Importance and Impact of Class Diagram in Software Development

Article · April 2020

CITATIONS
2

READS
596

4 authors:

Abdul Hafeez
Sindh Madressatul Islam University
28 PUBLICATIONS **161** CITATIONS
SEE PROFILE

Mansoor Khuhro
Sindh Madressatul Islam University
46 PUBLICATIONS **132** CITATIONS
SEE PROFILE

Muhammad Furqan
University of Engineering and Technology Lahore
84 PUBLICATIONS **1,494** CITATIONS
SEE PROFILE

Dr Imtiaz Husain
Sindh Madressatul Islam University
46 PUBLICATIONS **338** CITATIONS
SEE PROFILE

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 12(25), DOI: 10.17485/ijst/2019/v12i25/145739, July 2019*

Importance and Impact of Class Diagram in Software Development

Abdul Hafeez[1*], Mansoor Ahmed[1], Muhammad Furqan[2] ,Wasi-Ur-Rehaman[3] and Imtiaz Husain[1]

[1]Department of Computer Science, SMI University; karachi,ahkhan@smiu.edu.pk
[2]Department of Infromation Technology, Higher Technology Collage Muscat; muhammadfurqan@hct.edu.om
[3]Techway System, Musacat; wasi@gmail.com

Abstract

Background/Objectives: This work mainly focuses on the role of UML Class diagram in different phases of software development cycle and shows the industry responses regarding its impact and importance. **Methods/Statistical Analysis**: we conducted industry survey and collected data from software practitioners. Participants were asked about their usage of UML for modeling software artifacts and most frequent diagram they use. **Findings:** Result shows that UML Class diagram is most important element of UML and it performed highly significant role in software development. It is most frequently used diagram. **Application/Improvements:** Usage of UML models during the software development improve the quality of software and UML Class diagram is most important part of UML and help practitioners to perform analysis and as well as design in a systematic way.

Keywords: Software Class Development, Software Class Diagram, Software System

1. Introduction

UML is an industry standard and significantly used visual modeling language for the modern software system[1]. It hides the complexity of system and provides an appropriate level of abstraction[2]. It is used in the software industry for software specification, analysis, design and documentation. Nowadays, it is also used for code generation[3,4]. UML offers a number of diagrams for dealing various aspects of software modeling[2,5]. It uses simple diagrammatic notations for describing software. Therefore, customer can also easily understand the specification and design of the system. Currently, it is also used in engineering, ontology development, DBMS and other discipline of technology[6]. In Model Driven Architecture, UML considers as a core and the entire process of the MDA move around the UML Models[7,8].

2. Research Hypothesis

Research suggests that class diagram is the most important part of UML[9–16]. It performs an important role in software analysis and design[17]. It describes the system through concepts, their relationship and constraint over concept[17]. It's also a key ingredient of the MDA process[17,18]. Therefore, we advanced the following research hypothesis:

H0: The UML Class diagram is considered as a core diagram which is used in the different activities in the software development.

H1: Class diagram is not an important artifact in software development.

For determining importance and impact of class diagram in software development, an industrial survey was performed and investigated:

*Author for correspondence

- Which UML diagram mostly use in software development (TCD).
- Is Class diagram improves the understanding of business domain (ICDIUBD)?
- Is Class diagram improves the communication among the team members (ICDICAT)?
- Is Class diagram gives a blueprint of system to maintenance engineer to get a general idea of how the software is structured before investigating the code (ICDBIC)?
- Is Class diagram assists development of building other UML diagrams (ICDAUML)?

3. Research Methodology

In the survey, data were collected from (N = 71) software practitioners. The sample consisted of 13 (18.8%) international software practitioners and 58 national software practitioner (81.2%). Survey questionnaire consisted of three sections. In Section 1 practitioners were inquired about their demographic information (e.g. Name, email, Work Location, Position). In Section 2 they were asked about their usage of UML for modeling software artifacts and most frequent diagram they use. Finally, in Section 3 they were asked their use of different diagrams.

4. Research Results

The sample consisted of 32 (45.07%) software developers, 11 (15.49%) Software Analysts, 12 (16.90%) Team Leads, 9 (12.68%) Project Managers, 4 (5.63%) Software Designer and 3 (4.23%) Software Testers as shown in Figure 1.

In the sample 2 practitioners (2.82%) have more than 10 years of experience, 7 practitioners (9.86%) have 5 to 10 years of experience, 19 practitioners (26.76%) have 3 to 5 years of experience and 43 practitioners (60.56%) have 1 to 2 years of experiences as shown in Figure 2.

As per result shown in Figure 3, majority of the practitioners (89.73%) use UML for modeling software artifacts. Only (11.27%) stated they do not use UML.

