Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IDENTIFYING SIMILAR PATTERN OF POTENTIAL ASPECT ORIENTED FUNCTIONALITIES IN SOFTWARE DEVELOPMENT LIFE CYCLE

[1]MAZEN ISMAEEL GHAREB, [2]GARY ALLEN

[L]ecturer Assistant., Department of Computer Science, College of Science and Technology, University of Human Development

[2]Assoc. Prof., Department Computing and Engineering , School of Computing and Engineering,,

University of Huddersfield

E-mail: [1]u1152287@hud.ac.uk , [1]mazen.ismaeel@uhd.edu.iq, [2]g.allen@hud.ac.uk

ABSTRACT

Aspect Aspect-oriented programming is known as a technique for modularizing crosscutting concerns. However, there are no clear rules to help detect and implement Aspects in the software development lifecycle. Consequently, class developers face changeability, parallel development and comprehensibility problems, because they must be aware of aspects whenever they develop or maintain a class. These problems can be mitigated by using adequate design rules between classes and aspects in the design stage and then in implementation process. We need to define a similar pattern of aspect for many systems to explore. This pattern will help development process from the initial phases, especially with the aim of supporting modular development of classes and aspects. Adding to that shows some design patterns relationships with aspects. We discuss how several languages improve crosscutting modularity without breaking class modularity. We evaluate our approach using a real case study and compare it with other approaches to detect the Aspect Oriented in Design phase of software developments.

Keywords: *Aspect Oriented Programming; ASDL (Aspect Oriented Design language),cross cutting concern, modeling ,Design pattern.*

1. INTRODUCTION

A In the software development process, Aspects are difficult to identify because they are usually tangled and scattered across the entire system. Some aspects are obvious can be identify but others are more subtle difficult to identify. This makes it complex to locate all points of the system where aspects should be applied. To address these aspects in the software development lifecycle, developers need more support to find and analyze aspects in requirements documentation. AODL (Aspect Oriented design language) is a notation used to show the interaction between traditional UML models of base (Object Oriented) code and Aspect extensions, such as point cuts, join points and advice. The challenge of this study is to find similar pattern of aspect for many systems to explore its benefits from the initial phases of the development process and find the relationship between design patterns and aspect oriented programming to meet a modular solution for specific issue in the software engineering field. Many studies have been conducted on Aspect programming developments during recent years. One of the aspect viewpoint according to [1] defines Aspect, as “aspects tend not to be the system's functional decomposition, but rather to be properties that affect the performance or the semantics of the components in a systematic way” Kiczales attempts to differentiate between the aspects and components [1]. Another meaning of Aspect Oriented Programming is to conquer the issues emerging from crosscutting concern. It helps engineers to change the Object Oriented model progressively, so the crosscutting improves code reuse rate and practicality [2]. Aspect Oriented programming helps designers to conquer the issues connected with code scatterning and tangling over numerous framework units by lessening the duplication the code. Its backings a few crosscut concerns, for example, join points, point cuts, and

Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

advice. A Join point is one of the few purposes of the framework where concern crosscut a strategy or constructor, while a point cut is an inquiry about selecting obliged join points. Thusly, the guidance is the development that makes a move where the join point coordinated: some time recently, after and around in the particular framework [3]. This exploration will examine to utilize AODL aspect Oriented Design Language notations to symbolize the systems in the design stage and potential viewpoint in requirement stage and advancement stage also. Gary Allen and Saqib Iqbal [11] propose these documentations . This documentation represent the aspect and regular object utilizing UML documentations and models. Adding to that if would we be able to recognizing the Aspects in right on time outlines it is conceivable by applying it on Design designs. There were a few works research this issue utilizing the contextual analysis of the Car Crash management framework [11]. As indicated by [5] studies have demonstrated implementations of six GOF outline designs (Observer, Mediator, Prototype, Strategy, State and Abstract Factory) with viewpoint executions, the outcomes that shows most aspect-oriented programming enhances the configuration of base item arranged code. In next section will explain the previous work on Aspect Oriented Programming .

our Idea is to detect the Aspect in requirements stage using approaches combines the view points , goal based and them approaches to find the potential aspect in early stage of software design .

The remainder of this paper will be structured as follows. In Section 2 we describe the state of the art about AOP technology. In Section 3, we introduce UML Language AOP Notation. In Section 4, we describe AOP and its relationship with design patterns. In Section 5 we described the methodology approach to identify aspects. Section 6 we show the result of our survey. In Section 7 we shows our conclusion and future work.

