International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-2, Issue-3, July 2012

A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models

Maneela Tuteja, Gaurav Dubey

Abstract*— In recent years, software testing is becoming more popular and important in the software development industry. Indeed, software testing is a broad term encircling a variety of activities along the development cycle and beyond, aimed at different goals. Hence, software testing research faces a collection of challenges. A consistent roadmap of most relevant challenges is proposed here. In it, the starting point is constituted by some important past achievements, while the destination consists of two major identified goals to which research ultimately leads, but which remains as reachable as goals. The routes from the achievements to the goals are paved by outstanding research challenges, which are discussed in the paper along with the ongoing work.

Software testing is as old as the hills in the history of digital computers. The testing of software is an important means of assessing the software to determine its quality. Since testing typically consumes 40~50% of development efforts, and consumes more effort for systems that require higher levels of reliability, it is a significant part of the software engineering

Software testing is a very broad area, which involves many other technical and non-technical areas, such as specification, design and implementation, maintenance, process and management issues in software engineering. Our study focuses on the state of the art in testing techniques, as well as the latest techniques which representing the future direction of this area.

Today, testing is the most challenging and dominating activity used by industry, therefore, improvement in its effectiveness, both with respect to the time and resources, is taken as a major factor by many researchers

The purpose of testing can be quality assurance, verification, and validation or reliability estimation. It is a tradeoff between budget, time and quality. Software Quality is the central concern of software engineering. Testing is the single most widely used approach to ensuring software quality.

(Keywords: SDLC, Software quality, Testing techniq Technique .)

I. INTRODUCTION

I. Introduction: Software Testing

Software testing is the process of executing a program or system with the intent of finding errors. Software is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails. Most physical systems fail in a fixed (and reasonably small) set of ways. By contrast, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible.

Discovering the design defects in software, is equally difficult, for the same reason of complexity. Because software and any digital systems are not continuous, testing boundary values are not sufficient to guarantee correctness. All the possible values need to be tested and verified, but complete testing is infeasible. Exhaustively testing a simple program to add only two integer inputs of 32-bits (yielding 2^64 distinct test cases) would take hundreds of years, even if tests were performed at a rate of thousands per second.

The iterative process of software testing consists of

1. Designing tests
2. Executing tests
3. Identifying problems.
4. Getting problems fixed.

A.Objective of Testing

The objective of testing is to find problems and fix them to improve quality. Software testing typically represents 40% of a software development budget. There are four main objectives of testing:

(a) Demonstration: It shows that, system can be used for integration with acceptable risk. It demonstrates functions under special conditions and shows that products are ready for integration or use.

(b) Detection: It discovers defects, errors and deficiencies. Determines system capabilities and limitations quality of components, work products and the system.

(c) Prevention: It provides information to prevent or reduce the number of errors clarify system specifications and performance. Identify ways to avoid risk and problems in the future.

(d) Improving Quality: By doing effective testing, we can minimize errors and hence improve the quality of software.

II. TYPES OF SOFTWARE TESTING

There are various types of testing techniques that have been invented. Each testing technique serves a different purpose for testing different artifacts like designing, coding and planning software requirement specification.

All the testing techniques are divided into mainly three categories, black box, white box or grey box testing. Techniques which tests external behavior of the system are categorized in Black box testing and which tests internal behavior of the system are termed as white box testing and

Manuscript received: on July, 2012

Maneela Tuteja, Department of Information TechnologyDronacharya College of Engineering, Gurgaon, Haryana,.

Gaurav Dubey, Amity School of Computer Sciences, Amity University, Uttar Pradesh,India.,

which tests both internal and external behavior are called grey box testing.

Software Testing Techniques		
Black box Testing	**White box Testing**	**Grey box Testing**
Functional and system testing, Stress testing, Performance testing, Usability testing, Acceptance Testing, Beta testing, Ad hoc Testing, Regression testing, Intersystem testing, Volume testing, Parallal testing, Boundary value	Unit testing, Error handling testing, Desk checking, Code walk through, Code reviews and inspection, Code coverage testing, Statement/ Path/ Function/ Condition testing, complexity testing/ Cyclomatic complexity, Mutation testing	Integration Testing, Regression Testing.

III. SDLC AND QUALITY ASSURANCE

SDLC refers to software development life cycle, i.e. the various stages used in the life cycle of software development. There are various software development approaches defined and designed which are used during development process of software, these approaches are also referred as "Software Development Process Models".

