A Survey on UML Modeling and Design Inconsistency

Aumkar R. Ingalikar[1], Sharada G. Kulkarni[2]
1. Student, KLS GIT, Belgaum, Karnataka, India
Email - omkaringalikar@gmail.com
2. Department of CSE, KLS GIT, Belgaum, Karnataka, India
Email - sharadakulkarni@git.edu

Abstract: The importance of modeling language like Unified Modeling Language has been increased in every aspect of software design. A software designer, models a system architecture using different design elements. An element or component is an integral part of the design model. The study of modeling as well as developing consistent design makes good sense in order to give relief to the coders and testers. This paper highlights the UML and the defects in the software design models.

Key Words: Defect, Design, Inconsistency, Models, UML.

INTRODUCTION:

Nowadays, a modeling or designing became a latest trend in software development. The Booch, OMT (Object Modeling Technique) worked well and created a platform for designing a new notation that is UML (Unified Modeling Language). The latest study is highly focused on UML approach and it has covered plenty of jobs done in software design models. In addition, defects in software design models must be carefully handled to reduce more inconsistencies in design models. The mechanism is needed to resolve software engineering issues, especially in the design area. The design is the heart of software development process, hence it is important to take care of design issues. It reflects the overall architecture of the system and directs developers with specific paths through design. Therefore, it is a crucial aspect to remove defects or inconsistencies in software design models prior to the development process to save costs as well as efforts.

UML MODELING:

The importance of UML (Unified Modeling Language) has created a tremendous scope in software development for smooth functioning of design. A stand alone tool introduced by IBM (Rational Rose) is generally used for visual modeling and development that integrates an API level. For professional studies and industrial business, this tool helps to generate strict UML models. It supports various programming languages for the sake of implementation. It finds defects in a number of design components thoroughly without loosing associations among them. It simplifies the complex process of software design, creating a "blueprint" for the construction of software systems, but it is limited only within Class, Object and Sequence diagrams in terms of consistency checking. Recent studies captured a number of software engineering challenges to improve software quality via research. The basic problem is about inconsistencies in complex software systems. Generally, a design consists of hundreds of model elements therefore maintaining accuracy in an overall design is little difficult, but it can be achieved using consistency checkers where some rules are written for creating consistency.

The deficiencies present in software models interrupt the development progress so it must be identified as well as removed from the system for good results. A fault in design can create bad impact on next implementation phase; furthermore at the time of testing more efforts are required for resolving design issues. If the model element violates a design rule it must be instantly handled because of this approach instant model checking is obvious. The previous inconsistency checking was like a batch processing system where all the model elements tested after implementation. The rational Rose is incorporated with consistency checker where the consistencies are moderated for class, object or a sequence diagram. Two or more UML models continuously tested for instant results of defects. If any type of mismatch or missing object recognized, it can by tracked and modified as per requirement. But this is not enough for

few UML models, in fact rest of the UML diagrams carry the same importance. An UML diagrams are categorized into two types, structural and behavioral diagrams.

Almost structural diagrams are covered under Rational Rose but care must be taken for state chart, activity diagram and use case diagram. A use case has no direct relevance to the software development, but it helps a client to clarify the basics. A well formatted SRS helps to create a powerful design model, but it is again important to maintain quality in an entire design model. Similarly, design phase reflects the architecture of the software system and justifies a system thoroughly. It becomes the responsibility to assure quality in design as well. Certainly high level decisions are based on design models for further proceedings. The treatment of the model element acts upon a constraint or a rule. A well formatted rule generally accepted for healthy design model and written by any software engineer. No in depth knowledge or experience is required to construct a design rule. An automated approach can satisfy the need of instant model checking for tracking or detecting defects and it is already acquired in Rational Rose. Typically the software quality is measured in terms of cost and time along with these components, productivity and efforts play a crucial role in the development. Now managing a cost and follow a schedule is necessary for all software projects, but it is important to save time in verification and validation. The batch consistency checking takes more time and efforts, so the birth of instant consistency checking was needed.

RELATED WORK:

Before detecting faults or bugs in a design exemplar it is best to realize [1] the reason of an incompatibility. A design rule also named as a restraint and can be measured well to supply right version. Any misinforming data can make conflicts or errors in the total design, so there must be an efficient technique which will [2] detect an inconsistency in pattern and hence complete design can be generated. The former approaches brought a batch processing where the describing of inconsistencies took place in the more recent levels of development. All the bunch of errors transformed into a single batch and distributed back for improvement and quality of design. The revival of design language like UML has shown the up-gradation in Booch and OMT notations. Today the importance of UML has been increased in every aspect of operations where design plays an important role. Therefore, UML design [3] can be made capable with instant consistency checking for dynamic constraints, so it overcomes the batch processing approach. Sometimes inconsistency can be caused due to the change in any portion of a design, meanwhile other affected parts are possible to fix through automation. While resisting this [4] propagation of inconsistencies manual method was no longer useful, hence new models were added.