2. STATE OF ART

2.1 Invintion of AOP

Aspect Oriented programming (AOP) was found quite a while prior, before Demeter group. In 1997, Aspect arranged writing computer programs was authoritatively uncovered by Gregor Kiczales, with his partners in gathering name ECOOP97 [4]. AOP Aspect Oriented programming created approachs called Subject Oriented Programming. As per [6] AOP is an advancement method reduces that enhances capturing so as to program improvement the space related procedures in the framework, to better fit genuine area issues into code; in this manner it decrease troubleshooting time and expands clarity.

2.2 Introduction to Aspect Oriented Modelling

Engineers ought to be mindful of, and comprehend the product displaying or structural planning before gazing actualizing AOP. AOM (Aspect Oriented Modeling) is a way to deal with produce a sensible viewpoint arranged structural model. Early utilization of AOM being developed stage will lessen the product advancement danger of contentions and undesirable conduct rising amid usage. The cross cutting component is normal in AOP and AOM, however the distinction is between the stratagems versus source code, it could rise contrast method in speaking to it. Case in point, the code can speak to in single usefulness, while a model can speak to the framework with distinctive graph sees. Another contrast in the middle of AOP and AOM in code is viewpoint weaving is basically concerned with embeddings usefulness at system execution. AOM module comprises of real segments: essential model, angle model and arrangement model Fig.1[7] beneath demonstrates this model.

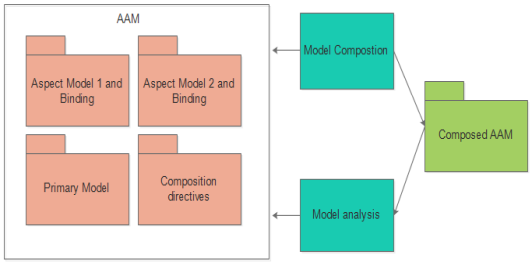

Fig 1 (AOM Approach Components)

Fig1 demonstrates an essential model, for example, an UML chart to depict a fundamental building design class outline and intelligent graph. The viewpoint model depicts a consistent compositional arrangement. An Aspect Oriented Architecture Model is an intelligent perspective of programming structural engineering. Another meaning of Aspect Oriented Allowing so as to program is rearranging of the advancement process detachment of formative undertakings. Likewise, Aspect Oriented Making so as to model enhances an object oriented application it more measured. AOP takes care of the code scattering issue in OOP. Diffusing means the issue of shared the usefulness of an application spread among numerous classes, which has a tendency to ease off the application and make it hard to keep up. Along these lines, AOP tackles this issue by uniting the scattered code in the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

perspective. A perspective is a cross cutting structure. It executes the usefulness, for example, security, logging and industriousness and non functional requirements.

2.3 Aspect-Oriented language development

AspectJ used to execute AOP, which is a basic viewpoint situated programming augmentation for the Java language. It is an open source programming expansion of Eclipse. Also, it will bolster measured usage of a scope of crosscutting concerns [9]. An AspectJ system comprises of two noteworthy parts, the first part is the base code, for example, classes and interfaces to do a fundamental usefulness of the project, and the second part is the viewpoint code, which incorporates the angles for catching crosscutting concerns in the system [5]. Perspective backings the primary AOP develops of join points, point cuts and aspects. A join points is a dynamic execution point in the system. Point cuts comprise of an accumulation of join focuses. An Advice is a to some degree extraordinary technique joined to the point cuts. At long last, an angle is a particular unit of AOP. Fig. 2 demonstrates the procedure of viewpoint improvement strategy [10].

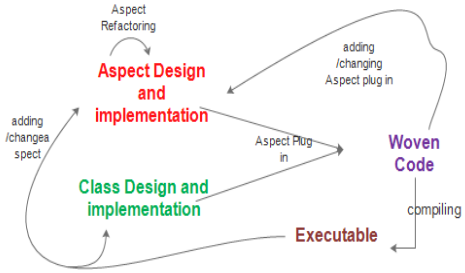

Fig. 2 (Process Of Aspect-Oriented Development Stages)

Late research demonstrates that there is not an apparatus to bolster a aspect oriented programming , for example, UML (Unified Modeling Language) . One of the ways to deal with outline an aspect documentation is to extend the UML documentation to bolster Aspect oriented units called Aspect Oriented Design model AODM. Accordingly, this methodology will help to demonstrate the angle programming weaving component and spoke to in UML, which will help engineers to create perspective programming documentation dialect [11]. AODM may demonstrate a Aspects as a secluded unit of crosscutting execution, which goes about as a compartment of the given individuals in the bit of source code [12].

