DOI: 10.24996/ijs.2021.SI.1.29

ISSN: 0067-2904

Unified Modelling Language (UML) Effect on the Total Cost of Ownership (TCO) for a Software Development

Ahmed Altaher
Al-Nahrain University Baghdad, Iraq

Abstract:
UML (Unfiled Modeling Language), known as the standard method for object-oriented (analysis and design) modeling, includes other languages which enables it to implement a prototype of the structure and behaviors of the product. This paper attempts to explore the observations about UML role on the cost of software maintenance, and hence on the Total Cost of Ownership (TCO) of a software product. It is therefore important to investigate the benefits obtained through modeling..

Keywords: Unfiled Modeling Language, Total Cost of Ownership (TCO), Software

Introduction:

"UML was develop as a general-purpose language together with fundamental categories to extend the UML towards problem" domain-specific profiles [1].

Software engineering rules, by using the offered tools and aspects in standard modeling structure, aim to make obvious the complexity, step by step progress (as much able in general) and probable errors in software creation or development task for all participates (founder, stack holders, programmers, etc.) The system model becomes as contract between the owner and builder, which is understand able to everyone.

System Model

A system model could be present as documentation document for the software, which during maintenance, helps the developing team to realize how the software produced so any changes or fixing may needed, there is map to direct expertise through the software engine.

If such a documentation was not available a reverse procedure is done from the existing software, so the experts conceive the system model, in this case the system model should be more dedicated where the developing team & the owner can discuss all required changes.

System Model Role:

Authors concludes that "system model has more than a crucial role in directing ahead the engineering process of developing software, but also in reverse engineering. Hence techniques are studied and developed to understand and document existing legacy systems, to update their functionality or to integrate them into larger systems" [1].

In continues of this reality now days one of important quality requirements for almost all of the software products, is the simplicity of software maintenance in which affects directly to the TCO of software product.

However, lots of researchers in this aspect announce that, evaluating this issue from different perspectives is necessary, since multi direct and indirect conditions are involved in which they affect it [2].

UML and TCO

According to the published documents by different researchers and authors the factors which affect the study about the UML role in software maintenance & TCO are: kind of the software (project),

* Email: altaher@nahrainuniv.edu.iq

Time (process, delivery, recapturing), correctness, bug fixing, suppliers knowledge about UML models, life term of the product, end users and etc.(Gilb, 2008)
Which each research may concentrates on some (two or three) figures to make the study and measurements.
Individual classes of change will have separate demands for each class, and in result of that there would be different methods & technical tools proposed as solution. Well defining the requirements and corresponding designs is very essential, (exactly what types of change are expected) [3].
Most they believe, in general for major products the system model if is not done in appropriate and correct structure, if it would not lead to failure it would cause increase in the TCO with no positive results.
Watson, 2014, "as soon as you start the later, you finish". Means that starting a project directly throw the final part programing will not end to finish in shorter time successfully.
Having a documentation of the product procedures during creation is helpful if not necessary. The same group of builders does not always do development of a product, here the developing group will have more prosperous results by having the documentation.

Case study

In study done by [2] they have attempt to realize that does UML documentation decreases costs related to code changes, for being able to do this analysis, they had to measure time consumed for completing the maintenance tasks of the experiment.
Therefore, the experiment had "one independent variable (use of UML documentation) and two treatments (UML, no-UML)".
It had "four dependent variables:
Time to perform the change excluding diagram modifications (T).
Time to perform the change including diagram modifications (T').
The correctness of the change (Co).
The quality of the changed design (Q)".
They have implemented the study in two different environment:

1- Oslo experiment involves 3rd year informatics students.
2- Ottawa (Carleton) experiment involved 4th year Computer/Software engineering students.

In each location they have divided, participate into two teams (UML, no-UML).
For the experiment, they have proposed two systems and required tasks as follow (Table-1):
The results of the two experiments are almost constant. For each of the dependent variables the outcomes were as follow:
(T): UML does overall help save effort.
(T'): No effort savings are visible.
(Co): UML had a notable, "positive impact on the most complicated tasks
(Q): (Ottawa experiment investigated" this variable) UML helped to achieve higher design

Table 1-comparing two systems outcome

System	Task	Description	Oslo	Ottawa (Carleton)
Simple ATM system	A	"Print out an account transaction statement"	X	X
	B	"Transfer money between two accounts"	-	X
Vending machine serving hot drinks	C	"Implement a coin return-button"	X	-
	D	"Make bouillon as a new type of drink"	X	-
	E	"Check whether all ingredients are available for the selected drink"	X	X
	F	"Make your own, customized drink based on the available ingredients"	Time sink	X

Conclusion

UML "contain the definition of a software process model [4] [5] in addition to methods which transforms a UML model into a analogous implementation in a programming language Over all having a structured plan such as UML methods for implementing a product by considering, the

individual requirements will cause a better result than not having one. UML is good for major products which delivering date can be fixed (especially not related to the market competition).
Nevertheless for small products or products determined to a short deadline delivery spending too much time on the preparations (planning, designing structures, etc.) are not acceptable and will be known as part of the problem.

References

1. Engels, G., HÜcking, R., Sauer, St. A. Wagner. **1999**. UML Collaboration Diagrams and Their Transformation to Java. Proceedings of the 2nd international conference on The unified modeling language: beyond the standard October 1999 Pages 473–488
2. Erik Arisholm, Lionel C. Briand, Siw Elisabeth Hove, Yvan Labiche. **2005**. The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. Simula Technical Report 2005-14 and Carleton Technical Report SCE-05-11, Version 2 (7 – 42)
3. Tom Gilb. **2008**. Designing Maintainability in Software Engineering: A Quantified Approach, Result Planning Limited, Copyright © 2008 by Tom Gilb, 1-10. Published and used by INCOSE with permission. Tom.Gilb@INCOSE.org
4. Jacobson, I. Booch, G. and Rumbaugh, J. **1999**. *The Unified Software Development Process*, Addison-Wesley, Reading, MA, 1999
5. D'Souza, D. and Wills, A. **1998**. *Objects, Components, and Frameworks with UML the Catalysis Approach*. Addison-Wesley, 1998