iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** V **Month of publication:** May 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Review of UML Tools and UML Based Software Metrics for Improving Software Quality

Priyanka Makkar[1], Dr. Sunil Sikka[2]
[1]*Amity University Haryana*

I. INTRODUCTION

Software design oriented metrics are useful means for improving the quality of object-oriented software. A number of object-oriented metrics have been suggested by various researchers. These metrics are particularly useful for identifying fault-prone modules and for predicting required maintenance efforts, productivity, and rework efforts. To obtain the design based metrics Unified Modeling Language models are one of the most commonly used artifacts. UML is a standardized, general purpose modeling language in the field of software engineering. The UML includes a set of graphic notation techniques to create visual models of object-oriented software-intensive system. In this paper various UML models, tools and their metrics are reviewed.

Keywords: UML, UML diagrams, metrics, UML tools

II. UML DIAGRAMS

UML has 14 types of diagrams divided into two categories. Seven diagram types represent Structural information, and the other seven represents general types of behavior, including four that represent different aspects of interactions. These diagrams can be categorized hierarchically as shown in the Figure 1.

A. Structure Diagrams

Structure diagram emphasis the things e.g. Class, Object, Component that acts as part of the system being modeled. Structure diagram represent are used to document the software architecture of a system. Following diagrams comes under this category.

1) *Class Diagram:* Describes the structure of a system by using classes, their attributes, and the relationships among the classes such as generalization, composition etc[2][9].
2) *Component Diagram:* Describes how a software system is split up into components and shows the dependencies among these components.
3) *Object Diagram:* Shows a complete or partial View of the structure of an example modeled system at a specific time.

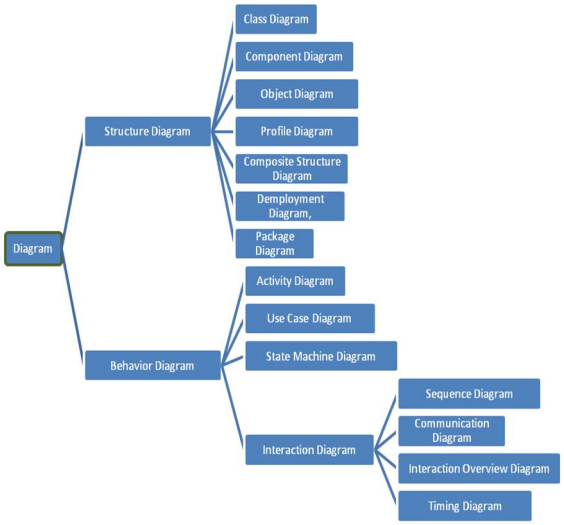


Figure 1 UML Diagram Hierarchy

4) *Profile Diagram:* operates at the meta model level to show stereotypes as classes with the <<stereotype>> stereotype, and

profiles as packages with the <<profile>> stereotype. The extension relation (solid line with closed, filled arrowhead) indicates what meta model element a given stereotype is extending[1].

5) *Composite Structure Diagram:* Describes the internal structure of a class and the collaborations that this structure makes possible.
6) *Deployment Diagram:* Describes the hardware used in system implementations and the execution environments and artifacts deployed on the hardware.
7) *Package Diagram:* describes how a system is split up into logical groupings by showing the dependencies among these groupings.

B. Behaviour Diagrams

It emphasizes what must happen in the system being modeled. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.

1) *Activity Diagram:* describes the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
2) *UML State Machine Diagram*: describes the states and the state transitions of the system.
3) *Use Case Diagram:* Describes the functionality provided by the system in terms of actors, their goals represented as use cases, and any dependencies among those use cases.

C. Interaction Overview Diagram

Provides an overview in which the nodes represent communication diagrams.

1) *Communication Diagram*: shows the interactions between objects or parts in terms of sequenced messages. They represent a combination of information taken from Class, Sequence and Use Case Diagrams describing both static and dynamic behavior of the system.
2) *Interaction Overview Diagram*: provides an overview in which the nodes represent communication diagrams.
3) *Sequence Diagram:* shows how objects communicate with each other in terms of a sequence of messages. Also indicates the lifespan of objects relative to those messages.
4) *Timing Diagram*: A specific type of interaction diagram where the focus is on timing constraints.

III. VIEWS OF UML

A. View of UML Diagrams

UML plays an important role in defining different perspectives of a system. These perspectives are:

1) Logical View design of a system consists of classes, interfaces and collaborations. UML provides class diagram, object diagram to support this.
2) Implementation: defines the components assembled together to make a complete physical system. UML component diagram is used to support the implementation perspective.
3) Process: defines the flow of the system. So the same elements used in *design* are also used to support this perspective.
4) Deployment: represents the physical node of the system that forms the hardware. The deployment diagram is used to support this perspective.
5) Use Case View: It connects all the four views. A Use case view represents the functionality of the system, so all the other perspectives are connected to the same.

