International Journal of scientific research and management (IJSRM)
\||Volume||3||Issue||5||Pages|| 2903-2909||2015||
Website: www.ijsrm.in ISSN (e): 2321-3418

Managing Inconsistencies In Software Design Model

[1]Sadia Sahar, [2]Tasleem Mustafa, [3]Farnaz Usman, [4]Farnaz Usman, [5]Aasma Khalid, [6]Sidra Hafiz, [7]Nadia Aslam

[1,2,3,4,5,6,7]Dept. of Computer Science
University of Agriculture
Faisalabad,
Pakistan
sadiasahar@ymail.com
tasleemustafa@hotmail.com
Shusman123go@yahoo.com
Aasmakhalid89@yahoo.com
pinkdreamflower@gmail.com
Nadiaaslam_7@yahoo.com

***Abstract*—Design is becoming the important phase of software development because of the demand of quality and complexity of software. One of the greatest challenges to the software engineers and researchers is to identify and resolve the inconsistencies; they meet during a system's design. If inconsistencies are detected earlier the design phase, mostly problems that might occur during the last stage of software development, can be avoided. Inconsistency is the violation of the rules of software development, the rules must be clearly defined so that one can easily detect the inconsistency and adopt a better solution to resolve it. This research gives a four step inconsistency resolving strategy i.e., to know what the model is? ; To give proper consistency rules in the model?; to explore how the inconsistency happens? and to proposed an effective solution to remove these inconsistencies. The first and second steps include formation of consistency rules using. The rules are applied in the design phase and then classified different violation of these consistency rules by knowing how they can happen. A frame based Inconsistency Management System (FIMS) has been developed that analyse different inconsistencies in the class and the sequence diagram of UML. It also gives a proper solution to handle these inconsistencies.**

Keywords-component; software design; consistency; inconsistency; UML

I. INTRODUCTION

Software system is gaining hold on greater part of our socio-economical activities. It is enhancing the functioning of all aspects of our society; May it be a grocery store, building designing architecture, vehicle manufacturing, communication technology, online commerce and banking, local administration and defence even in all aspects of society. The software systems become critical to all of them. Day to day increasing demands for software system to ease working of every field of life is giving growth to growth to many functional requirements as shortage of time to market the stuff, friendly user interface, seamless operating environment and so on. This is leading to increase the complexity of software development (Olsan and Grundy, 2001). To master this complexity software engineer build and employed well defined development process such as waterfall model and spiral model (Boehm, 1988). The developers tackle software problems in four main phases. These are typically identified as Requirement elicitation and analysis, Architectural and detail design, Implementation and Testing (Ibrahim *et al.*, 2011). But in development process a large number of different descriptions are used by software engineers. These descriptions include different analysis modes, design, specification, program codes, user guides, style guides, test plans, change requests, schedules and process models (Jackson, 1995).

Design is becoming the important phase of software development process due to demand of quality and complexity of software (Zaretska et al, 2012). The industries are trying to search out the techniques which automatically produce software and can also improve quality with reduction of cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. The increase in demands of software and its quality lead the complexity of software further enhance the scope of project and hence the time. Organizations also inquire about techniques to manage the complexity of systems. They also need to solve physical allocation, concurrency, duplication, safety problem, load balancing and fault tolerance. The Unified Modelling Language (UML) is the best design based solution to counter these needs (Larman, 2004).

UML is de facto standard for modelling languages used for software design (Zaretska *et al.*, 2012). Most of the software design consists of thousands UML models so a greater chance of inconsistency between software design is present. As Inconsistency is the violation of rules or relationship that must be obeyed by different description in software development process (Nuseibeh *et al.*, 2001), the rules must be clearly defined so that one can easily detect the inconsistencies in his software design and can propose a better solution to resolve them. In this paper emphasis is made on the types of inconsistency and the consistency rules.

II. RELATED WORK

(Nuseibeh *et al*., 2001) also proposed a model for inconsistency management. He just suggested the model that is analyse inconsistency, classify them, detect them and then resolve these detected inconsistencies.

