Article can be accessed online at http://www.publishingindia.com

Implicit Testing - An Improved Way of Testing Software

Shraddha Avhad Bahalkar*, Shankar Ramamoorthy**

**SW Test Team Lead, Amdocs DVCI Pvt Ltd, Cyber City Tower 2, Magarpatta city Pune, Maharashtra, India. Email: savhad@amdocs.com*

***Quality Business Partner, Amdocs DVCI Pvt Ltd, Cyber City Tower 2, Magarpatta city Pune, Maharashtra, India. Email: Shankar.Ramamoorthy@amdocs.com*

Abstract

This Paper talks about implicit testing in software, Implicit is defined as, "Suggested though not directly expressed, Implied, indirectly suggested; unconditional, contained within." Implicit testing has many dimensions however it is slightly ignored or can say untouched. This Paper is going to explore and get to know it in detail for better usability, get the maximum returns from the software to achieve better customer satisfaction. This paper will talk about the comparison between Maslow's Hierarchy of Needs and Software needs to explain the correlation between human needs and software needs and different ways of conducting implicit testing; for example, Intuitive Testing, Exploratory, Accessibility, Heuristics, Hypothesis and Usability Testing. Life cycles of Software needs.

Keywords: Implicit needs, Implicit testing, Hierarchy of Needs

1. Introduction

In the software context, Implicit needs means needs or requirements which are not mentioned in any specification documentation such as BRD/SRD/HLD. Now, one will ask why one needs to test this "Implicit Need" when it is not mentioned in any specification documentation, absolutely correct!

We have been taught that "Software testing is a procedure of using a software system or application with the concentrating on discovering the software issues/problem or defects", and defect is nothing but "Deviance from the requirement stated in the software system/Application functional requirement document"

So will a good tester always refer to specification documentation (like BRD/SRD/HLD/User Story) and skip testing implicit needs?

Or shall we stop referring to specification documentation?

One should keep using/referring to specification documentation. There is more to test apart from these Documents and that is called as "Implicit Needs of a Customer", if one skips testing it then one is not doing his/her job devotedly. We can say that one is not working on enhancing customer experience.

This technical Paper discusses the Hierarchy of Needs and different types to conduct implicit testing and life cycle of software needs.

2. Comparing two Needs Models

Here, we discuss the Hierarchy of Needs, and compare Maslow's Hierarchy of Needs Model [1] with a Software Needs Model. For a better understanding of the topic, this paper will do a high level study of Maslow's Hierarchy of Needs.

A. Concept of Maslow's Hierarchy of Needs[1] - Abraham Maslow created a model. He explained that elementary needs such as physiological and safety must be fulfilled before superior needs such as love, esteem or self-actualization. In this model, only when an elementary need is satisfied, can the next superior need start to be fulfilled. Maslow's hierarchy of needs diagram follows:

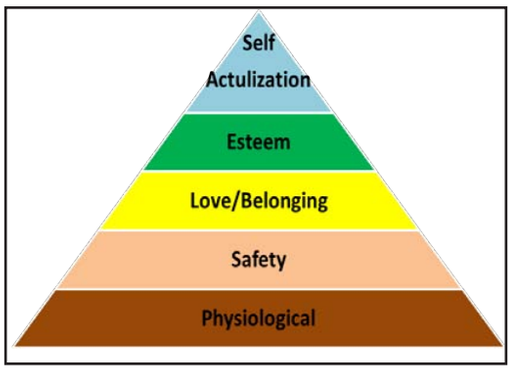

Fig. 1: Maslow's Hierarchy of Needs

Physiological Needs are low level needs which are basics to endure life, such as:

- Air
- Water
- Food /Nourishment
- Sleep

As per Maslow's theory, if low level needs are not fulfilled then one will try to fulfil these needs first. Superior needs like social or esteem needs are not sought after until one has fulfilled the basic needs to one's living.

Safety Needs When physiological needs are fulfilled, then one will start thinking about safety and security.

These needs can be fulfilled by:

- House/ Home
- Medical facilities
- Job/ service
- Financial security

As per Maslow's theory, if safety needs are not fulfilled then one will try to fulfil these needs first. One will not think about superior needs.

