Volume 3 | Issue 3 Article 5

January 2014

DYNAMIC DETECTION OF DESIGN INCONSISTENCY DURING SOFTWARE DEVELOPMENT USING DAID APPROACH

SUMA. V
Dayananda Sagar College of Engineering, Bangalore, India, sumavdsce@gmail.com

ARFA BAIG
Department of Information Science and Engineering, Dayananda Sagar Institutions, Bangalore, India, arfa.61@gmail.com

DIVYASHREE B J
Department of Information Science and Engineering, Dayananda Sagar Institutions, Bangalore, India, divyabj08@gmail.com

N B SONALI
Department of Information Science and Engineering, Dayananda Sagar Institutions, Bangalore, India, sonalinayakank@gmail.com

S AKSHAYA
Department of Information Science and Engineering, Dayananda Sagar Institutions, Bangalore, India, akshayashankar@gmail.com

Recommended Citation
V, SUMA.; BAIG, ARFA; B J, DIVYASHREE; SONALI, N B; and AKSHAYA, S (2014) "DYNAMIC DETECTION OF DESIGN INCONSISTENCY DURING SOFTWARE DEVELOPMENT USING DAID APPROACH," *International Journal of Computer Science and Informatics*: Vol. 3 : Iss. 3 , Article 5.
DOI: 10.47893/IJCSI.2014.1140
Available at: https://www.interscience.in/ijcsi/vol3/iss3/5

DYNAMIC DETECTION OF DESIGN INCONSISTENCY DURING SOFTWARE DEVELOPMENT USING DAID APPROACH

SUMA V[1],ARFA BAIG[2],DIVYASHREE B J[3],N B SONALI[4] & S AKSHAYA[5]

[1,2,3,4,5]Department of Information Science and Engineering, Dayananda Sagar Institutions, Bangalore, India
Email:sumavdsce@gmail.com
arfa.61@gmail.com,divyabj08@gmail.com,sonalinayakank@gmail.com,akshayashankar@gmail.com

Abstract-Evolution of software has lead to the fast growth of technology whose impact can be witnessed in all the domains of scientific and engineering applications. Hence engineering high quality software is one of the core challenges of all IT industries. The software models which are being used for the development of the software products may lead to inconsistencies. Nevertheless, theexistence of several methodologies during the development process in order to overcome inconsistencies operates at static mode leading towards expensive nature of rework on those inconsistencies. Therefore, this paper presents a dynamic model which resolves the aforementioned issue by capturing inconsistencies dynamically in an automated mode using Dynamic automated inconsistency detection (DAID) model. The implementation results of DAID capture the design inconsistencies dynamically at the time of their injection points in lieu of inconsistency detection during validation testing. This approach of dynamic design inconsistency detection reduces cost, time and its associated overheads. Further implementation of DAID in an automated mode increases productivity, quality and sustainability in IT industries.

***Keywords**: Software Quality; Consistency Checking; Software Development Life Cycle; Design Techniques, Analysis of inconsistencies.*

I. INTRODUCTION

For the persistence of software industry, total customer satisfaction is necessary. Hence emergence of high quality software is necessary. Attributes of high quality software includes development of software product within the scheduled budget, resources,time and more importantly to be flaw free. Hence it is rudimentary for any software generating organization to implement highly structured development process. Since quality does not emerge as an end factor but is realized as a continual process, the entire software development life cycle comprising of requirements analysis phase, design phase, incrimination phase and testing should be individually and cumulativelydefect free. Further, most of the ITindustriesuse objects oriented approach to implement their applications. It is worth to note that designing software models enables one to convert the theoretical requirement specifications into implementable code. However, from our adherent interpretation on several projects across several software industries indicate that the design phase is more error prone due to the existence of conventional mode of designing. Currently, there exists several design models to detect design. The current approach of design models uses batch consistency and type based approaches which detects the inconsistencies in the design model. However, these approaches have a major limitation as it is time consuming and use manual annotations .It further leads to accelerated cost with diminished quality and productivity which is a foremost risk for any software company. Hence, inconsistency detection in design models should be carried out to avoid unnecessary rework. However, it is seen that the problem of delayed detection of design inconsistency is due to lack of automated mode of detecting inconsistency. Further, they unearth these inconsistencies during validation testing rather than their detection dynamically at the time of their injection points. Hence the scope of this research is to reduce injection of design inconsistencies dynamically. This paper introduces the development of an automated design checking inconsistency checking technique usingDynamic automated inconsistency detection (DAID) model. The implementation results of DAID model is presented in this paper using a sample ATM case study. Section II of this paper provides background work for the above-stated problem. Section III elucidates the research design and Section IV provides the description of the developed model. SectionV provides implementation of DAID model and results .Section VI summarizes the entire work.