(Liu *et al.,2002*)proposed a rule based expert system that detects inconsistency between UML diagrams. In a rule-based expert system, the inference engine links the rules contained in the knowledge base with data given in the database. When the goal is set up, the inference engine searches the knowledge base to find a rule that has the goal in its consequent. If such a rule is found and its IF part matches data in the database, the rule is fired and the specified object, the goal, obtains its value. If no rules are found that can derive a value for the goal, the system queries the user to supply that value. Rules are overlap so difficult to add or remove rules in rule based schema. If two or more rules are active at the same time then it is difficult to decide which one to execute next? In FIMS Frames represent here a major source of knowledge, and both methods and demons are used to add actions to the frames.

(Kotb and Katayama, 2005) worked on inconsistency detection between UML diagrams by using XML language. This approach does not give the way to represent information about UML model in XML. It also does not describe how to check the xml file. FIMS covers both of these deficiencies.

(Alanazi and Gustafson, 2008) also made a research on inconsistency management in UML diagram. His work is based on just state diagram while the proposed work analyse the inconsistencies between all two UML diagrams class and sequence diagrams and also gives a list of common insistencies that can be present in all diagrams of UML language.

(Ibrahim *et al*. 2011) also analysed inconsistencies between UML diagrams. He used activity diagram and Use Case diagram for this purpose. He used Object constraint Language that to describe consistency rules that has less expressive power than Alloy language. Moreover He built three rules between these two diagrams while this work detects eleven consistency rules based on all two important diagrams i.e., class and sequence diagrams.

Another work is done by (Streaten and Brussel, 2011) on defining consistency rules between all these UML representations. But they just define the rules not suggested the solution to resolve them. In contrast this research not just describes different rules but also proposed the solution to manage inconsistencies.

(Peter and Zaidman, 2012) worked on the design inconsistencies that effects on code. He developed a tool to assess these inconsistencies but his tool just accepts java code.

III. MATERIALS AND METHODS

Most of the software design consists of thousands design model so a greater chance of inconsistency between software design is present. For a trustworthy software system it is very important to handle these inconsistencies present in different design model. So our goal is to build up an environment to handle inconsistencies in UML model that are for design phase of software development. I have proposed four steps to handle inconsistencies in any model used in design phase of software development. These steps are:

Step 1: Know what the model is?

Step 2: Know what are consistency rules in that model?

Step 3: Know how inconsistency happens?

Step 4: Proposed an effective solution to overcome these inconsistencies.

By following these steps we can find out inconsistencies present in any modelling language. I used Unified Modelling Language (UML) for this purpose.

3.1 Step 1: Know what the model is?

In the first we must know about the model in which we want to remove inconsistencies. We must know; what is that model; what are different principals to build up these models and what are different notations etc. Here I used UML model.

3.2 Step 2: Know what are consistency rules in that model?

We must know about the consistency rules that can be applied on the selected model. Some of consistencies rules present in UML model are as follows:

Use case diagram is not used in design phase of software development (Jacobson, 1994) so I did not include it into my research work. Each rule is defined in Alloy format.

Rule 1: Operations and attribute present in class must have data type (Liu *et al.*, 2002).

Rule 2: Abstract class used in class diagram must have at-least one concrete class that must have implementation of it (Streaten and Brussel, 2011).

Rule 3: Object diagram has same classes as mentioned in the class diagram

Rule 4: A method in a class cannot have more than one return attribute (Streaten and Brussel, 2011).

Rule 5: Class uses in sequence diagram must be present in class diagram (Liu, 2002).

Some other rules are defined by (Sahar *et al.*, 2014).

3.3 Step 3: Know how inconsistency happens?

When the above rules are violated it will be an inconsistency. Inconsistency can be intra-inconsistency, within a single UML model, or ultra-inconsistency, between two or more UML models. Inconsistencies can be classified as

1. Dualistic Inconsistency
2. Type Inconsistency
3. Navigational Inconsistency
4. Relational Inconsistency
5. Domain inconsistency
6. Missing object

Detail of all these types is explained by (Sahar *et al.*, 2014).