As can be seen in Figure 4, in response of most frequent diagram for modeling they use. (47.89%) reported class diagram, (29.58%) reported use case diagram, (11.27%) reported Activity diagram, (4.23%) reported component and deployment diagram, (2.82%) reported state chart diagram, (1.41%) reported sequence diagram and (0.0%) reported communication, composite structure, timing, and interaction overview. Our hypothesis is, the UML Class diagram is considered as a core diagram which is used in the different activities in the software development. Whereas, alternate hypothesis is class diagram is not an important artifact in software development. Correlation analysis was performed to observe the effectiveness of UML Class diagram. The Table 1 shows that the class diagram was found to be highly significant and integral diagram which improves the understanding of business domain as well as improve the communication among the team members with $r = 0.008$ at $P < 0.05$. Whereas, class diagram is very helpful in maintenances phase and help out maintenances engineer to get a general idea of how the software is structured before investigating the code with $r = 0.066$ at $P < 0.01$ and also assist the development of building other UML diagrams ($r = 0.012$; $P<0.05$).

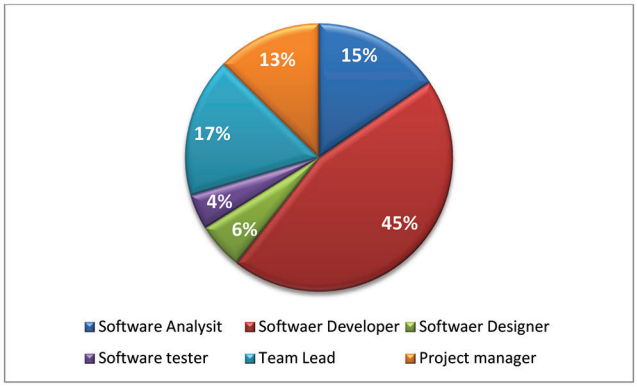

Figure 1. Percentage of Software Practitioner in survey.

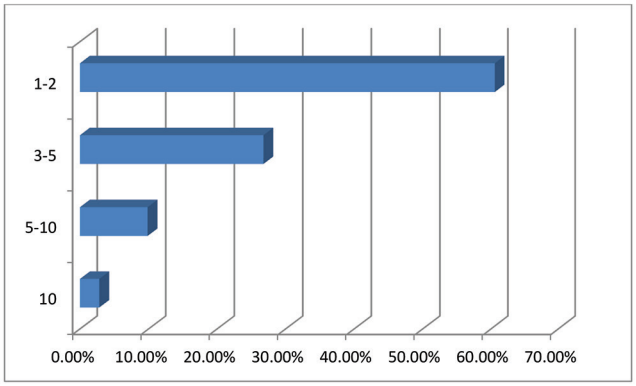

Figure 2. Participants industry experience.

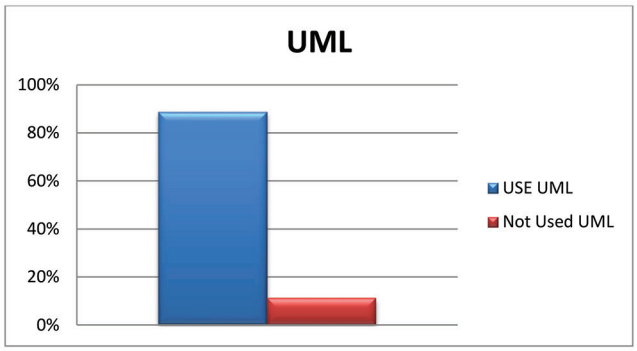

Figure 3. Use of UML in industry.

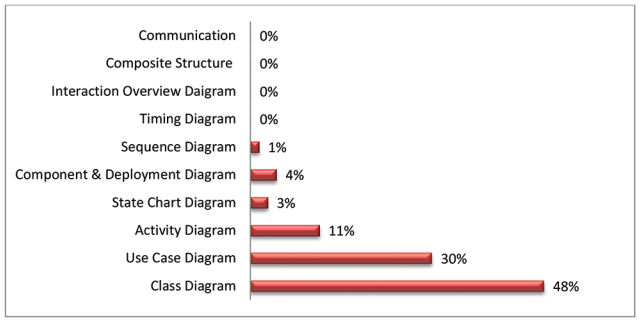

Figure 4. Most frequently used diagram of UML.

Table 1. Correlations

Control Variables			ICDIUBD	ICDICAT	ICDBIC	ICDAUML
Types of Class used for Software Development	ICDIUBD	Correlation	1.000	0.315	0.221	0.298
		Significance (2-tailed)		0.008*	0.066**	0.012*
	ICDICAT	Correlation		1.000	0.255	0.322
		Significance (2-tailed)			0.033**	0.007
	ICDBIC	Correlation			1.000	0.187
		Significance (2-tailed)				0.120*
	ICDAUML	Correlation				1.000
		Significance (2-tailed)				

5. Conclusion

UML Class diagram is most important element of UML and it performed highly significant role in software development. It is most frequently used diagram. It improves the communication among the team member, gives a blueprint of system to maintenance engineer to get a general idea of how the software is structured before investigating the code and assists development of building other UML diagrams.

