I.J. Education and Management Engineering, **2019, 5, 26-38**
Published Online September 2019 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2019.05.04

Available online at ***http://www.mecs-press.net/ijem***

A Brief Study on Enhancing Quality of Enterprise Applications using Design Thinking

Suman De[a], Vinod Vijayakumaran[b]

[a.] *Post Graduate Student, Dept. of Computer Science & Technology WILP Division, BITS Pilani, Rajasthan, India*
[b] *Professor, Dept. of Computer Science & Technology WILP Division, BITS Pilani, Rajasthan, India*

Received: 01 August 2019; Accepted: 10 August 2019; Published: 08 September 2019

Abstract

With the increase in automation for business processes, the competition among the software vendors have exponentially increased. The goal of empathizing with customer by understanding and delivering a product exactly as the customer demands, has become the primary focus for every software development firm. The lack of interaction between the development team and the end-user has always been the primary reason for software products not adhering to end-user expectations and multiple steps have been taken to address the same.

Design Thinking has been highly rated in terms of understanding the customer requirements and has found its way in terms of how a product is to be developed. This paper looks at various software testing methodologies and how design thinking can impact the same in terms of enhancing the quality and finally delivering the right product to the customer.

Index Terms: Design Thinking, Software Testing, Software Development, Enterprise Application, Quality

1. Introduction

Software Testing is a fundamental practice to enhance the quality of the delivered product and at the same time enhance communication with a duplex feedback flow within the development teams and also the customers. It not only aims at perfecting the delivered code but also ensures that the right boxes are ticked

* Corresponding author.
E-mail address: write2sumand@gmail.com.

in terms of feature delivery as per the customer's expectations. This avoids project failure with respect to bringing clarity on user requirements, having a well-defined quality control and a continuous check on the feature delivery by means of regression. The various phases in the software testing lifecycle starting from acceptance testing, integration testing, system testing, unit testing forms the basis of a final product which is bug free and is resilient to delivery of irrelevant features and ascertain a sense of developing a product keeping the end-user in mind.

Design Thinking on the same lines, drills further into understanding the mindset of the end-user and does a complete behavior analysis of the persona to conclude the actual requirements of the same. It is a methodology used by designers to solve complex problems and find desirable solutions for clients. A design process is more focused towards the solution and not the problem. It is more inclined towards the actions to be performed to address the reality of the business context. Design Thinking is based on logic, imagination, intuition, and systematic reasoning, to explore possibilities of the unknown and to find desired answers that benefit the end user. To have a clear image of what Design Thinking is, Roger Martin, the author of Design of Business, has mentioned, "Design-thinking firms stand apart in their willingness to engage in the task of continuously redesigning their business to create advances in both innovation and efficiency—the combination that produces the most powerful competitive edge." It has been widely observed that when design principles are applied in addressing the creativity and strategy of a firm, the success rate for innovation dramatically improves. Design-led companies such as Apple, Coca-Cola, IBM, Nike and Whirlpool have outperformed the S&P (Standard and Poor) 500 over the past decade by an extraordinary 219%, as per an assessment by the Design Management Institute, run in 2014.

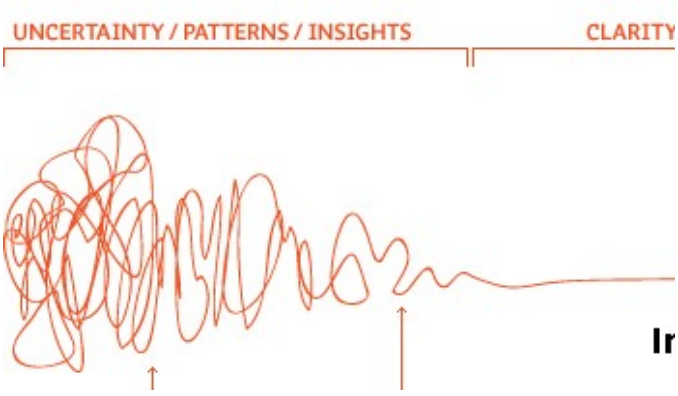

Fig. 1. From chaos to order using Design Thinking

While evaluating various types of products, multiple things are to be considered. For example, the needs of a solution On-Premise is much different from cloud. Test code for cloud products must prove its effectiveness and efficiency daily, just as production code does. With an application-specific test infrastructure, you can fulfill the requirements of high-quality test code, which are complete in terms of readability, adaptability, and efficiency. Having said that, there are practices which remain constant and there is a high possibility in terms of providing a very user-centric approach to testing that facilitates in enhancing and improving the quality of solutions delivered to the customers.