Traditionally instant feedback of consistency checking was beneficial, but the computational cost was more; hence implementation of transformation of low level models into an intermediate model for comparison with high level model came into place. Following the model driven architecture [5] helped for refinement of class design models by separating into high and low level. Instead of checking the consistency of independent model and their relationships in the structure can assure consistency of its own. But the rest of the models should hold same consistency, but here this strategy fails. Indeed [6] global consistency must be maintained not only the individual's and therefore merging technique achieved it for variety of conceptual models.

Even though the detection of inconsistency is crucial the respective actions must be taken against each design error this form of work is known as a repair [7] action and it can be distributed in all the design documents. By using a markup language, it is possible to manage an inconsistency; first the elements are linked and represented in Extensible Mark up language. The final documents are arranged in hierarchy using [8] X Path. An objective is to do refinement in all UML design models can not be possible manually. There needs an automated approach for inspection of errors in UML design a suite called as a modeling and visualization framework will support a number of necessary tasks to model and analyze UML diagrams. [9] MINERVA is one of the examples which supports graphical representation of UML diagrams and translate into an intermediate textual representation.

Furthermore, an approach which drives a knowledge base for drawing conclusions as per UML specification, especially applied for managing consistency [10].

CONCLUSION:

The defect in UML design can increase the complexity and reduces the efficiency of a design model. This deficiency can decelerate the performance of a software project. The role of a developer is not only to develop an application as per requirements and architecture but the essential fact is clear understanding of requirement specification and architectural design. This is possible if design remains consistent in its entire life therefore it reduces the size and complexity in overall design. The automated way is to carry out defects are referred as inconsistencies must be reflected, reported and ultimately deleted from the system design. The principles of software engineering define the importance of software development life cycle by describing into phases. In fact software enhances its ability in terms of quality if it satisfies the quality aspects like conciseness, preciseness, completeness, correctness and consistency. It would really help to improve the quality of software by detecting and tracking inconsistencies in software design models.

REFERENCES:

1. Egyed A, "Determining the cause of a Design Model Inconsistency", IEEE Transactions on Software Engineering, vol. 39, Nov 2013 .
2. Egyed.A, "Automatically detecting and tracking inconsistencies in software design models", IEEE Transactions on Software Engineering, vol. 37, 2011.
3. Groher, A. Reder, and A. Egyed, “Instant Consistency Checking of Dynamic Constraints”, Proc. 12th Int’l Conf. Fundamental Approaches to Software Engineering, 2010.
4. Y. Xiong, Z. Hu, H. Zhao, H. Song, M. Takeichi, and H. Mei, "Supporting Automatic Model Inconsistency Fixing", Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), Amsterdam, The Netherlands, 2009.
5. W. Shen, K. Wang, and A. Egyed, “An Efficient and Scalable Approach to Correct Class Model Refinement”, IEEE Transactions on Software Engineering, vol. 35, 2009.
6. M. Sabetzadeh, S. Nejati, S. Liaskos, S. Easterbrook, and M. Chechik, "Consistency Checking of Conceptual Models Via Model Merging", Proceedings of the 15th IEEE International Requirements Engineering Conference (RE), New Delhi, India, 2007.
7. C. Nentwich, W. Emmerich, and A. Finkelstein, "Consistency Management with Repair Actions", Proceedings of the 25th International Conference on Software Engineering (ICSE), Portland, Oregon, USA, 2003, pp. 455-464.
8. C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein, “Xlinkit: A Consistency Checking and Smart Link Generation Service”, ACM Transactions on Internet Technology (TOIT), vol. 2(2), pp. 151-185, 2002.
9. L.A. Campbell, B.H.C. Cheng, W.E. McUmber, and K. Stirewalt, “Automatically Detecting and Visualizing Errors in UML Diagrams”, Requirements Engineering Journal, vol. 7, pp. 264-287, 2002.
10. Zisman and A. Kozlenkov, “Knowledge Base Approach to Consistency Management of UML Specification”, Proc. 16th IEEE Int’l Conf. Automated Software Engineering, pp. 359-363, 2001.