IV. POPULAR UML TOOLS [3][6]

A. Rational Rose

Rational Rose is a modeling tool from Rational Software Corporation. Rational Rose (the Rose stands for "Rational Object-oriented Software Engineering") is a visual modeling tool for UML. It comes in different versions suited to different requirements. Rational Rose provides support for all the standard features such as UML diagram support, forward and reverse engineering support, and documentation and round-trip engineering support. Rational Rose also provides support for version control, IDE integration, design pattern modeling, test script generation, and collaborative modeling environment. In addition, Rational Rose also supports the designing of data models within the same environment. An interesting feature of Rational Rose is the ability to publish the UML diagrams as a set of Web pages and images. This enables you to share and distribute your application design where the Rational

Rose tool is not installed.

B. Poseidon

Poseidon from Gentle ware has its roots in the Agrium open source project. The Agrium modeling tool evolved as an open source effort and is a useful, full-featured UML tool freely available under the Open Publication License. Gentle ware has taken Agrium a step further and turned it into a good modeling tool. Poseidon comes in different flavors suited to different requirements. Poseidon supports forward and reverse engineering and documentation generation by using special-purpose plug-ins. Gentle ware has not forgotten its open source moorings and offers the Poseidon for UML Community Edition 1.5 free for individual software developers.

C. Together control center

Together Control Center (formerly from Together soft) from Borland is an entire suite of visual modeling tools for UML. Together Control Center supports UML diagrams, MVC modeling, forward and reverse engineering, and round-trip engineering, as well as integration with IDEs such as IBM Web Sphere Studio. It supports comprehensive documentation and a powerful collaborative modeling environment. An added feature of Together Control Center is the pattern repository. The pattern repository similar to the template-driven modeling concept makes frequently used diagrams and design patterns readily available for reuse in modeling. Together Control Center supports the Rational Unified Process as well as the Extreme Programming methodologies.

V. UML BASED SOFTWARE METRICS [4]

To achieve the software quality it is important to measure the software quality. In software engineering we have lot of software metrics available to measure software quality in terms of reliability, efficiency and maintainability etc. software metric is used to quantify various attributes of the software. Since the beginning of software engineering researchers are working on the metrics that can be used at early stages of software development for that purpose UML models are commonly used for driving the metrics at the early stage of software development.

A. Metrics for Class [9][2]

1) Number of the attributes in a class – unweighted
2) Number of the attributes in a class – weighted
3) Number of the operations in a class – unweighted
4) Number of the operations in a class – weighted
5) Number of the associations linked to a class
6) Coupling between classes
7) Depth of inheritance tre
8) Number of the super classes of a clas
9) Number of the elements in the transitive closure of the super classes of a clas
10) Number of the subclasses of a clas
11) Number of the elements in the transitive closure of the subclasses of a clas
12) Number of messages sent by the instantiated objects of a clas
13) Number of messages received by the instantiated objects of a class

B. Metrics for Message[2]

1) Number of the directly dispatched messages of a message
2) Number of the elements in the transitive closure of the directly dispatched messages of a message

C. Metrics for Use Case[8]

1) Number of actors associated with a use case
2) Number of message associated with a use case
3) Number of system classes associated with a use case