A quality software is one that consists of the least possible number of bugs or defects, is delivered on time and falls within the stipulated planned budget. It also meets all expectations of customers and is maintainable after the same has been released. Design Thinking, as discussed above, provides a very methodical approach

of looking at a requirement specification document and translate it into a manual that helps a quality expert form better test cases and understand scenarios better. With reference to Fig. 1, the journey that starts with a chaotic view of having multiple requirements and problem statements ends with a channelized set of scenarios and test cases that addresses the day-to-day activities of the actual end-users of the solution.

2. Design Thinking in Software Development

There are significant differences concerning the quality of how problems are characterized and approached in design and science. The paradigms of problem-solving in science originated in epistemology, the studies of finding out what is true or not and led to a strong focus on analytical thinking. The paradigms of problem-solving in design though originate in finding out what novel solution fits best in a social or technical system. The discussions on how both paradigms can be assembled together are essential in areas with a strong tradition of dealing with analytical thought process although they form an invaluable part of the design process.

The core of Design Thinking is innovating through the eyes of an end user. It comprises of an on-field study that builds empathy for people who are facing the problem. It is the requirement of the hour to observe what the people need; what technology can do and what is having a higher Return on Investment for the organization. To propose and to develop an innovative solution that really matters to the mass, the mindset of exploring the pain points of the end-user in its work environment is very important. It would be unethical to hypothesize (like legacy software development processes) the people's problems and try to fit in solutions that are irrelevant to them.

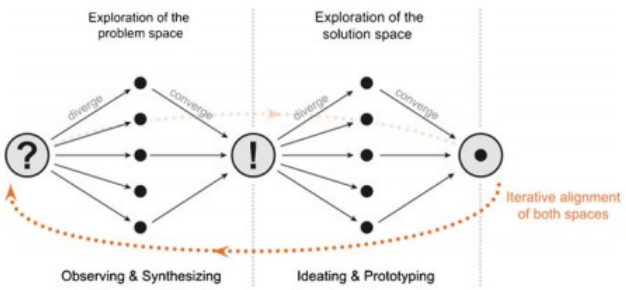

Fig. 2. Problem and solution space in Design Thinking

Hence, Design Thinking is one process that is paving a way towards this endeavor. The five-step framework for Design Thinking is:

1. Empathize – Empathy is the foundation of a human-centric design process. The primary point to be considered is that that the problems which are tried to be solved are rarely of the developer than they are those of users.

2. Define – This part is the unpacking and coalescence of empathy findings and transforms them into conclusive needs and perceptions. It deals with defining a clear and meaningful understanding of what is required to solve the problem. It aims at focusing on the end-user and its context before coming up with a practical and appropriate problem definition.

3. Ideate – The purpose of ideation is to focus on the exploration of multiple solutions for the problem identified that are faced by the user.

4. Prototype – The main aim of developing a prototype is to get ideas and explorations out of the head into the physical world. The most successful prototype developed is the one that helps people or end-users relate to the daily activity they do and will really help them solve their problems.

5. Test – The refinement of solutions and to learn further about the end-users can be performed out in the form of feedbacks or surveys. Testing is the chance to get their opinion on the solutions that will be delivered.

The objective is to explore the capabilities of design thinking for broadening the problem statement and the problem-solving abilities in the software development processes. Comparing to any other industry, the software industry must deal with various programming stack, tools and services and a technical perspective often plays a central role in problem-solving and the solution development. This has become a mandate as the development process and channelized the requirement for highly trained professionals who can deal with complex problems and roadblocks, such as the feasibility of a solution, infrastructure, architectures and so on.