Social Needs When one has fulfilled low level physiological and safety needs, only then will one start thinking about superior needs like social needs. Social needs are related to communicating with other people:

- Family
- Friends
- Love

Esteem Needs When one fulfils social needs, then one starts thinking about other superior needs, like esteem needs. Esteem needs are categorized as internal or external. Examples of internal esteem needs are self-esteem and accomplishment. Examples of external esteem needs are social status and respect.

- Dignity
- Accomplishment
- Responsiveness
- Appreciation
- Status

Self-actualization Need is the top most need of Maslow's hierarchy of needs. It is related to one's development as a person. This need is never fully fulfilled; as there are constantly new prospects to grow.

Self-actualization needs include:

- Reality
- Honesty/ Integrity
- Knowledge
- Significance

As per Maslow, only a small proportion of people reach the self-actualization level.

B. Concept of Hierarchy of Software Needs

The Hierarchy of software needs is a model describing hierarchy of needs in the context of software. Project teams (Dev and QA teams) start working on specified needs, which are all documented, and once all needs are met, sometimes on different levels of documentation, then only can they think about higher level needs such as Implicit needs. The hierarchy of Software Needs is shown in the following diagram

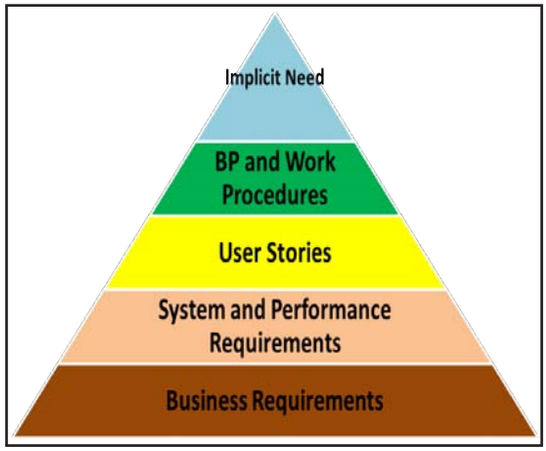

Fig. 2: Hierarchy of Software Needs

Business Requirement Business requirements are nothing but foundation blocks for creating an application or System. A client creates BRs. BRs are provided to deliver value, products, systems, software, and processes to satisfy business requirements of clients. These Requirements are basic and needed to endure the project. Unless we have BRD, the project team cannot think about higher level needs/requirements document. This is a basic need to start with any project.

System and Performance Requirements are more detailed and descriptive requirements. They describe what is expected from an application or Software system to satisfy the client's needs. SRs are written as statements, opinions, and performance requirements; including levels of security, safety, dependability of the application or software system. This is the second most important need to start work on any project.

User Story are requirements from the view of end user, why and how the user is going to use any particular functionality. It describes the user's details, and their expectations from the application. A user story aids to produce a basic explanation of a requirement.

This is the third most important need to start work on any project.

Business Processes are a gathering of correlated, arranged events that produce a precise real life business scenario for the end user. Most of the time, a business process is presented as a flow chart. Methodology and tools to this layer are widely applied in Amdocs[4].

Work Procedures are used to describe the Business Process in detail. This allows clear documentation of how a particular activity should be performed. This can be an optional document to start the project work.

Implicit Needs means needs or requirements which are not mentioned in any specification documentation, like BRD/SRD/HLD/US/DD. There is more to test, apart from these Documents and that is called, "Implicit Needs of a Customer". Once all the stated/specified needs are implemented step by step, the Project team should think about implicit needs. One should start thinking like an end user of the application to know implicit needs better. Implicit testing has a very wide horizon, which includes various types of testing.

Currently, the IT world has become very competitive and Customer Satisfaction is the key to success. As an IT professional, One should always think to pleasantly surprise a customer with delivering something extra (obviously useful) to them. Software quality may defined as explicit as well as implicit requirements and expectations; however some people tend to accept quality as compliance to only explicit requirements and not implicit requirements. This trend needs to change, and the tester needs to give more attention to cover implicit needs of the customer for a higher level of customer satisfaction. For example, a Telecom domain has many new fields apart from Voice and Text. The testers need to explore the new fields and think about the implicit needs of a new world, like better usability, accessibility of the telecom application, such as customer web sites and portals to see usage, pay bill, generate reports, etc.

Digitalization is occurring very quickly in developed countries and it is a need of time to use websites or a portal for day to day work. In India, the Government is promoting Digitalization for a better e-Governance model. In this scenario, there is a huge scope of implicit needs understanding and testing for better usability and accessibility of web applications.

