Empirical study on Usage of Object Oriented Analysis and Design Techniques among Software Development Professionals

Kulwant Kaur*
Hardeep Singh**

Abstract

The UML (Unified Modeling Language) is considered to be most widely accepted in software industry. In this paper, we provide empirical findings from a created a web survey designed for UML users on the utility of UML amongst fifty one professional software engineers using Object Oriented Analysis and Design. During survey various aspects were considered which include standards and procedures maintained, design and implementation of a System and effectiveness of measures used during software development. There is lot of variation in UML usage. In general the use of UML has a positive effect on the quality of the system, that is, it reduces defect during software development.

Keywords : Unified Modeling Language, OOAD

1. Introduction

The Unified Modeling Language (UML) is used to graphically represent the architecture, organiza tion, logic, and design of a software system using a powerful graphical notation. The graphical constructs of the language allow the description of both the static and dynamic properties of the object model, covering virtually all object model analysis and design aspects treated in different analysis and design methods. (Hazem Hamed, Ashraf Salem 2001)

The use of a graphical notation drastically simplifies the object model design process however, it requires professional and experienced software design background in order to produce a perfected object model, and no matter how well trained and experienced the software designer may be, he will always be in need to automated design analysis tools in addition to the design description tools provided by UML. The analysis and checking of an object model design is considered one of the most important activities carried on by the software designer, this is because in complex software systems, where the object model is mostly deployed, design considerations and constraints are too many to handle for a single designer or even a complete design team increasing the need for analysis tools that will automate the process of checking the object model. (Hazem Hamed, Ashraf

*Department of Computer Applications, Apeejay Institute of Management, Jalandhar, INDIA.
**Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar, INDIA.

Salem 2001)

The use of UML in various industrial sector like Business Orientated Systems; Technically Orientated Systems and Embedded and Control Systems varies from system to system. We can categorize UML projects into small, medium and large. Thus we can analyze through survey

- Design and Implementation of a system
- During implementation, how strictly program structure, dependency relations, inheritance relations, classes and methods and order of method calls are implemented.
- Investigation are needed to check degree of completeness, inconsistency in UML design
- Productive activities of Software Development and analyze imperfections etc.

In addition, we investigate the OO software professionals about impact of quality in terms of final software product, model and productivity. We will also check with professionals regarding understanding ease of adoption of the tool used and aspects of UML framework. It also some queries related to software development with respect to maintenance and testing. By considering above adoptions a survey was performed using web based questionnaire in which various software professional engineers of different companies participated.

This paper presents a review of related work in the following section. Section 3 describes the respondent's background. Section 4 explains Standards and Procedures Maintained by the organizations. In section 5 we describe the designs and implementation of a system using UML. The explanation of utility and preference of metrics is described in section 6. Section 7 presents the conclusion.

2. Related Work

Briand L. C., Labiche Y. Penta M. D. , Yan-Bondoc H. (2005) elaborated the results of two controlled experiments that investigate the impact of using OCL on three software engineering activities using UML analysis models: detection of model defects through inspections, comprehension of the system logic and functionality, and impact analysis of changes. The results show that, once past an initial learning curve, significant benefits can be obtained by using OCL in combination with UML analysis diagrams to form a precise UML analysis model. But, this result is however conditioned on providing substantial, thorough training to the experiment participants.

According to Arisholm E., Briand L. C., Hove S. E. and Labiche Y. (2006), the availability of UML documentation may result in significant improvements in the functional correctness of changes as well as the quality of their design. However, there does not seem to be any saving of time and for simpler tasks, the time needed to update the UML documentation may be substantial compared with the potential benefits, thus motivating the need for UML tools with better support for software maintenance.

Ordonez Mauricio J., Haddad Hisham M. (2008) elaborated the role of metrics in software quality and work is an effort to bring more attention to software metrics. Their study examines the practices of metrics in software industry and the experiences of some organizations that have developed, promoted, and utilized variety of software metrics. As various types of metrics are being developed and used, these experiences show evidence of benefits and improvements in quality and reliability.

Nugroho A. and Chaudron M. R. V. (2008) explored software engineers' opinions on common styles of using UML and how they perceive the impact of using UML on productivity and quality in software development. The results was based on the impact of using the UML on productivity is perceived mostly in the design, analysis, and implementation phases.

Nugroho A. and Chaudron M. R. V. (2009) discussed report on an empirical investigation on the impact of UML modeling on the quality of software system. In particular, they focus on defect density as a measure of software quality using an industrial case study and found that the use of UML modeling potentially reduces defect density in software system.

3. Respondents Background

The survey was conducted from May 2009 to February 2010. A web based questionnaire is used and only those software professionals are targeted who are implementing UML models. In total 193 responses were received and out of which 51 responses were considered practical for further analysis and other responses were incomplete as only few options were answered. The major respondents are originally from companies with more than 1000 number of employees involved in software engineering and company with age about 30 years and about 53% employees involved in Software Engineering in that company. Maximum companies surveyed have level of investment up to Rs. 10 crore +. Respondent's primary role on ongoing or completed UML projects id described in Table 1.

Table 1: Respondents primary role on ongoing or completed UML projects

Role	Number of Respondents	Percentage of Respondents
Programmer/developer	21	41
Project leader (day-to-day responsibility for this project)	16	31
Business/operations analyst (determining requirements)	12	24
Architect/designer	12	24
System testing/verification	5	10
Senior manager (overall responsibility across multiple projects)	4	8
Database administrator/analyst	3	6
Training/education (including student)	3	6
User interface designer	2	4
Client or client representative	2	4
System integrator	1	2
System Design Engineer	1	2

Respondents those participated in survey, include 74% from Business Orientated Systems (for instance sales and marketing support systems, personnel management and administration systems, financial information systems, reservation systems, office automation systems, etc.); 20 % from Technically Orientated Systems (for instance computer aided design/manufacturing systems, computer aided software engineering tools, geographic information systems, simulation systems, document image processing systems, etc.); and 6% from Embedded and Control Systems (for instance in industrial

automation and process control systems, signal processing systems, telecommunications etc.). Major Organization's primary involvement in software industry is based on Software user (developed in-house) and Software vendor (producing custom software systems) and few are involved in Software user (developed by a 3rd party).

47% of respondents originated from India, 23% from United Kingdom and 28% from United States. In terms of education background, it is observed that 12% respondents working as programmer since two years, 25% working since 5 years, and 63% respondents involved in programming from more than five years. Out of these programmers 16% have graduation qualification rest have masters. 16% of programmers undertaken less than 4 projects; 22% taken up to six projects; 16% was involved in 7-10 projects and 47% were involved with more than 10 projects. From these programmers who have more than 2 years experience in various languages, 49% were involved with Java, 37% in C++, 14% in C# and rest in have experience Oracle, SQL Server, PL/SQL, Pro *C, Siebel, Visual Basics, Testing tools programming, UNIX scripts , ASP, Delphi, Action scripting, AJAX (Web Development), COBOL, C, J2EE, .net etc.

4. Standards and Procedures Maintained

About 80% of Software professionals agreed that there is a procedure for maintaining awareness of the state-of-the-art in CASE or software engineering technology. According to some professionals, state of art is actually the development at particular time, but technology becomes matured and in practice, real life problems are not stable changing frequently, this is because thousands of case tools are generated day to day, also the process is not standardized from the programmer's side.

A. Software project on Contractual basis: As per 98% of employees, management formally assesses the benefits, viability, and risk of each software project prior to making contractual (or internal) commitments. According some software professionals, this gives the pre estimation to both client and developers which enable them to perform cost benefit analysis. Some have opinion some companies have formal procedure of conducting feasibility study involving viability, identifying risks, formulating a plan before making any contractual commitments.

B. Conducting periodic reviews: Maximum software professionals agreed that management formally conduct periodic reviews of the status of each software project. As per their opinion, periodic reviews provides the quantitative or qualitative information to Management (metrics, models and measurements), which is necessary to control budget as well as schedule overruns and with it all the inconsistencies will be determined earlier. Some suggested that audits should be carried out every six months by QAG/Audit group. Apart from this, peer reviews, internal QA, external QA is mandatory for all deliverables to the client.

C. Software Subcontracting Organizations: Only 55% organizations which contract out software development to other organizations to follow a disciplined software development process.

D. Project Independent Audits: Each project should have independent audits (such as inspections or walkthroughs) conducted for each major stage in the software development process. 90% software professionals were in favor and suggested that one should ensure the reliability of the software project, each code need to be walkthrough by second developer; these should be part of internal QA and peer reviews and Mandatory for all deliverables to the client sometimes.

E. Documented procedure for estimating software size : Only 74% of organizations developing predominantly in a single language such as C or COBOL. A documented procedure created is used for estimating software size (such as "Lines of Source Code") and thus for using productivity measures. According to Software professionals, coding is not always same from every programmer, also language dependencies is there, from specification we cannot judge the LOC value like in FP, also for example if one programmer write the code for a problem in 2KDLOC, it may be possible by other programmers to write the code for the same problem even in same language in less amount of delivered lines of code.

F. Software Development Effort : Almost 90 % professionals agreed with formal procedure used to produce software development effort, schedule, and cost estimates. As per one of the programmer, Procedure is available such as COCOMO, but again depends upon the LOC value or pre estimated constants A and B. All of these are not generalized but give the approximation.

G. Formal procedure for deliverable : All software professional agreed that there a formal procedure (such as a review or handover with sign-off) used whenever a deliverable (such as a user statement of requirements or system requirements) is passed from one discrete group to another (e.g. user to analyst to designer) to ensure it is properly understood. Some software professionals suggested there should be Work break down structure with team work process. According to them, at most stages, there are some instances where the handover is more informal, and this is usually between teams within the same area. There should be both review and handover with sign-off (exit criteria) for each phase of SDLC. Getting clear from Requirements from Business is a Challenge.

H. In-House systems and Package development : About 76% organizations where all software projects don't have to be funded externally such as In-House systems and Package development. According to respondents, there should be a mechanism to ensure that the systems projects selected for development qualitatively or quantitatively support/alleviate the organization's business objective / problems. According to a professional, Risk Analysis is the process which will control all these issues, Risk analysis is a technique to identify and assess factors that makes interruption in the success of a project or achieving a goal.

I. To ensure that the functionality, strengths, and weaknesses of the "system" : Only 79% organizations whose software is intended to replace a previously computer-based mechanical or clerical set of tasks and 6% showed inability to accept this option. As per opinion for professionals, reengineering of the Legacy system or Software maintenance and all functional requirements listed in requirements documents have to be met and passed. For strengths and weaknesses of system, performance testing is conducted and report/results should be shared with client.

J. Test planning commence prior to programming : According to 96% of software professionals, test planning should commence prior to programming beginning based on the user requirements and high-level design documents. Regarding

comments, some said because this will ensure the expected input output validation, also the programming structured will be created according to that. This is usually the case, but can sometimes be later. Testing team has to be part of discussions during user requirements and designing phase. High level test plan is formulated at this stage not always.

K. Independent Testing : 90% respondents agreed that independent testing conducted by users (or appropriate representatives) under the guidance of Software Quality Assurance before any system or enhancement goes live. According to them, testing always require something in executable form. Some suggested for some projects where new web systems will be used externally by third parties. UAT sign-off is mandatory for all enhancements and maintenance projects.

L. Check that the system configuration: Only 88% respondents agreed that there is a procedure to check that the system configuration (i.e. the programs and any data) passing user acceptance testing is the same as that which is implemented for live operation and that no changes are made directly to a "live" version of any system (other than through modification to its development version). Some suggested software configuration management handles the version and release issues. Some are of the view that all code is placed in VSS and testing servers and production servers get only the executables, not source. It is the responsibility of the integration team to ensure this.

5. Designs and implementation of a system

There are different ways to maintain design and implementation issues. In survey, discussed then on the basis of methods of correspondence between UML designs and implementation and how strictly UML design is implemented.

In order to study various aspects of software developers UML projects were categorized into three categories: Small, Medium and Large. 24% Software developers participated in the survey worked on small projects, 47% worked on medium projects and about 29% worked on large projects.

Regarding study utility of UML among Software professionals involved in projects for the past 10 years 24% chosen option for less than 25% of the projects, 28% used UML in 50% projects, about 12% used UML for 75% of the projects and only 8% used UML for more than 75% of the projects.

A. Ease of Adoption and understanding aspects of UML framework

An attempt was made in this study to determine the reasons Ease of adoption. Respondents were asked to rank the reasons for ease of adoption on a five point scale.

Table 2 reveals that the very high reason for Ease of use when tool have "user friendly design" (4.05); high when it can handle "multiple projects in the Workspace view" (4.00), somewhat low when it supports only "GUI Standards"

Table 2: Descriptive for Reasons for Ease of Use

	Minimum	Maximum	Mean	SD
Tool support latest GUI standards	2.00	5.00	3.98	0.79
Tool have user-friendly design environment	2.00	5.00	4.05	0.77
Program start with a reasonable set of defaults	1.00	5.00	3.68	1.04
Tool facilitate integration of new designers into a team	1.00	5.00	3.85	0.94
Tool have multiple projects in the Workspace view	2.00	5.00	4.00	0.74
Tool facilitate navigation through all the diagrams and classes in the model	2.00	5.00	3.90	0.86

Note: The mean is the average on a scale of 1 = Very Low; 2 = Low; 3 =Somewhat Low; 4 = High; 5 = very High; SD=Standard Deviation; Number of respondents=41

(3.98) and low when it facilitates "integration of new designers into a team" (3.85).

Regarding simplification the customization table 3 reveals that the high reason is to "facilitate customization of the view of a class and its details" (4.00) and somewhat low when tool "support customization capabilities through the use of macros" (3.68).

Table 3: Descriptive for Reasons Simplifying the customization

	Minimum	Maximum	Mean	SD
Tool support customization capabilities through the use of macros	1.00	5.00	3.68	0.93
Tool facilitate customization of the view of a class and its details	3.00	5.00	4.00	0.74

Note: The mean is the average on a scale of 1 = Very Low; 2 = Low; 3 =Somewhat Low; 4 = High; 5 = very High; SD=Standard Deviation; Number of respondents=41

In order to model existing code, table 4 indicates that the very high reason is to support "integration into data modeling tools" and "support Dynamic Model Verification by live generation of sequence diagram of real-time behavior" both have mean 4.0. It is high when "facilitate generation classes from existing database structures" (3.98) and somewhat low when "Tool generate either a sequence or collaboration diagram based on actual, recorded object interactions" (3.68).

Table 4: Descriptive for Reasons Modeling the existing code

	Minimum	Maximum	Mean	SD
Tool facilitate generation classes from existing database structures	2.00	5.00	3.98	0.79
Tool support integration into data modeling tools	2.00	5.00	4.00	0.77
Tool support Dynamic Model Verification by live generation of sequence diagram of real-time behavior	3.00	5.00	4.00	0.74
Tool generate either a sequence or collaboration diagram based on actual, recorded object interactions	2.00	5.00	3.68	0.85

Note: The mean is the average on a scale of 1 = Very Low; 2 = Low; 3 =Somewhat Low; 4 = High; 5 = very High; SD=Standard Deviation; Number of respondents=41

An attempt was also made in this find to determine the understanding following aspects of UML framework. Respondents were asked to rank the reasons for difficulty on a five point scale. Table 5 shows the descriptive for difficulty in understanding aspects for UML Framework.

Table 5: Descriptive for difficulty in understanding aspects for UML Framework

	Minimum	Maximum	Mean	SD
Understanding individual classes and their method	1.00	5.00	2.73	1.03
Using abstract classes and interfaces	1.00	5.00	2.73	0.95
Mapping your solution too framework code	1.00	5.00	2.93	0.98
Understanding the structure of inheritance hierarchies and object composition	1.00	5.00	2.76	0.97
Understanding design patterns	1.00	5.00	2.66	0.88
Understanding the dynamic structures of the framework	1.00	5.00	2.85	0.85
Choosing from alternative framework solution strategies	2.00	5.00	3.07	0.93
Understanding UML problem domain	1.00	5.00	2.68	0.99

Note: The mean is the average on a scale of 1 = Very easy; 2 = Easy; 3 = Moderate; 4 = Hard; 5 = Very Hard ; SD=Standard Deviation; Number of respondents=41

The correlations between descriptive for difficult aspects for UML Framework understanding were examined, as shown in Table 6, and it appears that correlations exist between certain difficulties.

Table 6: Correlation Matrix for difficulty in understanding aspects for UML Framework

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) Understanding individual classes and their method	1							
(2) Using abstract classes and interfaces	.695**	1						
(3) Mapping your solution too framework code	.649**	.540**	1					
(4) Understanding the structure of inheritance hierarchies and object composition	.612**	.797**	.714**	1				
(5) Understanding design patterns	.780**	.633**	.775**	.660**	1			
(6) Understanding the dynamic structures of framework	.554**	.537**	.582**	.440**	.629**	1		
(7) Choosing from alternative framework solution strategies	.387*	.305	.741**	.463**	.668**	.579**	1	
(8) Understanding UML problem domain	.582**	.762**	.697**	.859**	.676**	.508**	.461**	1

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

B. Correspondence between Design and Implementation

Generally, UML designs are considered as the basis for an implementation of a system. In order to implement a system, system has the same classes and dependencies as prescribed in the design. An implementation is proportional to design. Therefore maintaining correspondence is the effort to keep design and implementation identical. In order to check rate of importance of correspondence between design and implementation 41% opted for very important option, 29% software developers opted extremely important and only 2% opted for somewhat important.

Various methods are used in project by software professionals to maintain the correspondence between UML designs and the implementation. According to 64% software developers, they use manually review and update the mode, 15% software professionals used reverse engineer to implement code and 17% used round trip engineering technique. Only 4% indicated no special efforts need to spend on maintaining correspondence.

In order to maintain the correspondence between UML designs and the implementation, 36% opted that it is done after final software release. 36% agreed that design and implementation should be done on continuation basis during coding activities. Only 9% said maintenance can be carried out on weekly basis and 19% opted that it can be done on monthly basis. According to 6% software professionals, no special effort needs to be spent on maintaining correspondence.

To check importance of correspondence between implementation and associated UML design in project only 16% accepted it as extremely important, 27% agreed that it is very important 11% said it is fairly important and according to 46% software professionals it is important.

C. Implementation & completeness of UML design

i. Degree of Completeness: The purpose of a software design is to specify a system that is going to be developed. Everyone regard a design as having a high degree of completeness if it specifies all parts of a system. Based on the experience of software professionals, they were asked to rate the degree of completeness of the provided UML designs in describing the systems to be developed. None of the software professional rated very low. 2% rated low, 27% opted somewhat low, 59% rated somewhat high and 14% rated high.

ii. Factors that restrict from UML design in an implementation: To analyze this factor four point scale was used. Table 7 reveals that all factors force to deviate from a UML design in an implementation for completeness. But "Meeting approaching deadline" and "incomplete design" are seems to be more important than others.

Table 7: Descriptives of factors force to deviate from a UML design in an implementation

	Minimum	Maximum	Mean	SD
Meeting approaching deadline	1.00	4.00	2.23	0.72
Impractical design	1.00	4.00	2.19	0.79
Incomplete design	1.00	4.00	2.21	0.80
Design does not satisfy requirements	1.00	4.00	2.09	0.68

Note: The mean is the average on a scale of 1: Never; 2: Sometimes; 3: Often and 4: Very Often ; SD=Standard Deviation; Number of respondents=43

D. Clarification of the design in projects that use UML compared to projects that do not use UML

In order to clarify functioning of programmers, they were asked whether they approach designers and ask for Clarification of the design in projects that use UML compared to projects that do not use UML. According to software professionals only 9% ask very less often, 27% ask a bit less often, 46% were neutral about their reply, 16% agreed they ask somewhat more often and only 2% replied they ask much more often.

E. Use of UML to check activities of Software Development

To check utility of UML help or it hinder software professional is considered to be more productive in different activities of software development. Detailed results are depicted in table 8.

Table 8: Descriptive reasons for hindrance and its correlation matrix during software development

	Mean	SD	(1)	(2)	(3)	(4)	(5)
(1) Analysis/Problem Understanding	5.47	1.22	1				
(2) Design	5.70	1.08	.686**	1			
(3) Implementation	5.61	0.98	.376*	.199	1		
(4) Testing	5.30	1.166	.217	.112	.399**	1	
(5) Maintenance	5.30	1.166	.083	.093	.462**	.632**	1

Note: The mean is the average on a scale of 1 = Very hinder; 2 = Hinders; 3 = Somewhat hinders; 4 = Neutral; 5 = Somewhat helpful; 6 = Helpful; 7 = Very Helpful;
SD=Standard Deviation; Number of respondents=43
**. Correlation is significant at the 0.01 level (2-tailed).
*. Correlation is significant at the 0.05 level (2-tailed).