Taking an example of a shopping mall and its customers, designing a more fruitful shopping experience for a value-seeking customer is influenced by observation of the purchasing process of someone that goes to the shopping mall (offline shopping) on a "need to go" basis. The key insight learned is that consumers are busier and have less free time to shop in stores nowadays. The challenge here is to give such data mining capabilities for malls to understand and transform their offline shopping business to a futuristic online shopping experience for shoppers who shop on a "need to go" basis.

Getting a complete understanding of a product's look and feel, whether a solution will work or not, and how the conditions of interaction between user and software shape up, generally influences the ability to communicate about those gaps in technical terms. This is required to address situations that affect every decision about the software design which drills down at the level of architecture or code and, thus, cannot be solved without expert knowledge. Without being said, the recruitment in software industry has a strong tendency to take place within an "exclusive" expert's world instead of the domain expert's world. The word "software design" is, in fact, one of the few design terms that are in high visibility of addressing customer needs and the ideal way forward is to look at software development from a human-centric approach than just a mixture of a handful number of codes performing a significant number of operations and functionalities. Hence, Design Thinking plays a significant part in SDLC and also STLC by focusing on the product from the End-user's perspective.

3.Software Quality

A good quality product or an application is reasonably bug or defect free, released on time, is viable with respect to the involved budget, meets requirement specifications and/or expectations, and is maintainable. This is covered as part of ISO 9000:2015 standards for Quality Management Systems. This International Standard provides the fundamental concepts, principles and vocabulary for quality management systems (QMS) and provides the foundation for other QMS standards. This International Standard is intended to help the user to understand the fundamental concepts, principles and vocabulary of quality management, to be able to effectively and efficiently implement a QMS and realize value from other QMS standards. [13] Key aspects of quality for the customer include:

3.1 Good design

The software should comply to every requirement specification and adhere to the aspects of Usability Engineering at the same time. It is the right blend of functional completeness and ease of use that defines the success rate of an application to the end user. The color coding, font type, size and the styling of texts and other user interface controls are the building blocks of a user-friendly software.

3.2 Good functionality

Along with usability of the software or the application, it's practically important that the functional completeness is kept intact. The behavioral aspects of the software or application should be as per the acceptance criteria agreed upon initially. Deviation from the initial agreement with respect to outcome from the initially accepted behavior is highly unaccepted.

3.3 Reliable

Functional testing often checks against the initial requirement specifications and highlights the ones that has been developed. For a feature, it is necessary that it works for all possible expected inputs and not fail for a certain case. Let's consider a situation for analytics concerning several elements. Such an application should not stop functioning if the number of elements crosses a certain limit unless specified. This aspect underlines the reliability aspect of software.

3.4 Consistency

The primary objective of a software is to ensure that it has a common behavior for a desired situation. The same goes with data as well and every software needs to be made consistent with respect to behavior and data.

3.5 Durable

Durability is almost on the similar lines of reliability. The behavior of software shouldn't break with respect to any change in the usage, security, etc. aspects. The functionality should remain intact.

3.6 Good after release

Once the application or suite is released to customers, the most significant part of SDLC comes into picture, maintenance. The real expectation is to provide relevant sales services to maintain customer satisfaction and improve the same with necessary follow ups. Let's say after using a product for a year, the customer wants to make a few changes to the product. For such scenarios, the modifications should be addressed and completed as early as possible and should be released to the customers on time with quality.

3.7 Value for money

The most important thing is to deliver a product to the customers that has value for money. The application suite should abide by the requirement specifications. It should work as expected and should be user friendly. Providing an additional feature or two out of the initial requirements results in getting an extra mile of trust from the customer and more often than not, the end user. These extra features make the application suite more user friendly and guarantees ease-of-use. Hence, it adds value for money for the customers.

4. ISO 9000:2015 Quality Standards

ISO 9000:2015 describes the fundamental concepts and principles of quality management which are universally applicable to the following:

- organizations seeking sustained success through the implementation of a quality management system;

- customers seeking confidence in an organization's ability to consistently provide products and services conforming to their requirements;
- organizations seeking confidence in their supply chain that their product and service requirements will be met;
- organizations and interested parties seeking to improve communication through a common understanding of the vocabulary used in quality management;
- organizations performing conformity assessments against the requirements of ISO 9001;
- providers of training, assessment or advice in quality management;
- developers of related standards.

ISO 9000:2015 specifies the terms and definitions that apply to all quality management and quality management system standards developed by ISO/TC 176. [13]

5. Software Testing Terminologies

Just like Software Development, there are significant terms in a Software Testing which are needed to be looked at to understand the subject matter better. The intention is to cover the terminologies which will be required to understand the content of this paper.

5.1 Unit Testing

This is the level where individual component or units of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing. Unit Tests should be the first line of code for any development. [14]

5.2 System Testing

System Testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. This test checks if the software is performing as expected post the integration of the units or components.

5.3 Integration Testing

This level of software testing is relevant when the various pieces or modules of the complete end to end software has been developed and unit tested. This ensures that after integration of various modules, the complete software is working as expected and doesn't break even after the submodules are brought together to complete the necessary flow of a product use case.

5.4 Acceptance Testing

Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well. The goal of acceptance testing is to establish confidence in the system. It is most often focused on a validation type testing. We will be exploring further on this when we talk about the proposed idea. [15]

5.5 Regression

Regression testing is the process of testing changes in a software to ensure that the older versions still works with the new changes. This is a normal part of the software development process and, in larger companies, is done by quality experts and testers. The Quality Expert along with the product management develop test scenarios and exercises that will test new units of code after they have been written. All such test cases comprise the test bucket. Before a new version of a software product is released, the old test cases are run against the new version to make sure that all the old capabilities still work.

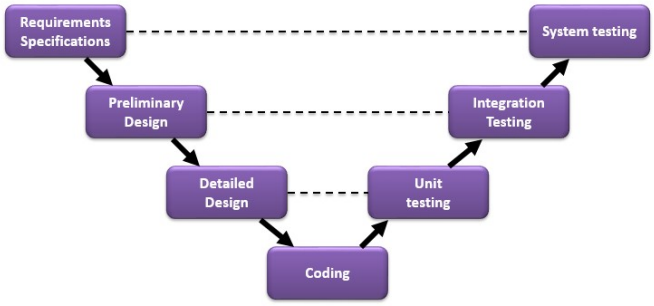

Fig. 3. V-model of Software Testing

6. Phases of Software Testing

Like Software Development Life Cycle, there are multiple phases in a Software Testing Life Cycle which help in the final delivery of a proper product. There is a common tendency to get confused with Software Development Life Cycle (SDLC) with Software Testing Life Cycle (STLC) and we will be covering the different phases associated with the later. Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model. Proactive defect tracking – that is defects are found at early stage. [15] We look at the Verification and Validation (V-Model) model (in Fig. 3) as for this type of testing, the testing of the product is planned in parallel with a corresponding phase of development.

6.1 Requirement Analysis

Requirement Analysis is the very first step in Software Testing Life Cycle (STLC). In this step, Quality Assurance (QA) team understands the requirement in terms of what we will tested & identify out the testable uescases. For conflicts, missing or unclear requirement, then QA team follows up with the various stakeholders like Business Analyst, System Architecture, Client, Technical Manager/Lead etc. to better understand the detailed knowledge of the requirement. [14]

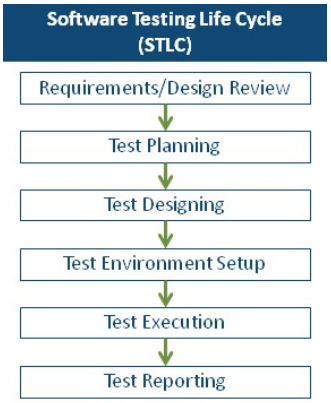

Fig. 4. Software Testing Life Cycle

6.2 Test Planning

Test Planning is the most important phase of Software testing life cycle where all testing strategies are defined. In this phase, the Test Lead or the Quality Expert is involved to determine the efforts and cost estimates for testing the entire project. This phase kicks off once the requirement gathering phase is completed. Based on the requirement analysis, the Test Plan preparation is started. The Result of Test Planning phase will be to devise the Test Plan & do the Testing Effort estimation.