II. LITERATURE SURVEY

Several researches is progressive in detecting the inconsistencies in software models. Authors in [1], state that the consistency is performed using inter-view point rules to detect the inconsistencies [1].To make the consistency checking easier authors of [2] have used graph representation comprising of class and sequence diagram [2].However, author in [3] recommends UML design models tend to detect and repair inconsistencies and they further suggest the designer to be aware of design flaws inorder to overcome the design inconsistencies [3].Authors of [4] has proposed technique to tolerate and manage inconsistencies during software development[4].Authors in paper [5] provide various options and approaches for fixing inconsistencies [5].Author in [6] suggests use of various metamodels and consistency rules for fixing inconsistencies during development activity [6].However, author in [7] has introduced a new technique to automatically decide when to evaluate the consistency rules and works with black box consistency rules [7].Authors in [8] expresses that regression testing techniques to identify inconsistencies and other flaws existing in the software [8]. Authors in [9] state that major defects are seen at design phase for the applications developed in product based industries. Therefore, authors in [10] have proposed development of dynamic model to reduce design inconsistencies dynamically.

III. RESEARCH DESIGN

Consistency rules are conditions on a model, using which the model is validated. Instantaneous consistency checking thus requires a perceptive of how the model changes. This research proceeds with the identification of design problem in an automated manner by listing out the various inconsistencies related to every part or phase of the designed model. With the knowledge of existing works this research lead us to propose a dynamic and automated technique of detecting the design inconsistencies during software development process. Consequently, the next step was to contrast the existing techniques with respect to cost and time objectives that lead us to major differences stating the efficiency of our approach. From the data analysis, this investigation has directed to introduce an innovative technique through which design inconsistencies are detected in an automated manner without human intervention right at the time of its inception .This is achieved with the implementation of DAID (Dynamic and Automated Inconsistency Detection) model.

IV. DAID MODEL

The architecture of DAID model is shown in Figure1.This section provides an explanation on the operations of DAID model using a sample ATM example.

Working of DAID model starts with the generation of SRS from the designer followed by the design specifications. Traditionally, the inconsistencies detected aremodified in implementation phase which leads to rework. Our model perceives the inconsistencies in the design before the model enters the implementation phase in order to unearth design flaws.

Consider the case study which is an ATM application. DAID Model verifies card using the rule checker. In case an inconsistency is detected (Card is invalid), the rule checker reports the inconsistencies to the designer. The designer in turn, rectifies the model and reverts it back to the DAID model for rechecking. This rechecking mechanism is allowed to happen only for two cycles. This will help us to check the efficiency of the designer based on the design complexity as well as the time constraint. In cases where there are no inconsistencies (such as a card is valid) detected, the above mentioned process is not performed and the design directly enters code construction phase.

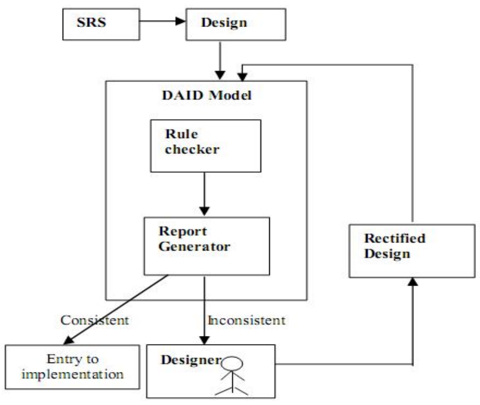

Figure1.DAID Model [10]

An ATM application has several classes andin this paper a sample class is depicted in figure 2. Accordingly, accounts class can be either of type savings account or current account.There can be inconsistencies observed during the design of this class. Table 1 depicts a sample list of design inconsistencies that may occur in the sampled ATM application.