To check completeness of UML design during Software Development, we check for imperfections in UML designs may appear in various forms like

i. Inconsistency which reveals contradicting information in portraying a design concern.
ii. Understandability relates to the degree in which concepts are easily inferred from a design.
iii. Inaccuracy relates to the lack of preciseness in specifying design concerns.
iv. Incompleteness refers to a design's low coverage in specifying all parts of a system.

F. Utility of UML for quality properties of the Final Software Product

The use of UML influences the quality properties like satisfying requirements, correctness, modularity, testability and understandability of the final software product. All these factors were considered important by the software professionals.

Some software's are facing the problem to master the model quality. The respondents were asked to identify the reasons under three parameters "Standardizing the notation", "Maintaining the model integrity and comprehensiveness" and

"Guaranteeing the model consistency" on a five point scale. Table 9 shows the descriptives reasons for Master the model quality. According to table, "define and check the naming and notation rules" is most preferred standardizing notation, "check the impact of modifications" is most desired for maintaining the model integrity & comprehensiveness and usage of simulation functions to check diagram meaning and to check modeling accuracy in diagrams and to "debug" the UML model" is considered to be best for Guaranteeing the model consistency.

Table 9: Descriptives reasons for Master the model quality

	Minimum	Maximum	Mean	SD
Standardizing the notation				
Use same graphic notation for requirements and specifications	1.00	5.00	3.95	0.88
Facilitate Collaboration within the design team for naming rules	2.00	5.00	3.95	0.85
Define and check the naming and notation rules	2.00	5.00	4.05	0.75
Maintaining the model integrity and comprehensiveness				
Apply the proper corrections when modification is made	2.00	5.00	3.98	0.83
Check the impact of modifications	2.00	5.00	4.10	0.81
Checks integrity online when designers edit the model, or offline with impact analysis functions	2.00	5.00	3.95	0.85
Guaranteeing the model consistency				
Uses Simulation functions to check diagram meaning and to check modeling accuracy in diagrams and to "debug" the UML model	1.00	5.00	4.03	0.83
Check integrity automatically	1.00	5.00	3.90	0.98

Note: The mean is the average on a scale of 1 = Very Low;
2 = Low; 3 = Somewhat Low; 4 = Somewhat High;
5 = High; SD=Standard Deviation; Number of respondents=40

6. Utility and preference of Metrics

A. **Project Resources and Estimates :** According to 94% software professionals, records of actual project resources and timescales versus estimates should be maintained (at individual resource/resource-type level) and regularly analyzed/ fed-back into the estimating and scheduling procedures. Some professionals suggested only then cost and quality objectives are met

B. **Software Size Maintenance :** As per records of software size maintained for each software configuration item, over time, and fed-back into the estimating process, only 75% supported it and according to 24% software professionals it is not applicable to their firm. According to software professionals, only rework and impact analysis will be carried out as per requirement.

C. **Source of Errors in Software Code :** When software professionals asked about statistics on the sources of errors in software code gathered and analyzed for their cause, detection and avoidance measures, 84% agreed that their firms are taking measures for it. According to some of professionals, this is necessary for debugging, ensure prevention in incoming future against similar type of errors and should not be a formal step in the project

D. **Test Efficiency :** 83% of software professionals supported that statistics on test efficiency (% of errors actually detected by an activity against the maximum theoretically possible) gathered and analysed for all testing stages in the development process and commented it as better way to provide high percentage of reliability.

E. **Usage of Project Tracking throughout the Software :** In order to attain more accuracy 81% of software professionals supported, "earned value" project tracking used throughout the software development process (actual versus planned deliverables analyses, designed, unit tested, system tested, acceptance tested over time) to monitor project progress.

F. **Performance expectations and Computer Resources Constraints :** As non functional requirement satisfaction 83% of software professionals, supported estimates made and compared with actual for target computer performance (e.g. memory utilization, processor throughput and file/channel I/O and disk usage. This condition is applicable to explicit or implicit performance expectations or computer resource constraints.