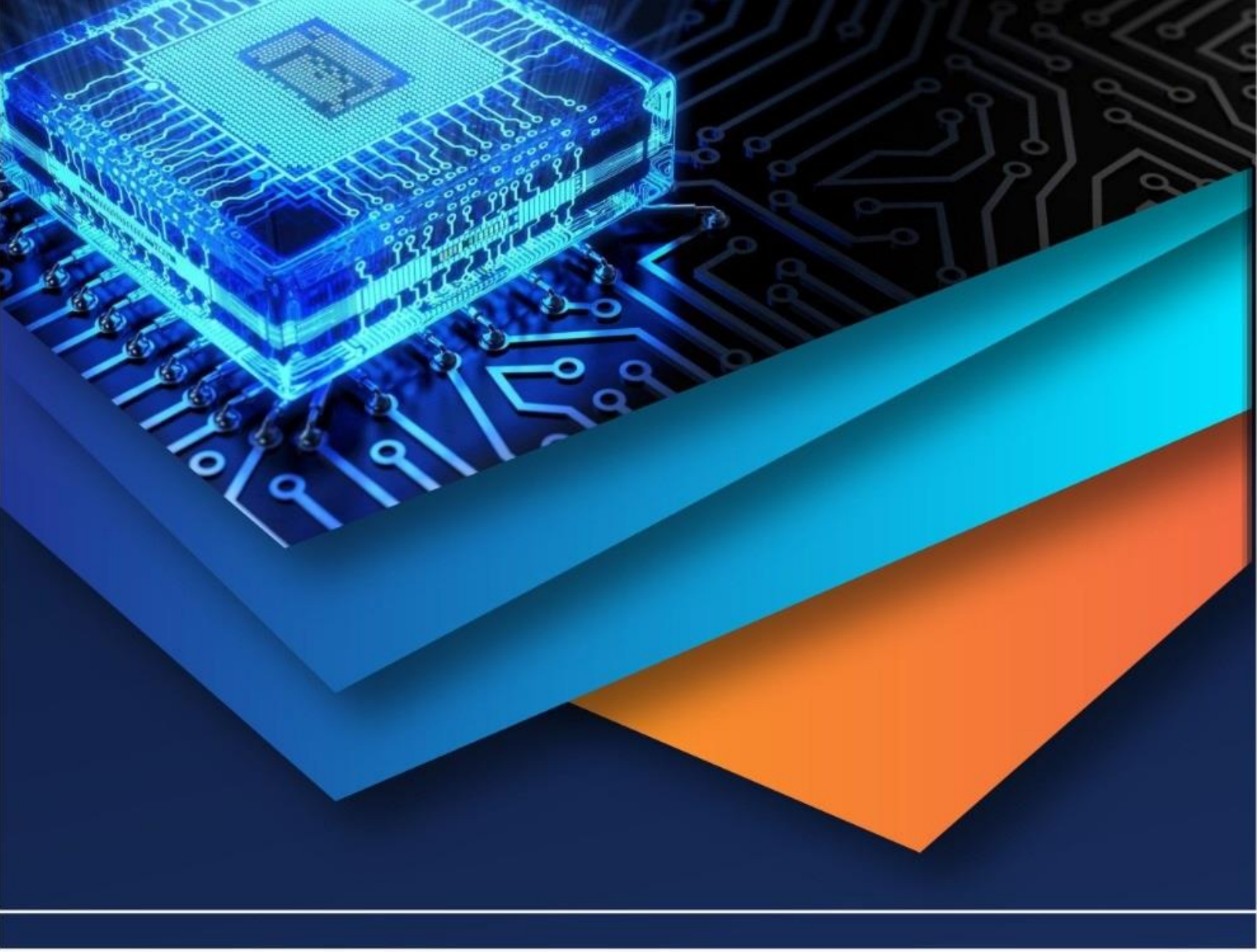
VI. CONCLUSIONS

A brief review of UML diagrams, tools and metrics is presented in this paper. More software metrics are available in the literature

based on the various UML diagrams which need to be study and analysis.

REFERENCES

[1] L. Nenonen, J. Gustafsson, J. Paakki, A. I. Verkamo, Measuring object-oriented software architectures from UML diagrams; In: Proc. 4th Intl. ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, France, June 2000, 87-100
[2] M. Genero, M. Piattini, C. Calero A Survey of Metrics for UML Class Diagrams**,** in Journal of Object Technology, Vol. 4, No. 9, Nov-Dec 2005.
[3] Ye Peilei, A object-oriented development process and UML modeling tools, 2009 IITA International Conference on Services Science, Management and Engineering, 978-0-7695-3729-0/09, IEEE.
[4] Ariadi Nugroho, Michel R. V. Chaudron, The impact of UML modeling on defect density and defect resolution time in a proprietary system**,** Emprical Software Engineering August 2014, Volume 19, issue 4 pp 926-954.
[5] Soumya George, Sivapriya Avinash, John T Abraham" Object Oriented Design Metrics" in journal computer society of india Volume 11, Number 3, March 2014
[6] Namita Khurana, Rajender Singh Chhillar , Usha Chhillar A Novel Technique for Generation and Optimization of Test Cases Using Use Case, Sequence, Activity Diagram and Genetic Algorithm Journal of Software Volume 11, Number 3, March 2016
[7] Orenyi A., Basri S., Tan Jung L., Object-Oriented Software Maintainability Measurement in the past Decade, International Conference on Advanced Computer Science Applications and Technologies, 2012, 257-262.
[8] Preety Verma Dhaka, Amita Sharma "Effect of Different UML Diagrams to Evaluate the Size Metrics for Different Software Projects" in Global journal of computer science and technology, vol. xv ,issue VIII version I,2015, ISSN: 0975-4172
[9] M. Genero, M Piattini, C. Calero: "A Survey of Metrics for UML Class Diagrams", in Journal of Object Technology, vol. 4, no. 9, November-December 2005, pp. 59-92.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY