THINK INDIA JOURNAL

ISSN:0971-1260

Vol-22-Issue-10-November-2019

Agility And Linearity In UML Diagram For Design Patterns In Software

Prerna Bisaa[1] ,Dr Vishal Goar[2]
[1]Prerna Bissa,Research Scholar,Tantia University,ShriGanganagar
{prerna.bissa@gmail.com}
[2]Dr Vishal Goar,assistant professor,Govt. Engineering College Of Bikaner
{dr.vishalgoar@gmail.com}

***Abstract* –design pattern is a specification and visualization that is used in the software engineering field. It can be defined as a general purpose language that uses a graphical design notation which can create an conceptual model. This conceptual model can then be used in a system. This system is called the designmodel. These design patterns have some standard for specifying, constructing, visualizing and documenting the object of software systems, for business modelling and other software systems. This study focuses on the importance of design diagrams (class, use case, sequence, activity, component, deployment, collaboration) used for very Large-sized projects i.e. s/w that requires more than 2 years for development. As per the data analysis design diagram considered as the most important diagram among all design diagrams while Collaboration diagram considered as least important diagram. Sequence diagram has slight upper hand over use case diagram & treated marginally more important diagram.Also component diagram plays vital role than deployment diagram during s/w development.comparision performed on iterative diagram and sequential diagram.**

***Keywords* – class diagram, UML, sequence diagram,**

I Introduction:UML, design patterns are a visualize language that is used in the software engineering field. It can be defined as a general purpose language that uses a graphical designation which can create an conceptual model. This conceptual model can then be used in a system. This system is called the DataPattern model. The Object Management group is responsible for defining DP, and they do this via the metamodel.
The Modeling Language is commonly used to visualize and construct systems which are software intensive. Because software has become much more complex in recent years, developers are finding it more challenging to build complex applications within short time periods. Even when they do, these software applications are often filled with bugs, and it can take programmers weeks to find and fix them. This is time that has been wasted, since an approach could have been used which would have reduced the number of bugs before the application was completed. it should be emphasized that DPare not limited simply modeling software. It can also be used to build models for system engineering, business processes, and organization structures. A special language called Systems Modeling Language was designed to handle systems which were defined within UML 2.0. The Unified Modeling Language is important for a number of reasons. First, it has been used as a catalyst for the advancement of technologies which are model driven, and some of these include Model Driven Development and Model Driven Architecture. It emphasis has been placed on the importance of graphics notation, DP is proficient in meeting this demand, and it can be used to represent behaviors, classes, and aggregation. While software developers were forced to deal with more rudimentary issues in the past, languages like UML have now allowed them to focus on the structure and design of their software programs. It should also be noted that UML models can be transformed into various other representations, often without a great deal of effort.
Three ways to apply UML:

- UML as a sketch:informal and in-complete diagram(often handsketh on whiteboard) created to explore difficult parts of the problems or solution spaceexploiting the power of visual languages.
- UML as a blueprint:relatively detailrd design diagram used,reverse engineering to visualize and better understanding existing code in uml diagram

UML as a programming language: complete executable specification of a software system in uml,executable code will be automatically generated

Three perspective to apply design pattems:it describes raw digram types,suc as class diagram and sequence diagram it does not superimpose a modelling perspectives in these for example the same design notation can be used to draw pictures of concepts in the real world or software classes in java

There are three perspectives of design patterns:

a) Conceptual perspective the diagrams are interpreted as describing things in a situation of the real world or domain of interest.
b) Specification (software) perspective the diagrams (using the same notation as in the conceptual perspective) describe software abstractions or components with specifications and interfaces, but no commitment to a particular implementation (for example, not specifically a class in C# or Java).
c) implementation (software) perspective the diagrams describe software implementations in a particular technology (such as Java).

A method superimposes alternative terminology on top of the raw UML. For example, in the UP, when the UML boxes are drawn in the Domain Model, they are called domain concepts or conceptual classes; the Domain Model shows a conceptual perspective. In the UP, when UML boxes are drawn in the Design Model, they are

ISSN:0971-1260

Vol-22-Issue-10-November-2019

called design classes; the Design Model shows a specification or implementation perspective, as desired by the modeller.

Finally, there are a few **UML case studies** which will help you to get an idea of how to start using **UML diagrams** to represent the information of a software system. These are the fundamental elements in UML. Every diagram can be represented using these building blocks. The building blocks of UML contains three types of elements. They are:

1) Things (object oriented parts of uml)
2) Relationships (relational parts of uml)
3) Diagrams

Things
A diagram can be viewed as a graph containing vertices and edges. In UML, vertices are replaced by things, and the edges are replaced by relationships. There are four types of things in UML. They are:

1) Structural things (nouns of uml – static parts)
2) Behavioral things (verbs of uml – dynamic parts)
3) Grouping things (organizational parts)
4) Annotational things (explanatory parts)

IILiterature Review –

"On the Contribution of UML Diagrams to Software System Comprehension" The study of IritHadar and Orit Hazzan, Technion – Israel Institute of Technology, Israel found that "UML was utilized by the students as a multifaceted expression tool. The way that the different teams sorted the diagrams in preparation for the comprehension process, the different essential diagrams that they leaned on, and the number of "trips" made to each of the different diagram types, all indicate that the process of comprehension and information extraction from UML diagrams differs between different people. It was also found that, when taken together, no one diagram type was globally less or more important than the others for the performance of the comprehension task. In other words, the differences in preference between the various teams cancelled out each other. The above conclusions are based on the work of senior computer science students."

Another study on *"A Critical Analysis and Treatment of Important UML Diagrams Enhancing Modelling Power"* of FahadAlhumaidanCollege of Computer Sciences and IT, King Faisal University, Hofuf, KSA found that "Unified Modelling Language (UML) is used at early phases of software development because of having a reasonable support of diagrams and notations but has not proved sufficient for the complete modelling of functional and non-functional requirements of a system. Based on our experience of applying UML, some weaknesses in the diagrams are identified in this paper and a treatment is presented. For example, most of the UML structures are based on graphical notations and are prone to causing errors. The hidden semantics under the diagrams allow ambiguities at design level and multiple notations produce inconsistent and ambiguous models. Further, the models described using UML diagrams may have multiple interpretations and the recipients of the design may not be able to understand what has been put in the diagrams. There exists some well-established approaches, for example for- mal methods, which can capture the semantics hidden under the UML diagrams.

Formal methods are useful at all stages of software development because of having rigorous mathematical and computer tools support. However, at the current stage of development, formal methods are not sufficient in complete modelling of a system. In this way, UML and formal methods are both useful for design and specification of software systems but an integration of these approaches will facilitate the software development process."

III Need of the Study –

Unified Modeling Language is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development. During this different diagrams are drawn according to need & the types of requirement. The need of this study is to find out the importance of different diagrams (class, use-case, sequence, activity, component, deployment, collaboration) during very Large-sized software development. Actually to check whether UML diagrams has some role in very large-Sized s/w or not. Whether all diagrams plays vital role in s/w development or not; or we can focus on specific diagrams rather than drawing all the diagrams.

IVMethodology

In this paper, two diagrams and sequences of the samples were taken, and a document editor design pattern was drawn on them, to show how one diagram works quick and the other one is linearly. we have compare structural part and behaviroal part with example.in which one diagram is static and another one is dynamic,diagrm have creted by draw.io open source tool .Finally, there are a few **UML case studies** which will help you to get an idea of how to start using **UML diagrams** to represent the information of a software system.this case study diagram on edit text editor.