A. SDLC Models

Software development life cycle is basically a systematic way of developing software. It includes various phases starting from the functional requirement of software (means what software is supposed to do). After that designing takes place then development and then testing. After testing is finished, the source code is generally released for Unit Acceptance Testing (UAT) in client testing environment. After approval from client, the source code is released into production environment [1]. There is various software development approaches defined and designed which are used during development process of software, these approaches are also referred as "Software Development Process Models". Each process model follows a particular life cycle in order to ensure success in process of software development.

Various types of SDLC Models are :

- Water-fall Model
- Prototype Model
- RAD Model
- V Model

- ***The waterfall Model***

Waterfall approach was first process model to be introduced and followed widely in software engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate process phases. The phases in Waterfall model are: Requirement Specifications phase, Software Design, Implementation, Testing, Deployment of system & Maintenance [1]. All these phases are cascaded to each other so that second phase is started as and when defined set of goals are achieved for first phase and it is signed off, so the name "Waterfall Model". Fig 3.1 shows all the phases in water fall model.

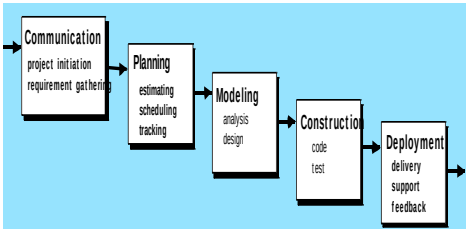

Fig (a) : *The phases of Waterfall Model*

Advantages

In Waterfall model, every phase is implemented in a sequential order. Waterfall model is used where the duration of project is very less, and it is best suited for small projects. Also, Waterfall model is suitable when the specification and requirements are clearly stated for the software project.

Disadvantages

In Waterfall model, the output of one phase forms the input of the next phase. This concept actually turns as its disadvantage i.e. When a mistake occurs in a particular phase, the same mistakes gets carried over to the last phase. Waterfall model is time intensive process and almost provides little or no option to change user requirements. This model is useful only when the requirements are free zed.

- **The Prototyping Model**

A prototype is a working model that is functionally equivalent to a component of the product. In many instances the client only has a general view of what is expected from the software product [27]. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs and the output requirements,

the prototyping model may be employed.

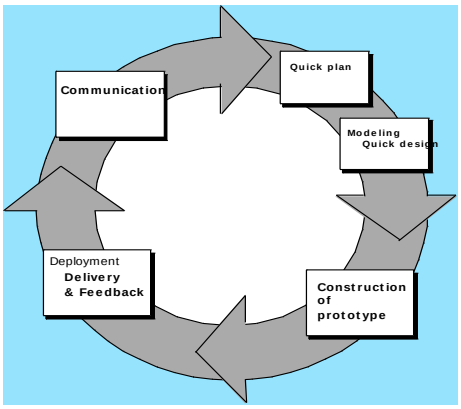

Fig. (b) The Prototyping Model

Advantages

With reduced time and costs, Prototyping can improve the quality of requirements and specifications provided to developers. Prototyping requires user involvement and allows them to see and interact with a prototype allowing them to provide better and more complete feedback and specifications.

Disadvantages

Insufficient analysis: The focus on a limited prototype can distract developers from properly analyzing the complete project. User can begin to think that a prototype, intended to be thrown away, it is actually a final system that merely needs to be finished or polished developer attachment to prototype.

- **The Spiral Model**

The spiral model, also known as the spiral lifecycle model, is a systems development method (SDM) used in information technology (IT) [1]. This model of development combines the features of the prototyping model and the waterfall model. The spiral model is intended for large, expensive, and complicated projects.

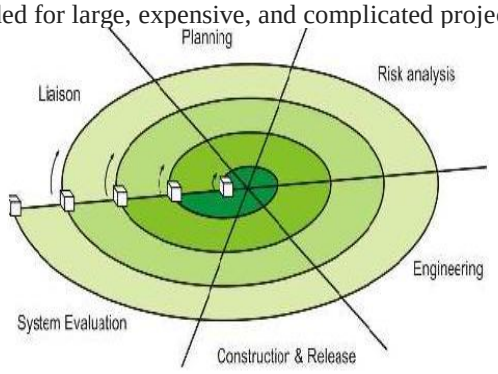

Fig (c) The Spiral Model

Advantages

Estimates (i.e. budget, schedule, etc.) Become more realistic as work progresses, because important issues are discovered earlier. It is more able to cope with the (nearly inevitable) changes that software development generally entails. Software engineers (who can get restless with protracted design processes) can get their hands in and start working on a project earlier.