Amdocs always pays more attention to their customer's implicit needs by setting up innovation labs and creates different useful tools/utilities to enhance the customer experience.

Amdocs introduced the BEAT-Testing framework for improved testing processes and customer satisfaction.

This paper discusses the different types to conduct implicit testing, as below:

3. Different Types to Conduct Implicit Testing

In the previous section, the implicit needs were presented as the tip of the testing needs. Although it is a new approach, there are several existing testing methods that are adequate to conduct implicit testing, including:

A. Intuitive testing
B. Exploratory testing
C. Accessibility testing
D. Heuristic testing
E. Hypothesis testing
F. Usability testing

A. Intuitive Testing Intuitive can be defined as "obtained by using your feelings rather than by considering the facts", OR "able to understand something by using feelings rather than by considering the facts"[3]. Now

one must have started correlating the word "Intuitive" and "Testing". Intuitive testing is also known as Error Guessing. This cannot be used as a primary testing technique, however one can find more issues and errors unnoticed by organized testing. Intuitive testing can be used to support and complete the choice of test cases through systematic testing methods. Intuitive testing can be most effective when you have all the needed skills, experience and knowledge in that Domain for which you are testing an application.

Knowledgeable and expert testers are inspired to consider the conditions in which the software may not be able to perform. Testing is a science and an art; few testers are naturally good at it and other testers have good experiences or they have worked with a specific application or domain for a long time, therefore they are able to find many defects/issues. For this reason an error predicting/guessing approach, used after more formal testing methods have been used to some extent, can be very effective. It is also time effective as knowledgeable and expert testers use assumptions and guessing to find out the defects which otherwise they may not be able to find.

An expert tester will know which typical condition system may fail to perform or throw an exception, few conditions like division by 0, incorrect/blank input, blank files and the incorrect input data (these are special characters in input fields or alphabetical characters where digits are mandatory). They will always try to test a condition that is very unlikely to happen in the system/application.

The following factors can be used to guess the errors:

- Lessons learnt from past releases
- Historical learning
- Previous defects
- Production tickets
- Review checklist
- Application UI
- Previous test results
- Risk reports of the application
- Variety of data used for testing.

B. Exploratory Testing – Exploratory Testing is a testing method that permits to apply one's capability and talent as a tester in an influential way. It is testing of software without any precise strategies nor timeframe. Dr. Cem Karner[7]. conceived the term exploratory testing. This is a prescribed testing procedure where the tester does not need any test cases or test planning documents to start testing the system. The tester gets to know the functionality by exploring the system and by exploring and learning the system. Testers can design the test cases and at the same time execute it as well. Testers get to know the system first by exploring it, and based on this understanding, the tester can come up with the test cases/test planning and then start actual testing of the system. One can get better understanding from the below diagram

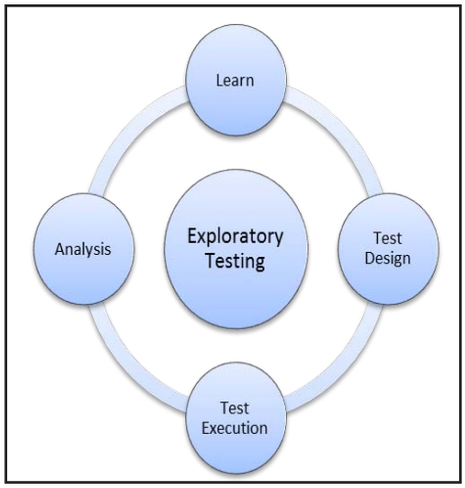

Fig. 3: Exploratory Testing

In exploratory testing, testers have to do the minutest effort for test design, however an extreme effort for execution to get to know the particular functionality of a system. This is helpful for the tester to decide what one can test next. This testing is a kind of on the job training; the tester learns the system behaviour during testing itself and creates a test plan or test cases. In this type of testing, testers have liberty in testing. The tester's experience as well as skill will give the edge for finding more defects in a system/application.

Some tips for exploratory testing methods:

- Find out the scope of the project.
- Identify the requirements along with functionality of the software system/application.
- Check out the constraint of the software system/ application.
- To validate stability of software system, tester should prepare test scenarios
- Comprehensive testing of the software as per the prepared test scenarios and identified requirements.