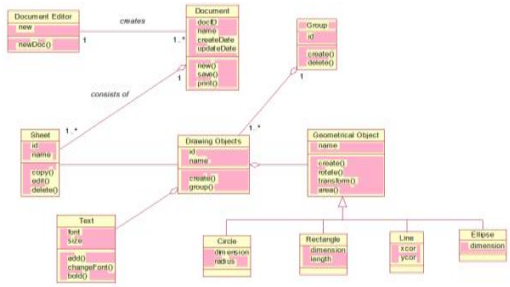


Fig.1 Class diagram

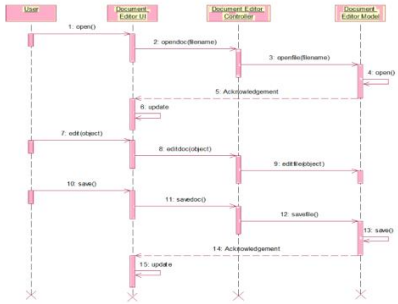


Fig 2 Sequence_Diagram

As a part of study they have to fill the questionnaire in which they have asked to select any one option out of 4 as per the importance of UML diagrams.

ISSN:0971-1260

Vol-22-Issue-10-November-2019

Question Options	No.Of Respondents	Points
MostDyn	12 Persons	12X4=48
Dynamic	12 Persons	12X3=36
LessDyna	00 Persons	0X2=0

Class Diagram – sequence Diagram

1. Most Static
2. Static
3. Less static

The points are assigned to each option of the above question.

1. Most Static - 4 points
2. Static - 3 points
3. Less Static - 2 points

Ex.If 10 respondents select options 1.Mandatory for class dia. then 10 X 4= 40 points will be considered.The diagram which gets max points will be considered as the most important diagram for Medium projects.

Following Results are found during the study for UML diagrams:

1.Class Diagram –

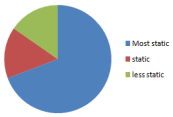
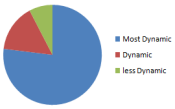


Fig.3 Responses for class diagram

Question Options	No. Of Respondents	Points
Most static	18 Persons	18X4=72
Static	4 Persons	4X3=12
Less static	4 Persons	4X2=8

Table I

2.Sequence Diagram



According to TABLE-II it can easily be observed that 12 respondent assumes that sequence is an very important UML diagram in Large-Sized s/w development

TABLE-II

V Conclusion:

From the study it has been observed that all the UML diagrams are not usually favoured for very Large-Sized projects as the risk factor, complexity is very high. It can also be concluded from the analysis that none of the diagram gets above 60% for the answer 1. static except Class diagram. As per the data analysis Class diagram considered as the most important diagram among all UML diagrams while Sequence diagram considered as important diagram. Sequence diagram has slight upper hand over use case diagram & treated dynamically more important diagram. Also these diagram plays vital role when deployment diagram during s/w development.During further research papers I would like to find out the importance of different UML diagrams in different category/types of s/w development

VI Refrences

1. www.agilemodeling.com/essays/umlDiagrams.htm
2. www.smartdraw.com/resources/tutorials/uml-diagrams/
3. www.uml-diagrams.org/
4. www.tutorialspoint.com/uml/uml_standard_diagrams
5. www.ibm.com/developerworks/rational/library/.../bell/
6. www.sinventions.com/whitepapers/uml/3000_borcon_uml.html
7. Principles of Object- Oriented Software Development - Anton Eliens ,Addison Wesley.
8. Object Oriented System Development - Ali BahramiMcGRAW-HILL International Edition.
9. Object-Oriented Software Engineering -Ivar Jacobson Pearson Education INC
10. Applying UML And Pattern - Craig Larman Pearson Education INC
11. UML Distilled - Martin Flowler Pearson Education INC
12. The Unified Modeling Language UserGuide - Grady Booch, James Rumbaugh, Ivar Jacobson-Pearson Education INC
13. The Unified Modeling Language Reference Guide- Grady Booch, James Rumbaugh, Ivar Jacobson-Pearson Education INC