Disadvantages

As it is highly customized so there is limiting re-usability. It is applied differently for each application. There is a risk of not meeting budget or schedule.

B. Quality Assurance

Everyone is committed to quality; however, the following statement shows some of the confusing ideas shared by many individuals that inhibit achieving a quality commitment: Quality requires a commitment, particularly from top management. Close cooperation of management and staff is required in order to make it happen.

• Many individuals believe that defect-free products and services are impossible, and accept certain levels of defects as normal and acceptable [1].

• Quality is frequently associated with cost, meaning that high quality equals high cost. This is confusion between quality of design and quality of conformance [2].

• Quality demands requirement specifications in enough detail that the products produced can be quantitatively measured against those specifications. Many organizations are not capable or willing to expend the effort to produce specifications at the level of detail required [3].

Technical personnel often believe that standards stifle their creativity, and thus do not abide by standards compliance. However, for quality to happen, well-defined standards and procedures must be followed [1].

Quality cannot be achieved by assessing an already completed product. The aim therefore, is to prevent quality defects or deficiencies in the first place, and to make the products assessable by quality assurance measures. Some quality assurance measures include: structuring the development process with a software development standard and supporting the development process with methods, techniques, and tools. The undetected bugs in the software that caused millions of losses to business have necessitated the growth of independent testing, which is performed by a company other than the developers of the system [8].

In addition to product assessments, process assessments are essential to a quality management program. Examples include documentation of coding standards, prescription and use of standards, methods, and tools, procedures for data backup, test methodology, change management, defect documentation, and reconciliation. Quality management decreases production costs because the sooner a defect is located and corrected, the less costly it will be in the long run [7]. With the advent of automated testing tools, although the initial investment can be substantial, the long-term result will be higher-quality products and reduced maintenance costs. The total cost of effective quality management is the sum of four component costs: prevention, inspection, internal failure, and external failure. Prevention costs consist of actions taken to prevent defects from occurring in the first place. Inspection costs consist of measuring, evaluating, and auditing products or services for conformance to standards and specifications [9]. Internal

failure costs are those incurred in fixing defective products before they are delivered.

IV. PROBLEM STATEMENT

A. Problem Definition

The study of various software development process models reveal that in almost all these models, software testing is included as one phase, but testing is required at each phase and not at a particular stage. The main purpose of software testing is to uncover errors which are not simply syntax errors in code but various other types of errors in all the documents produced during the software development ,e.g. software requirements document, design document, test plan etc. Various types of software testing techniques have been developed till date, but which type of testing technique will be suitable and sufficient for checking a particular document in which phase of software development life cycle (SDLC) is not yet clear. So here the problem is to

1. Identify the testing techniques which can be applied at different levels and phases of software development life cycles

Also, software quality is an essential part of any software project. Various quality assurance and control activities may be used to ensure quality in the software project. Different quality attributes need different types of testing to measure software quality. The problem is that out of numerous testing techniques possible, which testing technique should be applied to measure which quality attribute is not very clear. So the next problem takes here is

2. Identify the testing techniques which can be applied to measure which software quality attribute

B. Justification

By categorizing which type of testing to be applied at which phase of software development will help us plan for testing in that phase efficiently and to take full advantage of all the types of testing techniques to improve quality in that phase and consequently the overall quality of the software project. The relation between various quality attributes and the testing techniques required for each of these will help save time and produce quicker results and streamlined testing of the project for that particular software quality attribute.

This defines the statement of problem. Next we'll describes the Proposed Solutions according to the problem of statement.

C. Proposed solution

According to the problem statement above, a model "Software Development Life Cycle Testing Model" is proposed in which all types of testing techniques related to test all phases of SDLC are specified. V model of testing given by Mr. Perry includes only 5 phases of SDLC. Here this model is extended to include more phases of SDLC and select the types of testing technique that can be applied in each phase.