C. Accessibility Testing – day by day the web is becoming more important in everyone's lives; the web uses e-paper, social media, e-commerce, e-payments, Internet banking and many more. Therefore, the availability of information technology tools becomes important for everyone. Therefore, Accessibility testing is born.

Accessibility Testing can ensure that the system/ application is usable by differently able people, like having impaired vision, having hearing problems, colour blindness, senior citizens and other differently able groups. One can also describe accessibility testing as type of systems testing intended to identify differently able individuals who can use the system, with ease. It can be software system or application, hardware, or some other type of system. Differently able people can have an extensive kind of physical difficulties, which can be learning or understanding issues, problems with vision, problem with hearing and difficulties in movement.

Section 508, an alteration to the United States Workforce Rehabilitation Act of 1973, is a federal law mandating that all electronic and information technology developed, procured, maintained, or used by the federal government be accessible to differently able people.[5]

Many Telecom Service providers have introduced accessibility testing to ensure that the differently able community has full access to modern telecommunications technology.

Differently able people use assistive technology.[6] It can help people in functioning/ using a software product in a better way. For example, products like:

- Speech Recognition Software – this software helps differently able people to convert the spoken word to text that can be given as input to the computer.
- Screen Reader Software – this is used to read out the written text which is displayed on the screen of the computer
- Screen Magnification Software - this is used to magnify/enlarge the monitor/screen to make reading easy for visually-impaired people.
- People having motor control difficulties or movement issues can use special keyboards for easy typing

D. Heuristics Testing-Heuristics testing includes testing of code components, algorithms and projects having testing strategies depend on historical data about possibilities. Heuristics testing mostly allows finding where defects or issues can be found in the application by doing intellectual analysis.

Experts in the field can bring a closer focus on how one can do software testing to get better results of testing, therefore heuristics testing can also be called experience-based testing. Experts like developer and testers make decisions about how software testing can be performed to make the testing process more effective and result oriented. Please refer to the Heuristic model for better understanding. This model was designed by Mr James Bach. [2]

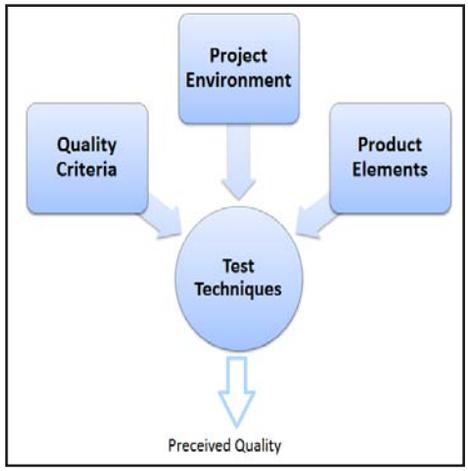

Fig. 4: Heuristic Test Strategy Model

Heuristic Test Strategy Model consists of components that can help in creating a better testing strategy. The ultimate use/purpose of this model is to make tester think about what is needed while creating test strategies. Also, it is proposed to be tailored and used to assist conversation and uninterrupted self-learning for expert testers.

Project Environment includes assets, limitations, restrictions, resources, team and other features in the project which may empower or restrict our testing. The tester should encounter restrictions or constraints, however sometimes the tester needs to agree to these restrictions or constraints. The tester's decision about constraints handling can depend on the situation and requirements.

Product Elements are software features that need to be tested. Software can be composite and indiscernible, so the tester needs to be very careful while creating a test strategy. The tester needs to plan to cover all easy as well as complex elements in a project.

Quality Criteria are the guidelines, standards, and procedures which can be used by the tester to identify issues in the product. Quality criteria can have many magnitudes and measurements.

Test Techniques are procedures for generating the tests. The test technique involves analysis of project environment, product elements, and quality criteria. This is a very important part of heuristic testing as results depend on the test technique used.

Perceived Quality is the outcome of testing. The tester can never know the "definite" quality of a software product, however one can do extensive range of tests for software product/application, to identify the well-versed evaluation of the software product quality.