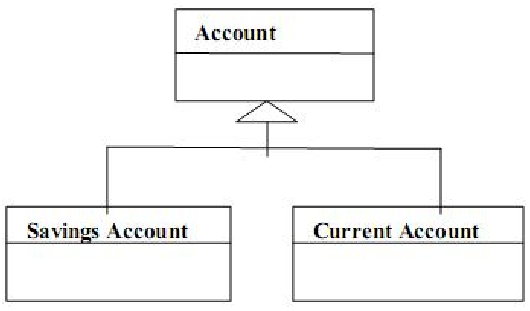

Figure2. Class Diagram

The ATM can't read the card.
The card has expired.
The ATM times out waiting for a response.
The amount is invalid.
The account is invalid.
The machine is out of cash or paper.
The communication lines are down.
Transaction is rejected of card duplication.

Table 1.Sample inconsistency list of ATM application

Figure 3.1, Figure 3.2, Figure 3.3, Figure 3.4 and Figure 3.5 illustrate the sequence diagrams for various sampled inconsistencies observed in ATM application.

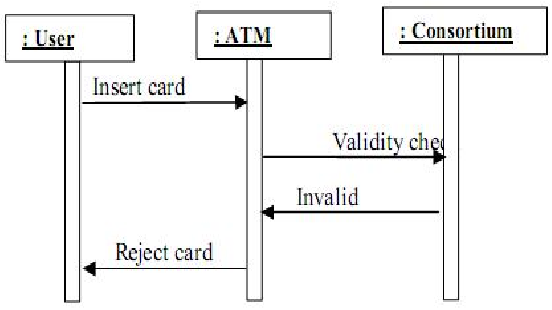

Figure3.1. ATM sequence diagram for inserting card

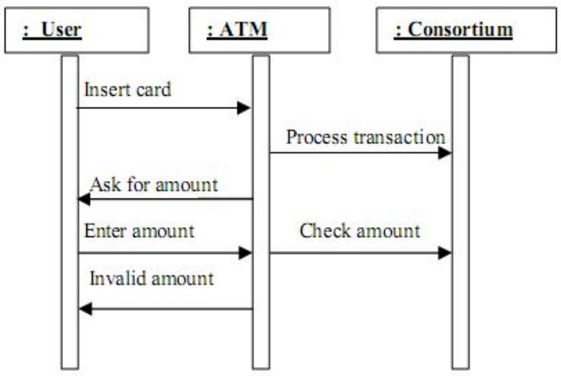

Figure3.2 ATM sequence diagram for checking amount

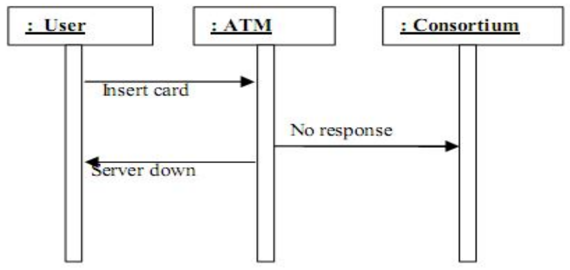

Figure 3.3 ATM sequence diagram for checking communication lines

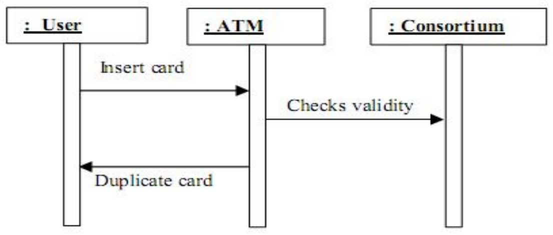

Figure 3.4 ATM Sequence diagram for card rejection due to duplication.

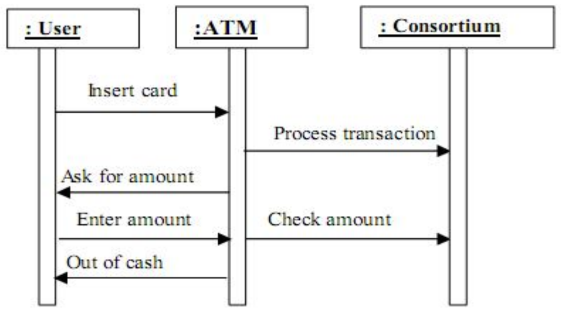

Figure 3.5 ATM Sequence diagram for out of cash

From the Figure 3.1, Figure 3.2, Figure 3.3, Figure 3.4 and Figure 3.5, we can infer that there can be inconsistencies such as rejection of card, invalid amount, out of cash and duplication of card which needs to be addressed at the time of their injection in order to reduce overheads during software development process.