As indicated by Stein "AspectJ is a usage for perspective situated programming in Java language ", including that cross cutting is a piece of the aspect that determines where the crosscutting code has been woven into base classes. Join Points in AspectJ are standard focuses in executable element programs. Join points present numerous activities, for example, calls to constructors and system execution. Likewise, they call classes and item instatement. In AspectJ Point cuts comprise of joint focuses. It indicates at which of the join focuses specific crosscutting conduct ought to execute. As far as a designator point cuts are 'if', 'this', 'objective', 'inclinations, or 'stream'. Designers will choose Join focuses relying upon the dynamic setting amid execution of the base code[13] . The creator ought to indicate at what time on the execution the guidance is to execute for example some time recently, after, or around particular source code. Another essential unit in Aspect J is presenting an extra part sort of classes, for example, strategies, constructor and another field for the class. What's more, it may change the super class kind of super interface by embeddings new introduction and speculation relationship to the class structure. Until this point there are so many research work need to identify aspect in design phase and development of software lifecycle but without any clear and bold guidelines to identify the Aspect Oriented model.

3. UML LANGUAGE VS NEW AOP NOTATION:

UML language is item arranged programming documentation language. UML gives the fundamental building pieces to model programming frameworks, for example, abstraction, relationships and charts. Adding to UML will give broadened UML documentation, for example, labeled qualities used to connect self-assertive data to a model component. Other than that, the augmentation absolutely bolsters new building obstructs that drive from existing ones. This new building called stereotyping, have the same structure (properties, association and operations) as the base framework obstruct that flourish with it.

In this manner, an UML augmentation has the capacity speak to an AspectJ fundamental reflection, for example, a Join points, Point cut and advice. Fig.3 delineates an UML representation for Join Points just, and demonstrates the correspondence connections to make or decimate an occurrence. In this manner, UML can't relegate or speak to the Join Point. AODM recommends representing so as to take care of this issue the correspondence as a pseudo operation that can just

Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

compose and read for a particular field. This verifies no execution may happen without calling a constructor or the instatement. Fig.4 demonstrates that UML could speak to a message that go between two cases. As it appears that the join focuses demonstrate the unique sorts of generalizations, for example,set ,get,execute, and initialize.

In AODM, point cuts are spoken to as unique generalization operations named as Pointcuts. As it is indicated in Fig. 3 (Stein, D., Hanenberg, and S. And Unland R., 2002).

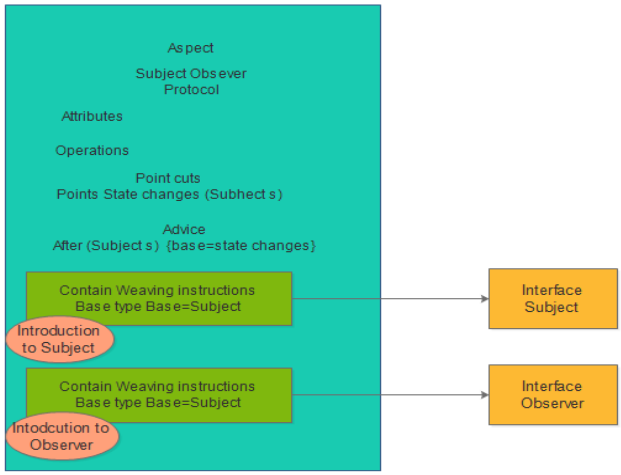

Fig. 3 (Model Of Aspect Oriented Design)

While in UML notation point cuts have an operational definition that has an arbitrary number of (output-only) parameters and their declaration and implementations as it shows in Fig. 4.

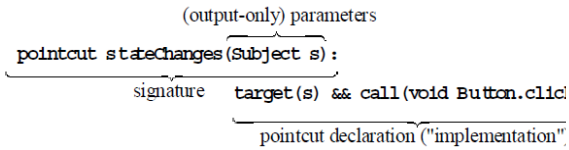

Fig. 4(Points Cut In Aspect And Similar To Operation In UML)

Like Point cuts, Advice can be spoken to as an operation, however one semantic distinction is that Advice does not have an extraordinary identifier. Hence, this may be a major clash in Aspect. Along these lines, AODM has illuminated this issue by characterizing by pseudo identifier that can't be overdriven. As it shows up in Fig. 5 .