1. Apply Testing on all Phases of SDLC

It has always been a big question when to start testing. Experts suggest that every step taken in the development of the system must be tested thoroughly in a formal manner. It means that testing must be done for requirements gathering, designing, coding, and even for testing phase. Testing of testing efforts may seem to be unusual and surprising but it is an important effort because one needs to be sure about the testing efforts to be able to rely on its reports. A good testing life cycle begins during the requirements elicitation phase of software development, and concludes when the product is ready to install or ship, following a successful system test. Fig given below shows that testing applied on all the phases (Requirement gathering, Designing, Coding, Testing, Implementation and Maintenance) of SDLC, not a particular stage. The study of various software development process models reveal that in almost all these models, software testing is included as one phase, but testing is required at each phase and not at a particular stage. In this SDLC testing model we applied the testing at all the phases of SDLC. By categorizing which type of testing technique to be applied at which phase of software development life cycle will help us plan for testing in that phase efficiently and to take full advantage of all the types of testing techniques to improve quality in that phase and consequently the overall quality of the software project. Well-defined traceable and controllable processes are required for enhancing the quality of the software products and gaining optimum benefits from applied effort. Software process is a stepwise sequence of activities carried with the focus of producing quality software in an economic manner it will be possible when we applied testing at all the phases of software development life cycle. Software testing is recommended to be started as early as possible in the earliest phases of the SDLC, most preferably in the requirement analysis phase itself and should be performed by skilled testers only and not by developers. Software development life cycle (SDLC) processes involve activities of software requirements, analysis, requirement specification, design, coding, testing, delivery, and maintenance. The testing phase can be used in all of these life cycle phases as an umbrella activity.

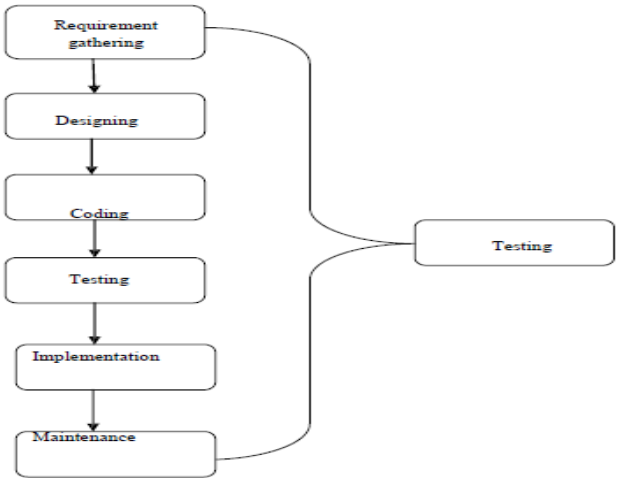

Fig. (d) Applying testing on all phases of SDLC

2. Identifying Testing Techniques according to Phase of SDLC

We identifying that which type of testing technique can be applied to which phase of SDLC. Fig (e) shows the phases of SDLC and according to testing technique.