G. **Post Implementation Software Problems :** According to 83% of software professionals, post implementation software problem reports logged and their resolution effectively tracked and analysed. According to some software professionals it requires much of the time to detect and uncover errors because whole of development already takes place, dependencies and interconnection is more, tracking and removing will sometimes involve Ripple effect

H. **Detailed Versions and variants of Software Systems :** Every types of documentation is available whether it is coding manual, test cases, configuration detail including versions and release, design documents, risk analysis reports etc. 94% software professionals agreed that records exist from which (and requiring nothing extra) all current versions and variants of software systems and their components can be quickly and accurately reconstructed in the development environment

7. Conclusion

During survey various aspects for standards and procedures maintained was considered such as Software project on Contractual basis, Conducting periodic reviews, Software Subcontracting Organizations contract out software development to other organizations, Project independent Audits, Documented procedure for estimating software size, Software Development Effort, Formal procedure for deliverable, In-House systems and Package development, to ensure that the functionality, strengths, and weaknesses of the "system", test planning commence prior to programming, independent testing, Check that the system configuration, respondents in general considered they are very important and actual needed during software development.

There are different ways to maintain design and implementation issues. In survey, discussed then on the basis of methods of

correspondence between UML designs and implementation and how strictly UML design is implemented. In the above discussion, issues covered include ease of adoption and understanding aspects of UML framework; correspondence between design and implementation; Implementation & completeness of UML design with respect to degree of completeness & factors that restrict from UML design in implementation; clarification of the design in projects that use UML compared to projects that do not use UML; use of UML to check activities of Software Development and utility of UML for quality properties of the Final Software Product. During discussion Software Professional view is discussed.

One of major aspect covered during this survey was utility and preferences in metrics during software development. In this regard various considerations were taken which include project resources and estimates, Software Size Maintenance, Source of Errors in Software Code, Test Efficiency, Usage of Project Tracking throughout the Software, Performance expectations and Computer Resources Constraints, Post Implementation Software Problems, Detailed Versions and variants of Software Systems,

Whenever increase in model coverage, it might help to reduce incorrect implementation of systems. There is lot of variation in UML usage. In general the use of UML has a positive effect on the quality of the system, that is, it reduces defect during software development.

8. References

1. *Arisholm E., Briand L. C., Hove S. E. and Labiche Y. (2006). The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. IEEE Transactions on Software Engineering, Volume 32 , Issue 6, Pages: 365-381*
2. *Blaha M. R., Rumbaugh J., Jacobson I. (2008). Object Oriented Modeling and Design with UML. Second Edition, Pearson education.*
3. *Booch G., Maksimchuk R.A. (2008). Object Oriented Analysis and Design with Application, Pearson education.*
4. *Booch G., Rumbaugh J., Jacobson I. (2008). The Unified Modeling Language User Guide. Second Edition, Pearson education.*
5. *Briand L. C., Labiche Y. Penta M. D. , Yan-Bondoc H. (2005). An Experimental Investigation of Formality in UML-Based Development. IEEE Transactions on Software Engineering, vol. 31, no. 10, pp. 833-849*
6. *Fenton N.E. and Pfleeger A.L. (2004). Software Metrics : A Rigorous & Practical Approach. Thomson Computer Press.*
7. *Hazem Hamed, Ashraf Salem (2001). UML-L: An UML Based Design Description Language., ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'01), pp.0438*
8. *Ivan Porres (2001). Modeling and analyzing software Behavior in UML, http://www.tucs.fi. ISBN: 951-29-2169-3*
9. *Lorenz M. and Kidd J. (1994). Object Oriented Software Metrics : A Practical Guide. Printice Hall Object Oriented Series.*
10. *Ordonez Mauricio J., Haddad Hisham M. (2008). The State of Metrics in Software Industry. IEEE Fifth International Conference on Information Technology: New Generations. Pp. 453-458.*
11. *Nugroho A. and Chaudron M. R. V.(2008). A survey into the rigor of UML use and its perceived impact on quality and productivity. ESEM 2008: 90-99*
12. *Nugroho A, Chaudron M. R. (2009) Evaluating the Impact of UML Modeling on Software Quality: An Industrial Case Study. Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems, Pages: 181 – 195.*
13. *Pressman R. (2005). Software Engineering: A Practitioner's Approach. McGraw-Hill.*
14. *Satzinger J.W., Jackson R.B., Burd S.D. (2007).Object Oriented Analysis and Design with the Unified Process, Thomson Edition*
15. *Sommerville I. (2004). Software Engineering. Sixth edition. Pearson Education.*