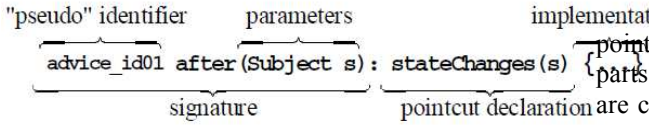

Fig. 5 (Advice And Operation Similarities)

In Gary and Iqbal the creators propose another documentation of AODL Aspect Oriented Design Language, this Language serves to show the Aspect with their properties and attributes alongside a conventional UML article chart. We will attempt to utilize this documentation in this paper . Both Aspect and Object can be utilized inside of an outline for single structure. This diminishes architect working expense to work with two distinct stages. In this manner, engineers picked UML to stretch out to contain Aspect for some reasons. One essential reason is that UML is most utilized apparatus for demonstrating. Furthermore, it is less demanding for designers to utilize one device rather two instruments together. At last, it is anything but difficult to utilize UML extensibility to plan Perspectives in light of the fact that it is anything but difficult to characterize another documentation and utilization them with the center documentation. In AODL utilizes an aspect documentation like class documentation in UML to display different segments, for example, aspects and point cuts. In any case, there is a connection between these parts. Case in point, point cuts contain the join point which guidance straightforwardly relies on upon. Accordingly, every part has its own qualities; subsequently [4] claim that AODL ought to speak to every documentation as one of a kind documentation, as demonstrated underneath in Fig.6:

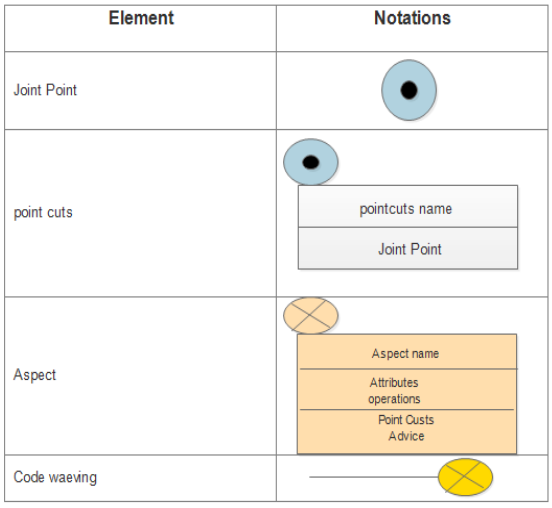

Fig. 6 (AODL proposed component notations)

In the AODL outline documentation, join points are spoken to as a snare. They unite alternate parts of the system with the point cuts. Point cuts are clarified as a rectangle box with a gathering of related join points. The container image is utilized for Aspects as a result of them having comparative

Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

qualities to classes in their conduct, as it shows above. Aspect documentation looks like class documentation likewise it has same similitude's of class and the cross circle demonstrates the cross cutting concern of the perspective. Code weaving is related which join the aspects with classes where aspect code is woven in. In addition, there are two models to plan weaving procedure, aspect static outline and aspect dynamic graph. Aspect Programming needs to demonstrate the join focuses in the programming. Arrangement graph in UML will demonstrate the join point in right on time configuration stage. This chart called a join point distinguishing proof graph. The conduct of join points is displayed utilizing a movement outline, which demonstrates the spot of join points and the framework action. Fig. 7 demonstrates to them beneath.

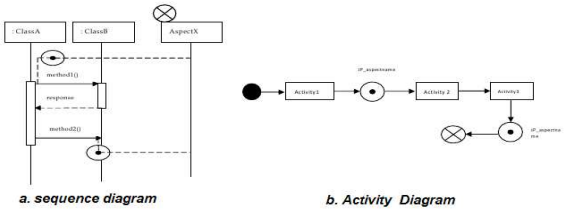

Fig. 7 (Sequence And Activity Diagram With New Notation Of AODL)

The fundamental AODL documentation is the Aspect documentation, which is spoken to as a major rectangle with numerous characteristics and operations. It has the aspect name at the top and circle cross to demonstrate the cross concern conduct. Fig.8 demonstrates a regular representation of an Aspect in AODL.

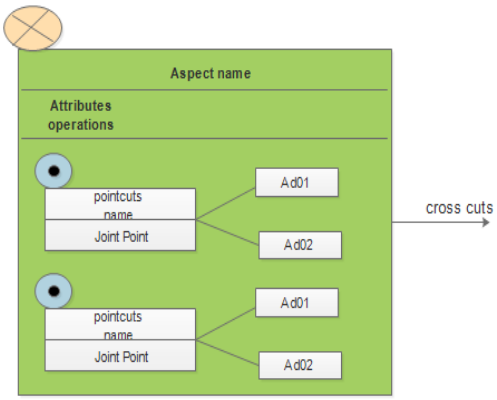

Fig. 8 (Aspect Representation In AODL)

aspects can be recognized in the early phases of improvement development using use case outlines, as demonstrated by Ivor Jacobson [10]. Nonetheless, Ivor contended that aspects couldn't be executed using use case diagram in light of the fact that tangling issues of the part in the use case outline. While Iqbal recommends that to repetitive the calling other part of utilization case and create utilization case segment independently. In this way, it is easy to find a crosscutting concern in use a case study [5]. Along these lines, it is anything but difficult to locate a crosscutting concern being used a contextual analysis [14]. He has demonstrated an illustration of ATM framework the withdraw money utilization case needs to add logging angle to the ATM use case as shows up in Fig.9.

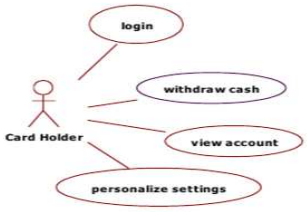

Fig. 9 (Atm Use Case)

When Draw this use case diagram of with cash draw with sequence diagram it show the interaction with all parts of aspects joint point , point cuts and aspects. It could identify the aspects and also can show aspect characteristics such as calling joint point and point cut (before, after and around)[5] .The aspect identification and showing properties of it in Fig.10.

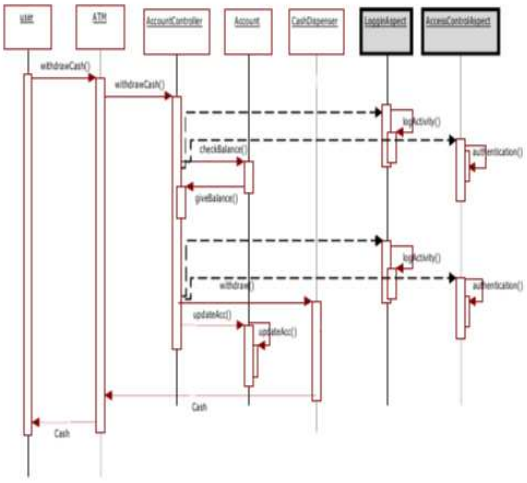

Fig. 10 (Sequence Diagram In AODL Notation Proposed Language)

4. POTENSTAIL ASPECTS IN DESIGN PATTERN

Berkane expressed that Aspect oriented giving to program supplements the Object oriented

Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

programming effective develops to handle structure and measured quality. This will help add to the best particularity of outline examples of these concerns. The examples comprise of two sections, section one recognized the aspects, classes, connections and operations identified with the arrangement. Second part is concerned with the quantity of implying practices and basic relations between segments. For example, in the connector outline design there are two classes, which cannot utilize the same interface that share parts, while an Aspect oriented, model will permit that by developing the interface of the Adoptee as is indicated Fig. 11 [15].

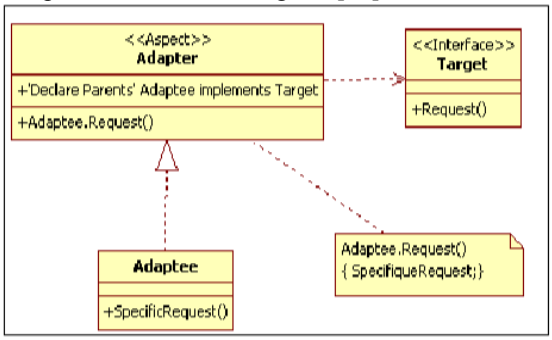

Fig. 11 (Adapter Pattern In Aspect Class Diagram)

5. OUR NEW METHOD TO IDENTIFY ASPECT CASE STUDY:

In this paper we try to identify Aspect in early development stage. We will use a case study of ATM machine the withdraw money part in Portuguese toll collection system. our approaches is use view point, use case and scenario-based and theme approaches to identify aspects in requirements stages so the main focus is that in all these approaches we could find common aspects for each approaches in System Requirement Specification SRS and then could be assure that is the potential aspect to impalement as it is shown in fig12.

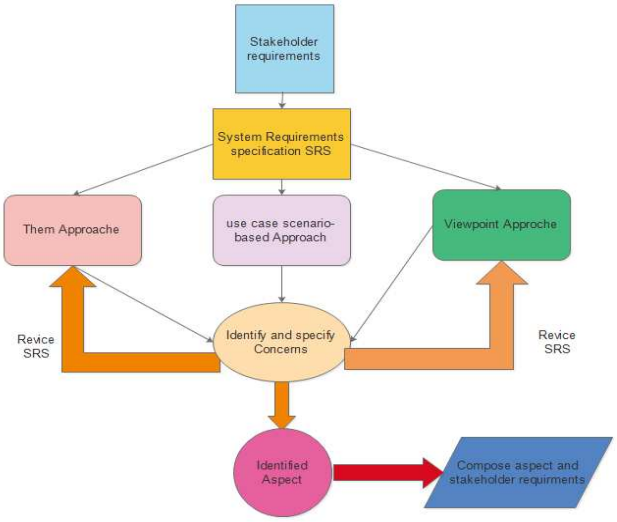

Figure 12 Proposed Approaches To Identify Aspects

First step examine the viewpoint approach, viewpoint based methodologies give a genuinely better approach for speaking to and abstracting necessities. We may additionally call them speaking to necessities on the parts of stakeholder, which bodes well on the grounds that every part's point of view and use is not quite the same as that of others. Catching the right point of view can bring about prerequisite fulfillment and convenience. In [19], prerequisites have been exhibited in Preview-like viewpoint which make prerequisites in XML based documentations. Aspects have likewise been spoken to in the same XML documentation cross-cutting relating viewpoints. The accompanying illustration, taken from [19], presents perspective-based representation. It shows a concentrate of a Portuguese toll accumulation framework in which a gadget called gizmo is introduced in a car and is initiated to pay tolls as the car passes the toll entryway as it is appears in fig13.

```
<?xml version="1.0" ?>
 <Viewpoint name="ATM">
  <Requirement id="1">
    The ATM sends the customer's card
    number,account number and gizmo
    identifier to the system for activation
    and reactivation.
  <Requirement id="1.1">
    The ATM is notified if the activation
    or reactivation was successful
    <Requirement id="1.1.1">
      In case of unsuccessful activation
      or reactivation the ATM is notified
      of the reasons for failure.
    </Requirement>
  </Requirement>
 </Requirement>
 <?xml version="1.0" ?>
<Concern name="Compatibility">
</Viewpoint>
```

```
<?xml version="1.0" ?>
<Concern name="Compatibility">
  <Requirement id="1">
    The system must be compatible
    with systems used to:
    <Requirement id="1.1">
      activate and reactivate gizmos;
    </Requirement>
    <Requirement id="1.2">
      deal with infraction incidents;
    </Requirement>
    <Requirement id="1.3">
      charge for usage.
    </Requirement>
  </Requirement>
</Concern>
```

Figure 13 Viewpoint Approaches Identification Of Aspect

Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The result of the viewpoint approached have shown the potential aspect could be found in the ATM machine part as it is shows in table1 [19] below:

Potetial Aspects	System Part
Security	Login Aspect
Performance	Responce Time
User managements	Admin could manage Multi User transaction to the System

Table 1 View Points Approaches Part Of ATM Part Of The System

Second approaches is use case scenario based approaches,

Ivor Jacobson in [20] states that execution of utilization cases crosscut the arrangement of segments and segment based systems neglect to accomplish utilization case measured quality. A bit of code of a segment may contain code of different utilization cases which will bring about code tangling issue and correspondingly on the off chance that we execute an utilization case, an arrangement of segments will constitute its usage which is a crosscutting property.

there are a few Aspects which are considered of course with each application, for example, execution, security, and adaptation to internal failure. These Aspects can be brought up in the Framework Requirements Specification (SRS). There are too application-related Aspects, for example, security Aspects for security basic frameworks, adaptation to internal failure Aspects for frameworks which should be running constantly, and synchronization Aspects for frameworks containing different running strings. We can distinguish these Aspects as competitor Perspectives from necessities of the framework amid the necessities designing stage. When we have recorded some of the hopeful Aspects we can call attention to those utilization cases which may have cooperation with these cross-cutting concerns (Aspects).

For instance in an ATM framework indicated in Figure 14, we can call attention to that the utilization case "withdraw money" will oblige contribution of a "logging" Aspect, adding to that for different clients as same time need "management" aspect for controlling clients activities . Therefore, we can highlight this utilization case to demonstrate that its further outline and usage will be influenced by the cross-cutting concerns, and it ought to be taken care of diversely contrasted with other utilization cases.