E. Hypothesis Testing- is a statistical test which is used to identify whether there is sufficient confirmation in sample data to conclude that a particular circumstance/condition is factual for the complete population. A hypothesis test observes two contrasting hypotheses about a population: the null hypothesis and the alternative hypothesis. The null hypothesis is the declaration being tested. Typically the null hypothesis is a declaration of "no effect" or "no difference". The alternative hypothesis is the declaration one wants to be able to determine is factual. From the sample data, the test concludes whether to discard the null hypothesis. One uses a p-value to conclude. If the p-value is less than or equal to the significant level (one can define the cut-off point) then one can discard the null hypothesis. For example, assume one want to conclude, if a coin is in good condition and proportionate. One flips the coin; it may result in half of the time Heads and other half of the time Tails. The alternative hypothesis can be the number of Heads and Tails will be very much dissimilar. Representatively, these hypotheses can be stated as

H0: $P = 0.5$

Ha: $P \neq 0.5$

Assume one flipped the coin 100 times, resulting in 60 Heads and 40 Tails. Assuming this outcome, one may be prone to discard the null hypothesis. One would determine from these results that the coin was probably not in good condition and not proportionate.

F. Usability testing- Usability testing is used to identify up to what extent a desired software application can be understandable, how easy it is to learn, how easy it is to operate and is it user friendly.

The primary focus is on:

- User friendly –application should be easy to use
- Easy to learn – application should be easy to learn and user should get familiar to the system/application very easily.
- User satisfaction - user should be satisfied with the complete experience of using the application/system
- Result oriented - application should perform well and give the correct results to the end user.

The word "Usability" has various magnitudes. It is all about the user's 'experience' during their interaction with an application and their 'feeling' towards it. A structured Usability Test translates this experience/feeling into a Validation Process. While doing usability testing, one should keep the following points in mind.

- **Easy to learn**: application/system should be easy to use for end users. The user should able to learn the application/system easily in the very first attempt.
- **Easy to memorize**: The user should able to remember and use functionality and navigations of the application/system even if the user does not use the application very frequently or has not used it after a long time.
- **Errors**: the application should have proper error handling and user friendly error messages. Even if the user makes an error, one should be able to recover from errors easily.
- **Contentment**: the user should like using the application/system. User experience should be satisfying.
- **Effectiveness**: the user should desire the performance and accuracy from application/system.

G. Other ways to find out and test Implicit needs –One should use different techniques to find out implicit needs, for example:

- Think like an End User
- Role Play
- Check User friendliness
- Check Error messages
- Use logical thinking/common sense
- Have futuristic view

- **Think like an End User** Start using an application forgetting that you are a software professional, try to do common mistakes while data entry, clicking on buttons, wrong navigation.
- **Role Play** –if an application has different types of users, then role play is the best way to validate implicit needs. One can act as if they are an end user with a particular role and start using the application. One may get authorization or role based access issues.
- **Check User friendliness** – Validate if application is user friendly. This means it should be easy to use. One can test Navigation, screen layout, font, colour. Please refer to the Usability testing mentioned above.
- **Check Error messages**- Validate all Error messages; they should be understandable to the end user. Each should not be similar to, "Technical Error", "Error 404", etc. Error messages should be in simple English and easy to understand.
- **Use logical thinking/common sense**- This is the most difficult part of implicit testing. As everyone knows "Common sense is very uncommon". Try to use common sense and past experience to locate issues in the application. Please refer to the Intuitive Testing and Exploratory Testing mentioned above.

5. Life Cycle of Software Needs

What is the Life Cycle of Needs – if one can see the pattern in Needs, one will know that all the implicit needs in early 90's are becoming stated or specified needs. So one can say that at some point implicit needs become stated needs and a new set of implicit needs will be generated. Can we say it is as a Life Cycle of Needs? "Yes", why not! The below diagram will explain it better

An example for a cycle of software needs can be seen in the development of usability testing.

This was never a stated need in the early 90's. It was understood by the vendor and implemented in some projects, in production the end user liked this experience and the end user gave positive feedback to the client. This way the client was also happy, and for future projects, clients will like to have such functionalities. They start mentioning it in their business requirements documents. This is the life cycle of any implicit need.

After a certain time, all implicit needs become stated needs and a new set of implicit needs is generated.