V.RESULTS

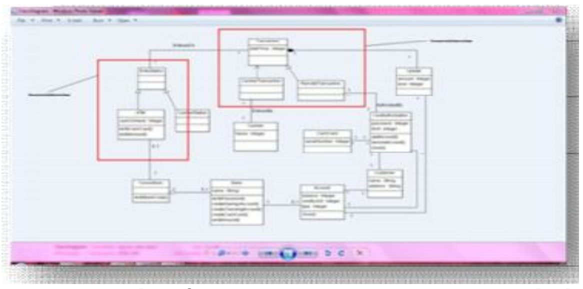

Figure 4.Snapshot of an ATM Transaction UML Diagram built in ArgoUML.

Consider the following UML Diagram of anATM application which is constructed in the ArgoUMLsoftware. ArgoUML allows us to construct any UML diagram which can be exported and given as input in SDMetrics.

The aim is to provide a high level comprehensive view of transformation of user requirements into consistent design specifications using a simplified UML model which is constructed in ArgoUML and it can be in the form of class diagram, sequence diagram and also use case diagram.
The completed design is then saved and in order to check for the inconsistencies, this file is exported with the help of XMI (XML metadata interchange) which enables easy interchange between ArgoUML and SDMetrics.
SDMetrics analyzes the structure of the UML models by making use of the design rule checking to automatically detect incomplete, incorrect, redundant or inconsistent design. It finds problems at the design stage, even before they are committed to source code.

Rule Checker:
The rules shown in the following table are cross checked by the rule checker against the UML diagram which is given as input.

Rule	Category	Description
AttrNameOvr	Naming	Class defines a property of the same name as an inherited attribute
CyclicInheritance	Inheritance	Class inherits from itself directly/indirectly
DepCycle	Style	Class has circular reference
DupAttrNames	Correctness	Class has two or more properties with identical names
Unused	Completeness	Class is not used anywhere
MultipleInheritance	Style	Class has more than one parent
Capitalized	Naming	Attribute names should start with lowercase letter
NotCapitalized	Naming	Class name should start with uppercase letter
Unnamed	Completeness	Class has no name
NaryAgg	Style	Association has three/more association ends

Table.2 Rules and their description

Report Generator
Once the model is analyzed with the help of the rules, the list of inconsistencies will be displayed by the report generator which checks whether the given model is consistent.
Once the rule checking for each model element in the UML diagram is completed, the metrics is calculated and is displayed as a table which is shown below in Figure 5:-

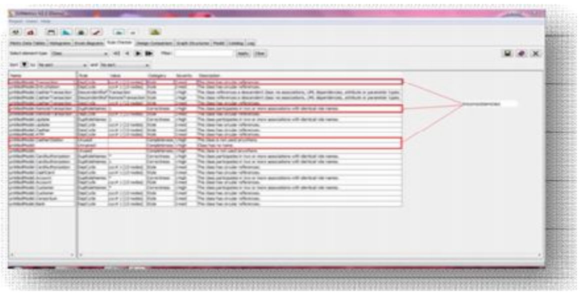

Figure 5. Snapshot of list of inconsistencies

Figure 6 and Figure 7 illustrate the cost and time analysis using DAID.

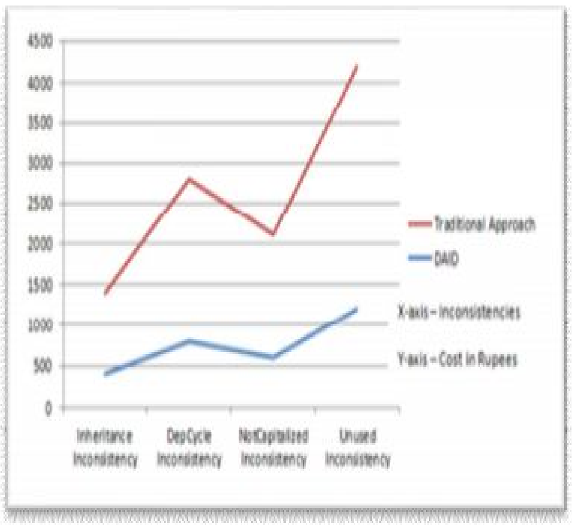

Figure 6.Cost comparison

From Figure 6, we can infer that an ATM application when developed using DAID, the cost required for rework of design inconsistencies is very low than when compared to cost required to fix design inconsistencies in the traditional approach. This cost reduction is observed to be in 40% as depicted in the graph, since inconsistencies are fixed at the time of its injection at the design phase itself. Similar inference can be drawn for all other inconsistencies.

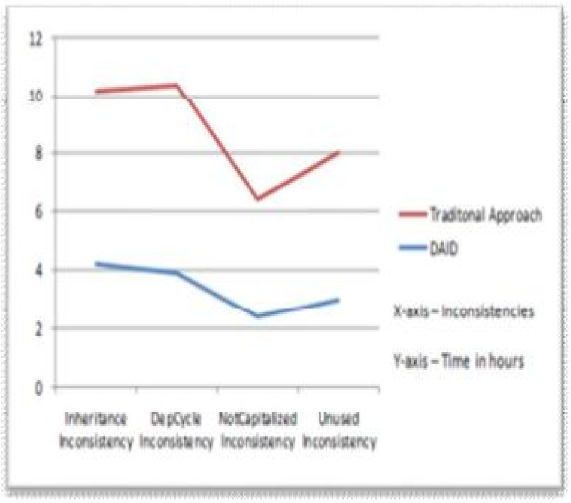

Figure 7. Time comparison

From Figure 7, we can infer that in the case study of ATM application, time required to fix inconsistencies detected during design phase dynamically using DAID has reduced to 4 hours which is less than the time required to rework on the design flaw when detected in the traditional mode after implementation of the design. This is due to fixing of the design inconsistencies at the time of its injection in the design phase itself. Similar inference can be drawn for all other inconsistencies.

VI. CONCLUSION

Development of quality software is one of the trivialactivities of any software industry. However, the current scenarioin most of the IT industries focuses on quality measurement at the final stages and not at all the phases. Since, design phase is one of the injections and dwelling point for several design inconstancies, it is imperative to resolve those flaws without enabling them to propagate. There exists several design quality models which operate in the static mode rather than dynamic detection of design flaws. This paper therefore introducesDynamic and Automated Inconsistency Detection (DAID)model along with consistency rules so as to facilitate fast, accurate, and dynamic detection of design inconsistencies using the predefined design rules right at the design phase itself.The approach described here works on UML models that can be exported to various formats such as XMI (XML Meta data interchange) and their consistency can be checked through software such as SDMetrics.Using the DAID model, the efficiency of the designer can be measured. Further, the complexity of the code and incomplete requirements specification can be tested. This approach further enhances the productivity and the quality of the product in the industry with reduction in cost and time for rework of defects.

REFERENCES

[1]. S.EasterBrook, A.Finkelstein,J.Kramer and B.Nuscibeh, **“Coordinating distributed ViewPoints:The anatomy of a consistency check”**,1994.

[2]. C. Forgy, “**Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem”**, 1982.

[3]. NentwichC,Emmerich W and Finkenstein A: **“Consistency Management with Repair Actions”**,Proc.of the 25th International Confernce on Software Engineering(ICSE), pp. 455-464, 2003

[4]. R.Balzer, “**Tolerating Inconsistency”**,Proc. 13th International Conference on Software Engineering, pp. 158-165, 1991.

[5]. A. Egyed, **“FixingInconsistencies in UML DesignModels”**, Proc.29th International Conference on Software Engineering, pp. 292-301, 2007.

[6]. A. Egyed, E. Letier, and A. Finkelstein, **“Generating and Evaluating Choices for Fixing Inconsistencies in UML Design Models”**, Proc. 23rd International Conference on Automated Software Engineering, 2008.

[7]. I. Groher, A. Reder, and A. Egyed, **“Incremental Consistency Checking of Dynamic Constraints,”** Proc. 12th, International Conference on FundamentalApproaches to Software Engineering, 2010

[8] GauravDuggal,Mrs.BhartiSuri**“Understanding Regression Testing Techniques”**.Visit

http://www.rimtengg.com/coit2008/proceedings/SW15.pdf

[9]. V. Suma, T. R. Gopalakrishnan Nair**, “Effectiveness of Defect prevention in IT for product development”**, International Conference, TeamTech 2008, IISc, Bangalore, proceedings pp 81, 2008.

[10]. Suma V, Arfa Baig, Divyashree B J, N B Sonali, S. Akshaya, “**Dynamic and Automated Technique of Detecting the Design Inconsistencies”**, International Conference on Innovative Computing and Information Processing (ICCIP - 2012), Mahendra Engineering College, Tamil Nadu, India, 29th – 31st March 2012.

⟐⟐⟐