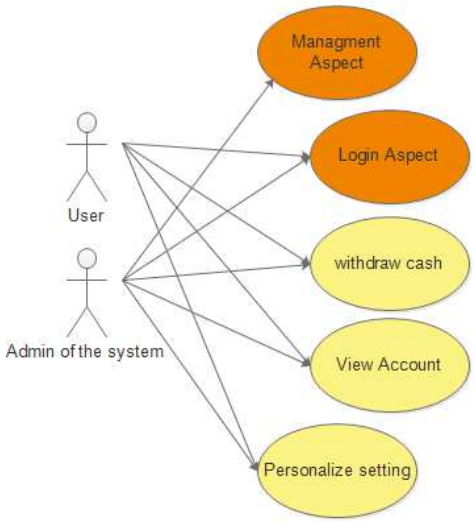

Figure 14 Use Case Diagram For ATM System

to distinguish aspects in an arrangement of necessities and how to model them in UML style outlines. The procedure we present here is the Theme way to deal with examination and configuration. we use is a hybrid of the symmetrical and asymmetrical paradigms. The word subject ought not be viewed as an equivalent word for aspect. Theme are more broad than aspects and all the more nearly include concerns as depicted above for the symmetric methodology. We see every bit of usefulness, aspect, or concern a designer may have as a different topic to be pandered to in the framework [21].

In Theme approach we will go throw withdraw cash from ATM . as shown in table below :

Frature	Requirements
ATM Withdraw Menu	R1 User can login to the system . R2 User Can withdraw money from the ATM . R3 :Admin user can manage all system request and maintance .
View Account	R4: user can check balance and genral informations.
Personloze setting	R5:User can modify personal information , password .

Table 2 Theme Approch For ATM Case Study

So to identify the potential aspect we have found that two requirement that all the system component will depend on such as cross cutting. R1 potential

Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

login aspect, R3 management potential Aspect as it is shown in Figure 15 below:

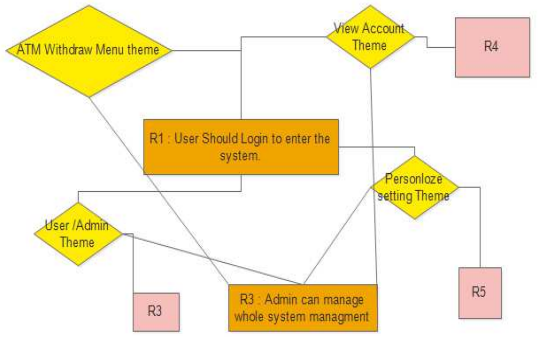

Figure 15 Cross Cutting Concern Potential Aspects

6. RESULTS

This research tries to investigate and find the best approach to identifying Aspects in the early stage of software design. Therefore the results shows that in these three approaches find potential aspects for ATM case study hve common Aspects and all agreed on Login Aspect and Managements Aspect as it is shown in table3 below :

	View Point based App	Use Case scenario approach	Them Approach
Login Aspect	Yes	Yes	Yes
User Managment Aspect	Yes	Yes	Yes

Table 3 Aspect In Three Different Approaches For ATM System

The thought is to make a worldwide principle or regulation to make it methodical over all aspects arranged segments and programming outline. Case in point, in requirement stage either from the stockholder or business investigation. These prerequisites are utilitarian necessities, for example, the activities of the business needs. Be that as it may, there are other non-Functional necessities, for example, logging, security, execution and exchange administration which ought to be considered amid the advancement stage. AOP can execute these non-practical acquirements independently and can compass over the whole plan of action. This makes it simpler to change or keep up this part later in the framework life-cycle [17]. Another approach to identifying aspects is to define stakeholder concerns, refine the stakeholder related concerns, define cross cutting concern, separate cross cutting concern and finally weave these cross cutting concern across the system [18]. There are also several other approaches that we have mentioned in state of art section, they also show that it is possible to identify cross cutting in UML design diagrams in the design stage. However, there is not a unique approach to identify ,where Aspects should be or when they should be triggered.