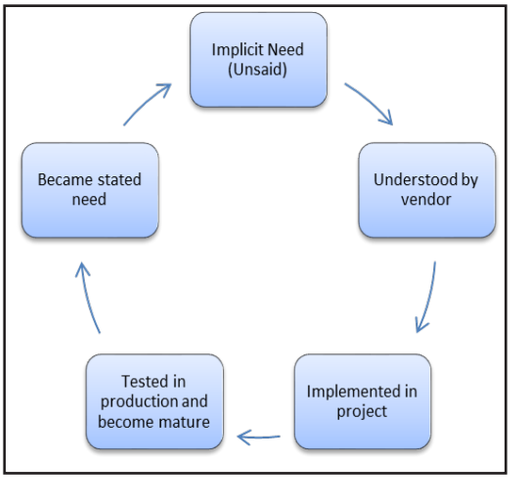

Fig. 5: Life Cycle of Software Needs

One can say that implicit needs are never ending; new sets of implicit needs will become generated as technology evolves.

6. Summary and Conclusion

This technical paper explained about implicit software needs as the top level of software requirements, similar to Maslow's psychology hierarchy of Needs. One cannot avoid implicit needs identification and the testing of it to give a better customer experience. One should always think about implicit needs of the customer and try to fulfil them for a better customer experience.

There are many benefits for testing implicit software needs, including the following:

- Impressing the customer
- Less production defects
- Goodwill building for customer to end user
- Positive marketing for our brand
- Birth of new stated requirements/needs

Although implicit needs is a new concept, there is a large variety of testing methods to target implicit needs.

This paper explained that Implicit needs gives birth to new explicit needs in the software Life cycle, which creates more business for the vendor and a better customer

experience for end users and customers. Implicit needs and its testing is unavoidable and a very important part of today's software world.

Although there are advantages of testing implicit needs, there are also disadvantages in Testing Implicit needs; some are listed here:

- Extension of budget
- No Budget to fulfil customer's Implicit needs
- Misalignment between development and Testing teams
- Misunderstanding of implicit needs
- Limited or no enough time for testing.

7. Acknowledgment

We would like to thank Gilli Shama (Data Scientist) for her guidance and timely feedback. We would like to thank Mr. Leo Paulose (Innovation Expert) for providing support and guidance for technical paper format. We thank Mr Prashant Beniwal and Mr Radhakrishna Pai (Line Managers) for believing in us. We are thankful of Amdocs and TIS management for providing the platform for writing this technical paper. Last but not the least; I (Shraddha) would like to thank my husband Mr. Yogesh Bahalkar and Technical Writer Mr Steven Edell for reviewing and giving feedback for this paper. Thanks to my (Shraddha) Team members, Mr Manikumar (Project Manager), friends and family for keeping me motivated throughout the writing process.

References

Maslow, A. H. (1943). Maslow's hierarchy of needs.

Originally Published in Psychological Review, 50, 370-396.

James Bech (1996- 2015). Heuristic Test Strategy Model James Bach's Blog-, Satisfice, Inc. Version 5.2.2 Date 5/20/2015 Retreieved from http://www.satisfice.com/, http://www.satisfice.com/tools/htsm.pdf

Intuitive -definition –An oxford dictionary

Shama, G., & Szyk, I. (2015). Business oriented testing in the telecommunications Industry. Test magazine, p. 26. Retrieved from http://issuu.com/31media/docs/test_magazine_september2015_web

Section 508 – reference taken from http://www.section508.gov/content/learn/laws-and-policies

Assistive technology reference taken from https://www.microsoft.com/enable/at/types.aspx

Dr. Cem Kaner, J.D., Ph.D. reference taken from http://www.kaner.com/pdfs/QAIExploring.pdf

Brief Bio of Author/s

Shraddha holds MCA post graduate degree and has 10+ years of experience in telecom testing domain. She is working as Test lead in Amdocs and worked with client closely. She is part of Amdocs Elite Expert group. She conducts training sessions for Overview of OSS and BSS at Amdocs DVCI level.

Dr. Shankar Ramamoorthy is currently working as Quality Business Partner in Amdocs Testing services. Dr Shankar is a TQM (Total Quality Management) professional, who has spent much of his 25-years career managing change and improving software Quality Engineering efficiency through Data and Metrics driven Quality Management Systems. He believes in evangelizing, coaching and mentoring test teams through inspirational leadership. Shankar is a computer science engineer from NIT Tiruchirappalli and is armed with a PhD in Software Quality Engineering from IIT Delhi, M. Tech from IIT-Delhi in Technology Management Systems, and an MBA from AIMA-Delhi.