6. References

1. Fu C, Yang D, Zhang X, Hu H. An approach to translating OCL invariants into OWL 2 DL axioms for checking inconsistency. Automated Software Engineering. 2017; 24(2):295–339. https://doi.org/10.1007/s10515-017-0210-9
2. Artale A, Calvanese D, Angelica I. Full satisfiability of UML Class diagrams. Conceptual Modeling - ER Lecture Notes in Computer Science. 2010; 6412:317–31 https://doi.org/10.1007/978-3-642-16373-9_23
3. Anastasakis K, Bordbar B, Georg G, Ray I. On challenges of model transformation from UML to Alloy. Software and Systems Modeling. 2010; 9(1):69–86. https://doi.org/10.1007/s10270-008-0110-3
4. Szlenk M. Formal-semantics-reasoning-UML-class-diagram. Dependability of Computer Systems. DepCos-RELCOMEX '06. 2006; 59:25–7. https://doi.org/10.1109/DEPCOS-RELCOMEX.2006.27
5. Cadoli M, Calvanese D, Giacomo G, Mancini T. Finate satisfibaility of UML Class diagram by constraint programming. Proc of the 2004 International Workshop on Description Logics. vol 104 of CEUR Workshop Proceedings; 2004. p. 1–15.
6. Maraee A, Balaban M. Efficient reasoning about finite satisability of UML Class diagram with contrained generalization sets. Model Driven Architecture - Foundations and Applications Lecture Notes in Computer Science. 2007; 4530:17–31. https://doi.org/10.1007/978-3-540-72901-3_2
7. Hafeez Khan A, Hyder Abbas Musavi S, Rehman A, Shaikh A. Ontology-based finite satisfiability of UML Class model. IEEE Access. 2018; 6:3040–50. https://doi.org/10.1109/ACCESS.2017.2786781
8. Shaikh A,Wiil UK. Overview of slicing and feedback techniques for efficient verification of UML/OCL Class diagrams. IEEE Access. 2018; 6:23864–82. https://doi.org/10.1109/ACCESS.2018.2797695

9. Gonzlez CA, Cabot J. Formal verification of static software models in MDE: A systematic review. Information and Software Technology. 2014; 56(8):821–38. https://doi.org/10.1016/j.infsof.2014.03.003
10. Balaban M, Maraee A. Finite satisfiability of UML Class diagrams with constrained class hierarchy. ACM Transactions on Software Engineering and Methodology - In memoriam, fault detection and localization, formal methods, modeling and design TOSEM Homepage archive. 2013; 22(3). https://doi.org/10.1145/2491509.2491518
11. Malgouyres H, Motet G. A UML model consistency verification approach based on metamodeling formalization. SAC '06 Proceedings of the 2006 ACM Symposium on Applied Computing; 2006. p. 1804–9. https://doi.org/10.1145/1141277.1141703
12. Maraee A, Balaban M. Efficient recognition of finite satisfiability in UML Class diagrams strengthening by propagation of disjoint constraints. Model-based Systems Engineering, MBSE. 2009; 1(8):2–5. https://doi.org/10.1109/MBSE.2009.5031714
13. Maraee A, Balaban M, Vicktor. Efficient recognition and detection of finite satisfiability problems in UML Class diagram. 1st International Workshop on Model Co-evolution and Consistency Management; 2008.
14. Berardi D, Calvanese D, Giacomo GD. Reasoning on UML Class diagrams. Artificial Intelligence. 2005; 168(1-2):70–118. https://doi.org/10.1016/j.artint.2005.05.003
15. Berardi D, Calvanese D, De Giacomo G. Reasoning on UML Class diagrams is EXPTIME-hard. Proc of the Description Logic Workshop; 2003. p. 28–37.
16. Shaikh A, Wiil UK. A feedback technique for unsatisfiable UMLOCL Class diagrams. Software Practice and Experience. Wiley Journal. 2013. https://doi.org/10.1002/spe.2211
17. Shaikh A, Wiil U. UML to CSP (UOST). A tool for efficient verification of UMLOCL Class diagrams through model slicing. Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering; 2012. p. 1–4. https://doi.org/10.1145/2393596.2393639
18. Shaikh A, Wiil UK. Evaluation of tools and slicing techniques for effcient verifcation of UMLOCL Class. Advances in Software Engineering Archive. New York, NY, United States: Hindawi Publishing Corp; 2011. p. 18. https://doi.org/10.1155/2011/370198