4: Proposed an effective solution to overcome these inconsistencies

Nuseibeh proposed a framework for inconsistency management which is widely used for handling inconsistencies. Main activities of this framework are as follows:

1. Measure the inconsistencies by locating, identifying and classifying them.
2. Analyze the impact of inconsistencies on the software process
3. Resolution of detecting inconsistencies
4. Monitor the consequence of handling action

All researches are based on these three steps involved in model consistency management (Ibrahim, 2011). Some are for measuring inconsistencies; some are for analysing the impact of those inconsistencies. I have proposed a model that is based on detecting and resolving inconsistencies at real time.

3.4.1 FIMS: Frame-based Inconsistency Management System

For detecting and resolving inconsistencies I proposed a Frame based model called FIMS. This model contains four parts i.e. UML Editor, FBE, FIMS detector and FIMS Resolver.

Following figure explains the whole scenario of the proposed model. At first step the user draws UML models in UML editor. The models enter the Frame based environment where they are separated into different frames. These frames then pass through FIMS detector that detects inconsistencies by using information provided by frames. These detected inconsistencies are resolved by the last part of the proposed model, i.e., FIMS Resolver.

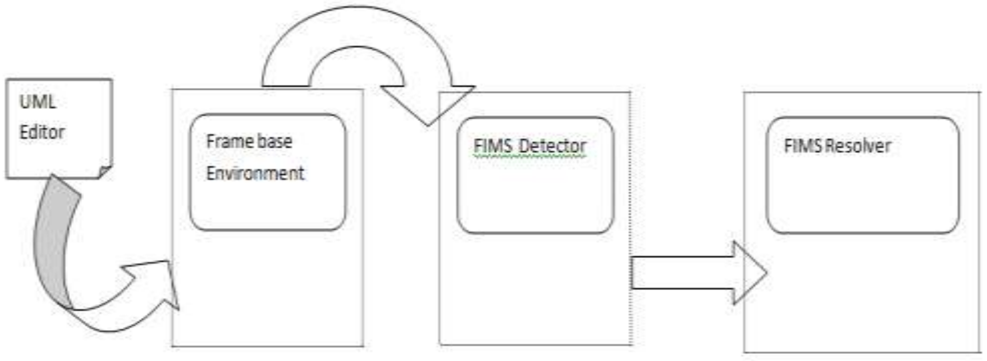


Fig 3.1: Frame based Inconsistency management system

3.4.1.1 UML Editor

UML Editor is the first part of FIMS. The user can communicate through this part to the whole system. It provides the interface for user to draw UML diagrams.

3.4.1.2 FBE (Frame Based Environment):

The second part of the FIMS model is FBE (Frame Based Environment). When a model is drawn in the FIMS it goes to FBE. Basic job of this component is to process UML model and keep information about different models in different frames. Each frame can be presented by a table, class, xml or any other format. For example when a user enters a class diagram in FIMS, the FBE puts class information in one place and information about their association in other place.

This information is now sent to the second part of FIMS that is FIMS Detector. This part is based on different conditions to detect different type of inconsistencies.

3.4.1.2 FIMS Detector:

Here I defined some daemons (patterns or conditions) to detect inconsistencies. These conditions have been written as:

IF Condition THEN show message

"Condition" is actually the pattern which is used to detect inconsistencies and the "show message" shows inconsistencies present in the model. If the pattern gets matched or condition becomes true then an inconsistency is present in the observed model and an appropriate message is shown to inform the user about the inconsistency.

Example 1: if two classes have the same name in a class diagram, it can be detected as

```
    If class1.name =class2.name
   Then
        Show inconsistency present in particular model
    End If
```

In this rule name inconsistency is detected. When the name of two different classes is matched then it will be an inconsistency.

Example 2: In this example FIMS detects missing object inconsistency in sequence diagram as:

```
If Seq_class.name present in class frame
        Then    no inconsistency
Else missing object inconsistency
```

3.4.1.3 FIMS Resolver:

The last phase of the framework is to resolve inconsistencies by using two methods

1. When changed method
2. When needed method

"When changed" method is applied when the relevant value of slot is changed and "When needed" method is used when a particular information is needed

Example:

In sequence diagram, if user draws a message directed from class1 to class2 the "when needed" method is applied to populate combo-box with methods of class1.

3.5 Implementation

For implementing this model I developed a tool named "FIMS" by using CSharp.net. It covers all of the four parts of the proposed model i.e.,

1. Provide UML Editor
2. FBE
3. Inconsistency Detector
4. Inconsistency Resolver

In UML Editor User can draw two types of diagrams that is class and sequence diagrams. XML language is used in frame based environment to keep information about UML models in different frames. Inconsistency detectors and resolver are developed in CSharp.net framework. FIMS can detect different inconsistencies in class and sequence diagrams.

IV. RESULTS

The objective of my work is the detection and resolution of inconsistencies in different UML design models. In this I have suggested four steps to overcome inconsistencies in design models. This thesis explores what actually the model is. It thoroughly describes each step to build up different models in Unified Modelling Language (UML).

The work also explains almost all consistency rules that must be hold in different UML representations. Alloy language is used to express these consistency rules. It is easy to read and understandable for both experts and non-experts. It also has very strong expressive power to explain different situation.

Proposed solution also presents different classification of inconsistencies. Inconsistencies also categorized according to their nature. I categorized these inconsistencies into six different types that are dualistic, type, missing object, rapport, navigational and domain with examples.

Frame Based Inconsistency management system (FIMS) deals with different types of inconsistencies. According to this technique each UML diagram will be converted into different frames. Different daemons and methods are applied on these frames to detect and resolve different inconsistencies present in UML diagrams.

To implement the framework, I have developed a tool in C#.Net. This tool covers all of the four parts of proposed model discussed in chapter number three. These parts are as follows:

1. UML Editor
2. FBE
3. FIMS Detector
4. FIMS Resolver

4.1.1 UML Editor

This provides an interface for real time drawing of UML diagrams. It covers two types of diagrams i.e., class diagram and sequence diagram. It provides a main page by which user can draw diagrams very easily. It offers very friendly user interface. To draw class diagram user must select class diagram option showing in left panel of main page or sequence diagram option for drawing sequence diagram. The following is the picture of the page where a user can start.

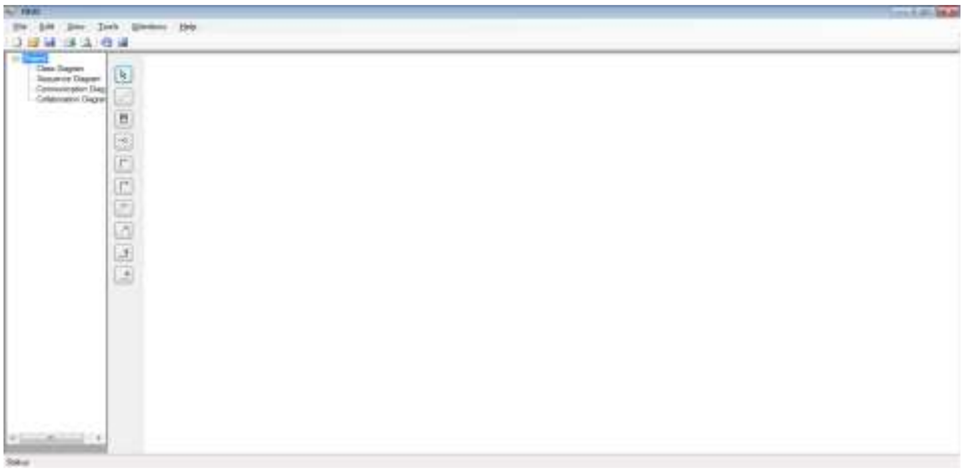


Fig 4.1: Main Form of FIMS Editor

4.1.2 FBE (Frame-Based Environment)

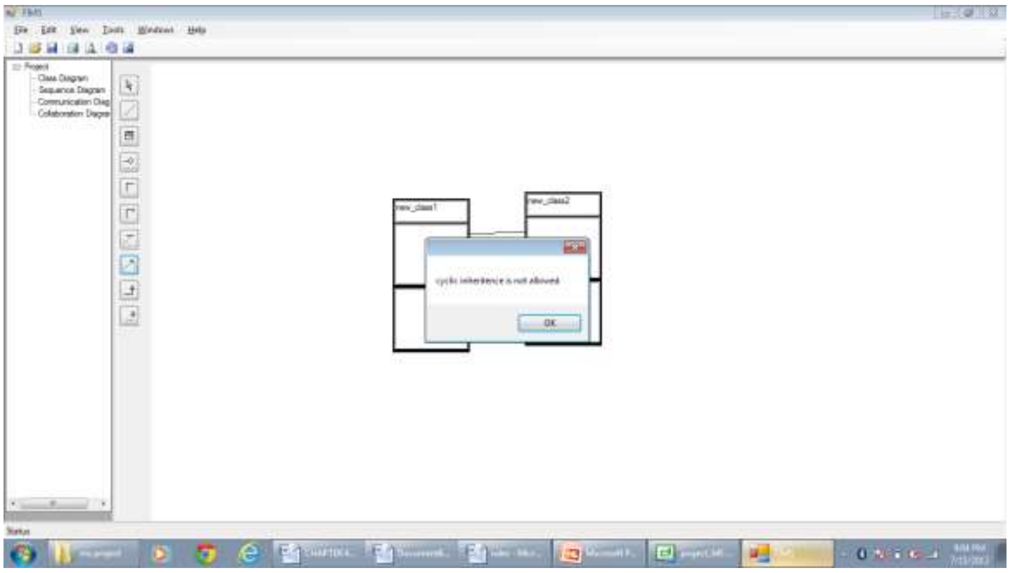
Basic job of this component is to process UML model and keep information about different models in the form of frames. For frame based environment I used extensible mark-up language (XML). All diagrams are converted into XML files. Each frames keeps different information about different UML diagrams. For example if one frame has information about classes, their methods and attribute then the other frame has information about association between classes present in class diagram. For example if user draw a class it create a frame like

```
<Class>
<name>  class_name</name>
<attribute 1>attribute_value</attribute1>
:            :                    :
<attribute n>attribute_value</attribute n>
<operation 1> operation _name</ operation 1>
:      :              :
< operation n> operation _name</ operation n>
</class>
```

4.1.3 FIMS Detector

This part of the tool detects inconsistencies by applying demons and methods on the frames created by FBE. Demons used **IF-THEN** structure. For example if we want to check cyclic inheritance between class1 and class2 in the class diagram Demon can be

"IF class1 is descendent of class2 THEN show cyclic inheritance inconsistency"

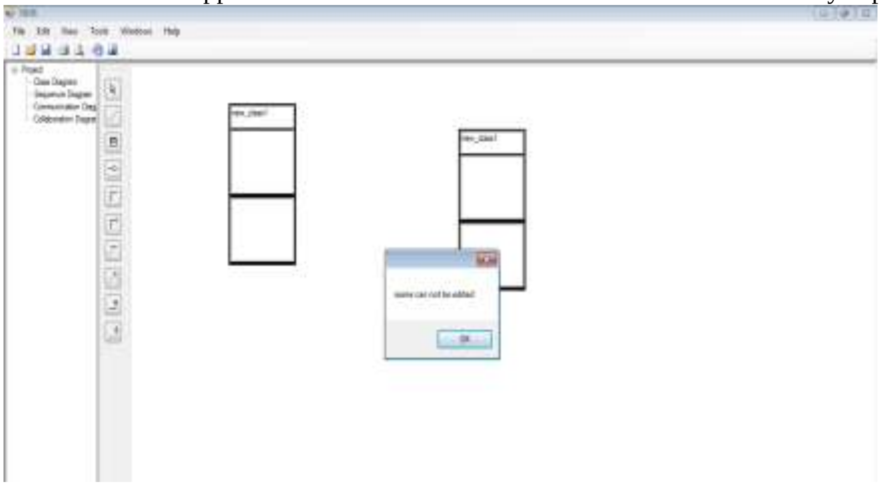


cyclic inheritance is not allowed

Fig 4.2: Detecting Cyclic Inheritance

If user gives same name to different classes in a class diagram then following condition will be applied

"IF class name is present in class frame THEN name inconsistency is present"



name can not be added

Fig 4.3: inconsistency detection

In the above diagram user gave same name to the two different classes but the tools did not give the permission to do this. This part is capable of detecting eleven inconsistencies between class and sequence diagram. These inconsistencies are as follows:

- ✓ Operations must have return type.
- ✓ Two classes in a class diagram cannot hold same name
- ✓ Parameter of method in a class must has a type
- ✓ attribute present in class must have data type
- ✓ Class uses in sequence diagram must be present in class diagram
- ✓ Operation used as message must be present in relevant class diagram
- ✓ Two attributes in a class diagram cannot hold same name
- ✓ Two operations in a class diagram cannot hold same name
- ✓ Parameter of operation in sequence diagram must be matched with operation used in class diagram
- ✓ Navigation direction between classes must be correct
- ✓ Cyclic inheritance in class diagram is not allowed

4.1.3 FIMS Resolver

FIMS Editor also resolves some inconsistencies that can be present in UML models. Two methods are used for this purpose

- ◦ When changed method is applied when the relevant value of slot is changed
- ◦ When needed method is used when a particular information is needed

For example in sequence diagram, if user draw a message directed from class1 to class2 then "WHEN NEEDED" method is applied to populate combo-box with methods of class1

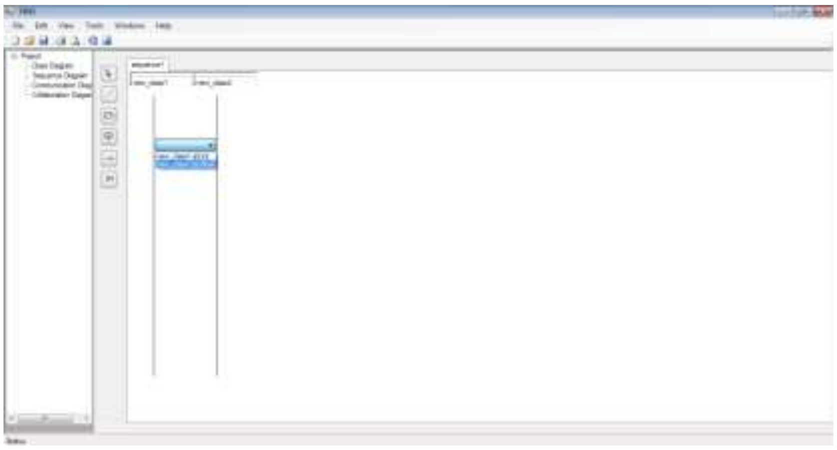


Fig 4.4: resolving inconsistency in sequence diagram

Diagram 4.5shows that a user wants to create cyclic inheritance between two classes but the tools did allow to do so.

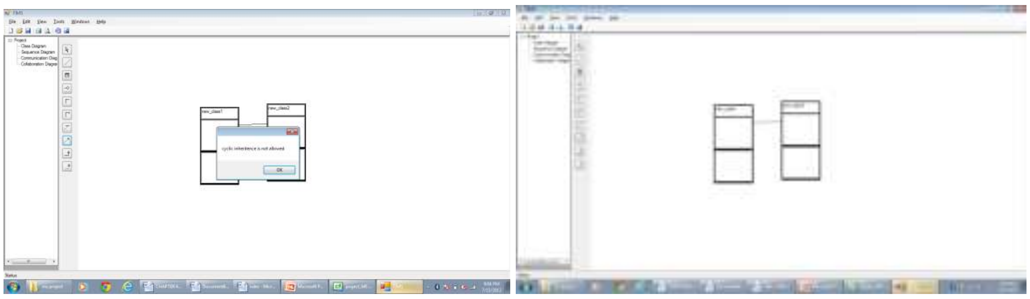


Fig 4.5: resolving inconsistency in class diagram

V. CONCLUSION

Design phase of software development contains different models ranges from twenty to thousand models. When these models integrate they become inconsistent. The goal of this research was to make a framework that not only detect inconsistencies in all UML diagrams but also resolve them. After conducting the whole study it is concluded that

- ✓ It is four step solution to overcome inconsistencies
- ✓ Describes whole model step by step with its expression.
- ✓ It covers two UML diagrams
- ✓ Explains more than seventy consistency rule of all twelve diagram of Unified Modeling Language.
- ✓ Classifies all inconsistencies into six categories
- ✓ It not only detects inconsistencies but also resolves them
- ✓ In FIMS rule are fired according to frame.
- ✓ In resolver appropriate action is fired according to classification of inconsistency.
- ✓ It is very flexible as new rules and scenarios can be added according to need.

Limitations and Future Work

Nothing is perfect in the world. Everything has some faults or some space to improve it. Hence the proposed research also contains some flaw in it which can be further removed in the future.

- More rules will be applied
- More diagrams will be covered
- To enhance performance
- Add more functionalities
- Design phase contain design patterns which are not included at present can be added in future
- Other phases rather design phase can be added to remove inconsistencies

REFERENCES

Alanazi M.N. and D.A. Gustafson. 2008. Inconsistency Discovery in Multiple State Diagrams. *World Academy of Science. Engineering and Technology*; 28(1) : 54-62

Boehm. B. W. 1988. A Spiral Model of Software Development and Enhancement. *IEEE Computer* ; 21(5): 61-72

Ibrahim N., R. Ibrahim, M.Z. Saringat, D. Mansor and T. Herawan. 2011. Consistency Rules between UML Use Case and Activity Diagrams Using Logical Approach. *International Journal of Software Engineering and Its Applications*; 5(3) : 119-134

Jackson M., 1995. Software Requirements & Specifications: a lexicon of practice, principles and prejudices, *Addison-Wesley publication Co*; pp:15

Kotb Y., T.Katayama. 2005. Consistency Checking of UML Model Diagrams Using the X.ML Semantics Approach. *Proceeding WWW '05 Special interest tracks and posters of the 14th international conference on World Wide Web*; pp: 982-983

Larman C. 2004. Applying UML and Patterns 3rd edition. *Prentice Hall Press* ; pp: 123-579

Liu W., S. Easterbrook and J. Mylopoulos. 2002. Rule-Based Detection Of Inconsistency In Uml Models. *Proceeding UML Workshop on Consistency Problems in UML-based Software Development* ; pp: 106-123

Nuseibeh B., S.Easterbrook and A.Russo. 2001. Making Inconsistency respectable in Software

Development. *The journal of Systems and Software*; 58(2): 171-180.

Olsan T. and J. Grundy. 2002. Supporting Traceability and Inconsistency Management between Software Artifacts. *Proceeding conference on (374) Software Engineering and Application* ; pp: 133-135

Peter R. and A. Zaidman. 2012. Evaluating the Lifespan of Code Smells using Software Repository Mining. *Proceedings of the 16th European Conference on Software Maintenance and Reengineering published in IEEE Computer Society* ; pp:411-416

Zaretska I., O. Kulankhina, H. Mykhailenko and R. Kovalenko. 2012. Checking Inconsistencies in UML Design. *Proceeding ICTERI: International Conference on ICT in Education, Research, and Industrial Applications* ; pp: 33-43

Sahar S., T. Mustafa, F. Usman, A. Khalid, N. Aslam and S.Hafeez. 2014. Description of Consistency Rules in Software Design Models. *Journal of Emerging Trends in Computing and Information Sciences* ; 5(5): 428-432

Straeten R. V. D. and V. U. Brussel. 2011. Description of UML Model Inconsistencies. *Vrije Universiteit Brussel, Department of Computer Science, SOFT-TR*; pp: 1-14