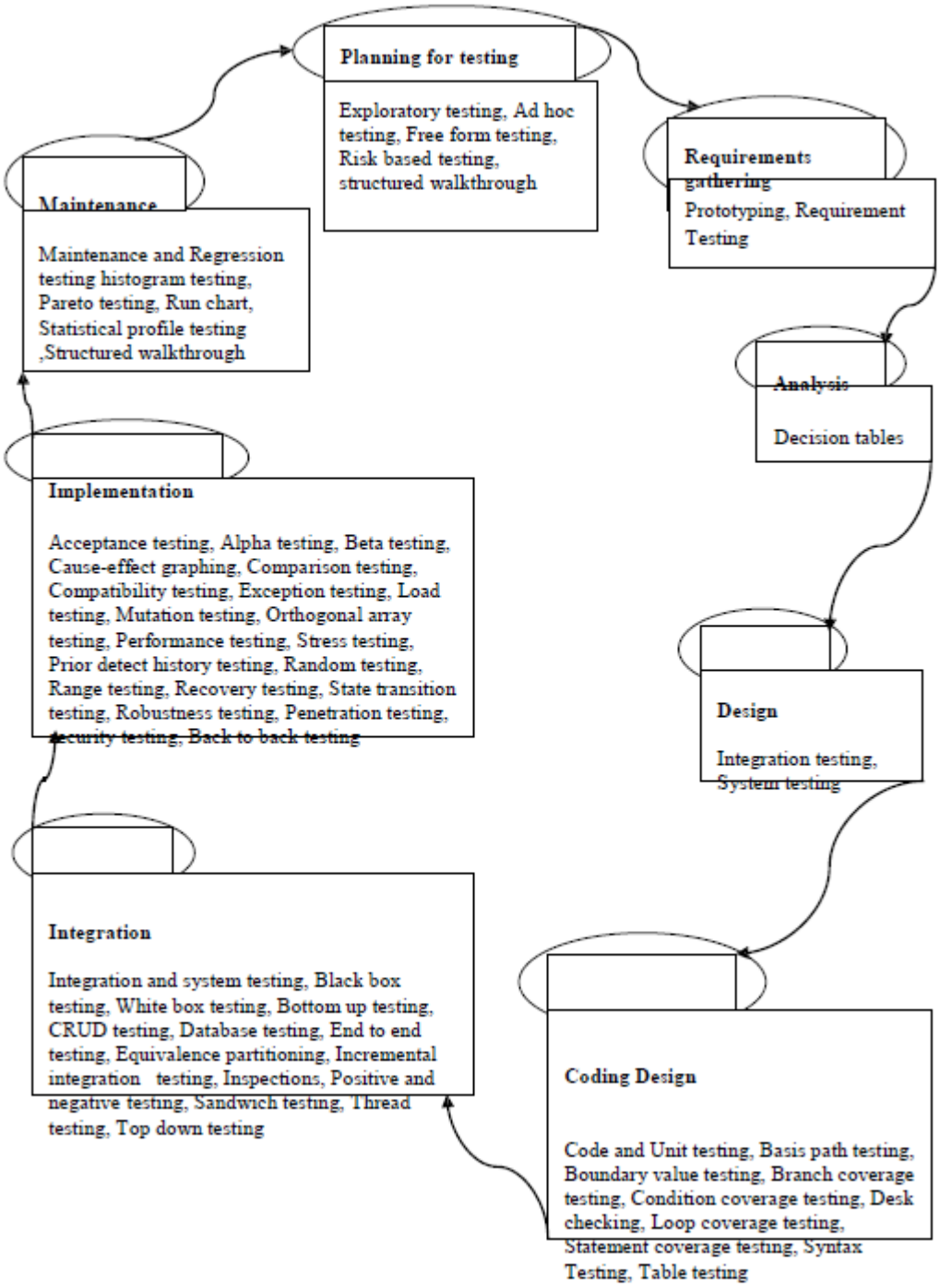

Fig (e) : SDLC Testing Model

3. Application of Testing to Measurement of Quality Attributes

Different quality attributes need different types of testing to measure software quality. Various types of testing according to the quality feature it applies to in the table 6.3. In given table we identified that for a particular software quality feature which type of software testing technique can be applied:

Table 1: Testing Technique According to Quality Features

Quality Attribute	Testing Types
Performance	Performance testing
Compatibility	Compatibility testing
Reliability	Stress testing, Robustness testing, load testing
Vulnerability	Penetrati on testing
Consistency	Database testing, Table testing
Correctness	Database testing, Table testing
Portability	Portabilit y Testing
Recovery	Recove ry testing
Completeness	Boundary/Statement/Loo p/Condition/Path coverage testing

V. CONCLUSION & FUTURE WORK

Software testing is the activity that executes software with an intention of finding errors in it. Testing should be performed at different levels, including module level testing, unit level testing, interface testing and system level testing. Testing is done both at developer end and customer end and it is performed by testers as well as the customer before delivery of the product but it can ensure a fair level of confidence in the predictable behaviour of the product in the provided conditions.

Quality is the main focus of any software engineering project. Without measuring, we cannot be sure of the level of quality in software. So the methods of measuring the quality are software testing techniques. This thesis report relates various types of testing technique that we can apply in measuring various quality attributes. Also which testing are related to various phase of SDLC. General SDLC processes are applied to different type of projects under different conditions and requirements. There are various type of SDLC model (Waterfall Model, RAD Mode, Iterative Model, Proto Type Model, Spiral Model, V-Model, etc). But in all these models, testing is applied after a particular stage and not in all the phases. In this thesis report, it is concluded that testing should be applied in all the phases of SDLC and not at a particular stage. Which type of testing technique can be applied to which type of SDLC phase is also summarized.

Future work for this area will be to take more new coming testing techniques and relating these to the phases of SDLC. This will help taking the maximum advantage of that testing technique. And this will be helpful to conclude that.