7. CONCLUSION AND FUTURE WORK

In this paper, we displayed starting points of examinations concerning distinguishing Aspects in programming design stage. The exploration demonstrates that there are numerous ways to deal with recognizing Aspect in Software development stages, however not all methodologies are connected in all cases or have particular principles or gauges that can without much of a stretch discovered cross cutting concern in any framework effectively , we have conclude that from each approaches we can identify part of Aspect then we can combine all in one framework . In future work we will exmine our approach to several case study and implement them using AspectJ then try to evaluate our approach by finding the different from the traditional Object Oriented implementation. .

8. ACKNOWLEDGMENT

We thank University of Human Development Staff and Huddersfield University. Thanks to Dr. Gary for his support.

REFRENCES:

[1] Kiczales, Gregor, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William Griswold. “Getting Started with AspectJ.” Communications of the ACM 44, no. 10 (2001): 59–65.

[2] Li, Hui, Mingji Zhou, GuiJun Xu, and Lingling Si. “Aspect-Oriented Programming for MVC Framework.” In Biomedical Engineering and Computer Science (ICBECS), 2010 International Conference on, 1–4, 2010.

[3] Gradecki, Joseph D, and Nicholas Lesiecki. Mastering AspectJ: Aspect-Oriented Programming in Java. John Wiley \& Sons, 2003.

[4] Iqbal, Saqib, and Gary Allen. “Designing Aspects with AODL.” International Journal of Software Engineering 4, no. 2 (2011): 3–18.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[5] Sant'Anna, Cláudio, Alessandro Garcia, Uirá Kulesza, Carlos Lucena, and Arndt von Staa. "Design Patterns as Aspects: A Quantitative Assessment." Journal of the Brazilian Computer Society 10, no. 2 (2004): 42–55.

[6] Kiczales, Gregor, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. Springer, 1997.

[7] France, Robert, Indrakshi Ray, Geri Georg, and Sudipto Ghosh. "Aspect-Oriented Approach to Early Design Modelling." IEE Proceedings-Software 151, no. 4 (2004): 173–85.

[8] Pawlak, Renaud, Lionel Seinturier, Jean-Philippe Retaillé, and Houman Younessi. Foundations of AOP for J2EE Development. Springer, 2005.

[9] Kiczales, Gregor, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G Griswold. "An Overview of AspectJ." In ECOOP 2001—Object-Oriented Programming, 327–54. Springer, 2001.

[10] Jacobson, Ivar. "Use Cases and Aspects-Working Seamlessly Together." Journal of Object Technology 2, no. 4 (2003): 7–28.

[11] Iqbal, Saqib. "Aspects and Objects: A Unified Software Design Framework," 2013.

[12] Stein, Dominik, Stefan Hanenberg, and Rainer Unland. "A UML-Based Aspect-Oriented Design Notation for AspectJ." In Proceedings of the 1st International Conference on Aspect-Oriented Software Development, 106–12, 2002.

[13] Khan, Shaukat Ali, and Aamer Nadeem. "UML Extensions for Modeling of Aspect Oriented Software: A Survey." In Proceedings of the 2010 National Software Engineering Conference, 5, 2010.

[14] Iqbal, Saqib, and Gary Allen. "On Identifying and Representing Aspects." In Software Engineering Research and Practice, 497–501, 2009.

[15] Berkane, ML, M Boufaida, and L Seinturier. "Reasoning about Design Patterns with an Aspect-Oriented Approach." In Information Technology and E-Services (ICITeS), 2012 International Conference on, 1–7, 2012.

[16] Bernardi, Mario L, and Giuseppe A Di Lucca. "Improving Design Patterns Modularity Using Aspect Orientation." STEP 2005, 2005, 209.

[17] Sirbi, Kotrappa, and Prakash Jayanth Kulkarni. "Stronger Enforcement of Security Using Aop and Spring Aop." arXiv Preprint arXiv:1006.4550, 2010.

[18] Rashid, Awais. "Aspect-Oriented Requirements Engineering: An Introduction." In International Requirements Engineering, 2008. RE'08. 16th IEEE, 306–9, 2008.

[19] Rashid, Awais, Ana Moreira, and Jo\=ao Araújo. "Modularisation and Composition of Aspectual Requirements." In Proceedings of the 2nd International Conference on Aspect-Oriented Software Development, 11–20, 2003.

[20] Jacobson, Ivar. "Use Cases and Aspects-Working Seamlessly Together." Journal of Object Technology 2, no. 4 (2003): 7–28.

[21] Baniassad, Elisa, and Siobhan Clarke. "Theme: An Approach for Aspect-Oriented Analysis and Design." In Proceedings of the 26th International Conference on Software Engineering, 158–67, 2004.