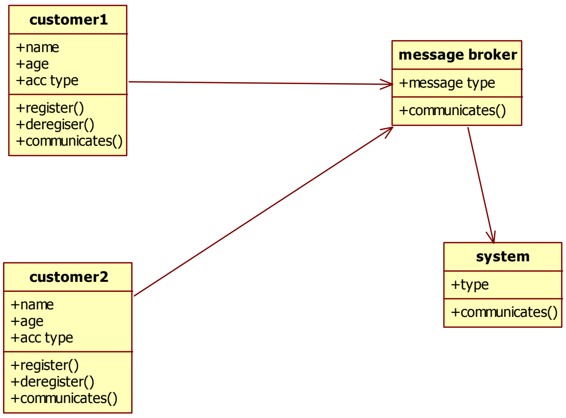


Fig. 3: Communication Pattern Class Diagram

VI. Conclusion and Future Enhancement

Various patterns like Proxy Pattern, and Mediator pattern, have been identified. By implementing and using all these patterns we can able to achieve reusability in the internet banking application. It is more effective from the side of customers as well as web designer. By implementing this modularity is improved. Thus the patterns are identified and achieve reusability in the internet banking application. After implementing the above mentioned patterns in the internet banking architecture we need to evaluate the system. The SAAM evaluation is proposed so that we can compare the results of the pattern architecture and the legacy or traditional system.

The future Enhancement is to evaluate the different architecture of the internet banking system by considering various quality attributes. By using the SAAM method the existing system needs to be evaluated and also the pattern based system is evaluated. By assuming weights of various scenarios which interact between classes and methods, in both the cases the results are compared and it provides the need to provide an efficient architecture of the System. The different architecture needs to be evaluated and the best one is chosen based on the evaluation results.

References

[1] Krutvhen ,Obikk, Stafford, “The past, Present ana the future of the Software Architecture”, IEEE Xplore Digital Library, 2006.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison Wesley, 1995.

[3] Francisco Montero, V ćtor L ópez-Jaquero, Jos é Pascual Molina (2008) Applying Usability Patterns in e-Commerce Applications., 111-120.

[4] Neil B. Harrison, Paris Avgeriou, Uwe Zdun, "Using Patterns to Capture Architectural Decisions," IEEE Software, vol. 24, no. 4, pp. 38-45, July/Aug. 2007, doi:10.1109/MS.2007.124

[5] Dhamayanthi N, Thangavel P, Structural Design Patterns and .NET Framework 2.0 ,Dhamayanthi, Journal Of Object Technology,2006

[6] Erich Gamma, Richard Helm ,Ralph Johnson and Vlissides,”Design Pattern Element Of reusable object oriented software”, ISBN:0-201-63361-2

[7] http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture_Study_Bankingtech_E_DE.pdf

[8] Bauer, H., Hammerschmidt, M & Falk, T. (2005). Measuring the quality of E-banking portals. International Journal of Bank Marketing, Vol. 23, No. 2, pp. 153-175.

[9] Mr. A. Meiappane, Ms. J. Prabavadhi, Dr. V. Prasanavenkatesan, "Visitor Pattern: Implementation of Enquiry Pattern for Internet Banking", International Journal of Engineering Research and Technology (IJERT), May – 2012.

[10] Xiaohong Yuan and Eduardo B, "Patterns for business-to-consumer ecommerce Applications", Journal of S/w Engineering & App (IJSEA), 2011.

[11] Mr.A.Meiappane, Ms.J.Prabavadhi, Dr.V. Prasanavenkatesan, "Strategy pattern: payment pattern for internet banking", International Journal of Information Technology and Engineering (IJITE), March 2012.

[12] Yann-Gae¨l Gue´he´neuc, "DeMIMA: A Multilayered Approach for Design Pattern Identification", IEEE transactions on software engineering, vol. 34, no. 5, september/october 2008

[13] Guy, E. S. (2003). Developing a pattern language from observation of the development of shared information spaces. In ECSCW 2003 Workshop 6: From Good Practices to Patterns, 8th European Conference on Computer Supported Cooperative Design,, Helsinki, Finland.

[14] Guy, E. S. (2005). ". . . real, concrete facts about what works . . . ": Integrating evaluation and design through patterns. In International ACM Conference on Supporting Group Work. Group '05, pages 99–108, Sanibel Island, Florida. New York: ACM Press.

[15] Martin, D., Rouncefield, M., Rodden, T., Sommerville, I. and Viller, S. (2001). Finding patterns in the fieldwork. In ECSCW'01, Bonn, Germany. Kluwer.

[16] Martin, D. and Sommerville, I. (2004). Patterns of cooperative interaction: Linking ethnomethodology and design. ACM Trans. Comput.-Hum. Interact., 11(1):59–89.

[17] Arvola, M. (2006). Interaction design patterns for computers in sociable use. International Journal of Computer Applications in Technology, 25(2/3):128–139.

[18] Tesch, R. (1990). Qualitative research: analysis types and software tools. Falmer Press, New York.

[19] Patton, M. Q. (2002). Qualitative research and evaluation methods. Sage Publications, Thousand Oaks, California

[20] Eisner, E. W. (1991). The enlightened eye: Qualitative inquiry and the enhancement of educational practice. New York, NY: Macmillan Publishing Company.

[21] Charles Zhang, Member, IEEE, and Hans-Arno Jacobsen, "Mining Crosscutting Concerns through Random Walks", IEEE transactions on software engineering, vol. 38, 2012.

[22] R.M.Noorullah, "Comparative Study in Utilization of Creational Design Patterns in Solving design problems", International Journal of Engineering and Technology Volume 1 No. 3, December, 2011.

[23] Marcel Birkner, Objected-Oriented Design Pattern Detection Using Static And Dynamic Analysis In Java Software, Graduate Program At University Of Applied Sciences Bonn Rhein-Sieg Sankt Augustin, Germany August 2007

[24] Aravinder Kaur, Kalpana Johari, "Identification of crosscutting concerns: A Survey" International Journal of Engineering Science and Technology, Vol.1(3), 2009,166-172.,

[25] Mahemoff, M. J. and Johnston, L. J. "Usability pattern languages: the language aspect", In Human-Computer Interaction: Interact 2001, pages 350–358, Tokyo, Japan. IOS Press.

[26] Lubor Sesera softec & fiit stu Slovakia, Fundamental banking patterns, ACM 2010.

Authors' Profiles

Mr. A. Meiappane, Associate Professor, Department of Information Technology, Manakula Vinayagar Institute of Technology, Pondicherry University, Puducherry. He holds M.Tech. In Computer Science and Engineering and pursuing his Ph.D in Computer Science & Engineering (Banking Technology) from Pondicherry Central University, India. His areas of Interest include Software Engineering, POSA, networks and distributed computing. He has many International and National Journal Publications. He has also reviewed software engineering books for Tata McGraw Hill.

Dr. V. Prasanna Venkatesan, Associate Professor, Dept. of Banking Technology, Pondicherry University, Puducherry, India. He has more than 20 years teaching and research experience in the field of Computer Science and Engineering. His research interest includes software engineering, Business intelligence, Software Architecture and banking technology. He has designed Multilingual Compiler. He has many international Journal publications. He is co–author of the book titled as "Service Composition and Orchestration: Concepts and Approaches" published by VDM Verlag Dr. Muller E.K.