VI. REFERENCES

[I] Accessibility Summit. (2006). Public Sector NeedsBetter Guidance On Web Accessibility, E-GovernmentBulletin (Issue 226, 13 November 2006)http://www.ukoln.ac.uk/webfocus/events/meetings/accessibility-summit-2006-11/egovernment- 2006-11-13.php (Accessed August 30th2007)

[II] Alexander, Dey. (2003). How accessible areAustralianusWeb03.http://ausweb.scu.edu.au/aw03/papers/alexander3/(Accessed August 30th 2007)BSI. (2005). PAS 78: Guide to good practice incommissioning accessible websites. British Standards Institute. http://www.bsi-global.com/en/Standards-and-Publications/Industry-Sectors/ICT/PAS-78/ (Accessed August 30th 2007) Carey, Kevin. (2005). Accessibility: The Current Situation and New Directions. Ariadne 44, June 2005. http://www.ariadne.ac.uk/issue44/carey/ (Accessed August 30th 2007)Chisholm, Wendy and Henry, Shawn. (2005). Interdependent components of Web accessibility.Proceedings of W4A at WWW2005: InternationalCross- Disciplinary Workshop on Web Accessibility.New York:ACM.Press.http://doi.acm.org/10.1145/1061811.1061818(Accessed Aug 30, 07)

[III] Clark, Joe.(2006).To Hell with WCAG 2. A List Apart No. 217. http://alistapart.com/articles/tohellwithwcag2(Accessed August 30th 2007)Cooper, Martyn. 2006. Making online learning accessible to disabled students: an institutional case study. ALT-J-Research in Learning Technology, Vol. 14, No. 1, pp 103-115.DDA (2005) Disability Discrimination Act 2005.

[IV] Web Accessibility 3.0: Learning From The Past,Planning For The Future, Nevile, L. and Kelly, B.ADDW08. University of York, 22-24 September 2008. Retrieved February 4th 2009:http://www.ukoln.ac.uk/web-focus/papers/addw08/paper-2/

[V] Contextual Web Accessibility - Maximizing the Benefit of Accessibility Guidelines, Sloan, D, Kelly, B., Heath, A., Petrie, H., Hamilton, F and Phipps, L.WWW 2006 Edinburgh, Scotland 22-26 May 2006.Conference Proceedings, Special Interest Workshops (CD ROM). RetrievedFebruary 4th 2009: http://www.ukoln.ac.uk/webfocus/papers/w4a-2006/

[VI] IMS. (2004). IMS Guidelines for Developing Accessible Learning Applications. Version 1.0 White Paper. IMS Global Learning Consortium. http://www.imsglobal.org/accessibility/#accguide (Accessed August 30th 2007) IWMW 20072007). Contextual Accessibility in Institutional Web Accessibility Policies. http://www.ukoln.ac.uk/webfocus/events/workshops/webmaster-2007/sessions/sloan/(Accessed August 30th 2007) Kelly, Brian, Phipps, Lawrie and Swift, Elaine. (2004).Developing A Holistic Approach for E-Learning Accessibility. Canadian Journal of Learning and Technology, 2004, Vol. 30, Issue 3.http://www.ukoln.ac.uk/web-focus/papers/cjtl- 2004/(Accessed August 30th 2007)

[VII] Kelly, Brian, Sloan David, Phipps Lawrie, Petrie Helen and Hamilton, Fraser. (2005). Forcing standardization or accommodating diversity? A framework for applying the WCAG in the real world. Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A) (Chiba, Japan, 10 May 2005). New York:ACM Press, 46-54. http://www.ukoln.ac.uk/webfocus/papers/w4a-2005/ (Accessed August 30th 2007) Kelly, Brian. (2006). Accessibility of resources in institutional repositories, Digital Repositories mailing List, 18 December 2006. on Aging. (2002). Older adults and information technology: A compendium of scientific research and web site accessibility guidelines. Washington, DC: U.S. Government Printing Office. Nielsen, Jakob. (1994). Heuristic evaluation. In Nielsen,J., and Mack, R.L. (Eds.), Usability Inspection Methods. New York: John Wiley & Sons. Pickard, Jack. (2006). Whistle Stop WCAG 2 : To Hell… and back. Blog post August 2006. http://www.thepickards.co.uk/index.php/200608/whistle-stop-wcag-2-to-hell-and-back/ (Accessed August 30th 2007) Raymond, Eric. (1998). The Cathedral and the Bazaar, First http://www.firstmonday.org/issues/issue3_3/raymond/

Maneela Tuteja,
M-Tech (Computer Science Engineering) from Amity School of Engineering & Technology, Amity Unicersity Noida,
B-Tech (Information Technology) from Kurukshetra University.
Having more than 5 years of professional experience in IT, Services & Education.