6.3 Test Case Development

The test case development activity starts once the test planning activity is completed. This is the phase of STLC where testing team notes down the detailed test cases. Along with test cases, testing team also prepare the test data required for testing. Once the test cases are ready, these test cases are reviewed by peer members or QA lead. Truth tables are often used to represent and track possible combinations of inputs and their respective outputs as shown in Table 1. [14, 17]

6.4 Test Environment Setup

Setting up the test environment is a vital part of the Software Testing Life Cycle. Basically, test environment relates to parameters, arguments and conditions based on which the test cases will be executed. This is an independent activity and can be started parallelly with Test Case Development.

Table 1. Truth Table to keep track of possible combinations of inputs

Possible cases	While	Case 1	If 1	Case 2	If 2	Case 3	If 3
1	F	–	–	–	–	–	–
2	T	T	T	–	–	–	–
3	T	T	F	–	–	–	–
4	T	F	–	T	T	–	–
5	T	F	–	T	F	–	–
6	T	F	–	F	–	T	T
7	T	F	–	F	–	T	F
8	T	F	–	F	–	F	–

6.5 Test Execution

In this phase, testing team start executing test cases based on prepared test planning & prepared test cases in the previous steps. Once the test case is passed then same can be marked as Passed. If any test case is failed, then corresponding defect can be reported to developer team via bug tracking system & bug can be linked for corresponding test case for further analysis.

6.6 Test Reporting

There are tasks for the closure activities which include the checking for the completion of the test. Whether all the test cases are executed or mitigated deliberately. It also includes checking for no severity 1 defects opened. Post this, a lesson learnt meeting is held and lessons learnt document is created. Let's look at Table 2 for a report with possible test cases and their status. [16]

Table 2. Example of a report with possible test cases and their status

Test Type	Total Cases	Executed Cases	Status	Comments
Functional	100	80	50 pass , 30 fail	
Compatibility	100	50	45 pass, 5 fail	
Usability	100	100	98 pass, 2 fail	
Final Regression	100	100	99 pass, 1 fail	

7. Areas of Quality Enhancements Using Design Thinking

Considering the multiple models and phases followed for software testing, there are multiple opportunities that opens to have a wider view in terms of providing the right set of features that the customer expects. Acceptance testing, for example, has the possibility of reaffirming the customers' and stakeholder's needs and validating the possible solutions even before building them. The objective is to understand the domain or Line of Business better and deliver the right set of functionalities to the user.

In terms of developing a solution, the end users are often being neglected and the word customer is taken into literal terms of the client who pays for the solution. The importance of an end user and their daily activities is often compromised. Next, we investigate various stages from the Software Testing Life Cycle model and the possible usage of a persona-based test plan creation and the evaluation of the accurate business scenarios.

7.1 Understanding Test Cases

The basic problem where most products fail is to miss out on understanding of who it's prospected end-users are. The buyer of a product, especially in the enterprise world, are not the end users. The account manager of a certain sales team understands a product whereas a customer has a clear problem that needs to be solved. The missing link is often the end-user who is supposed to use the product daily and the development team that builds the application. There are multiple assumptions being made and the list of add-ons hamper the delivery of the required functionality as expected by the client.

The solution to this problem lies in the problem itself. In a design led world of design thinking, it is quite

easy to visualize a set of personas who will be using the solution day in and day out. To have multiple interviews with a sample space of such end-users helps in identifying problems and how they currently work. Walking on the same lines, it becomes a chaotic way to identify the real needs but a classification of all their problems often provides a single channel to relate to what they really want. The next steps are to take them through the process via mockups or wireframes which acts as an invaluable channel to get the end-users opinions. An iterative approach of this step of framing wireframes and getting it validated with customers gives a holistic picture of what the customer needs and what the development teams need to address.

For a software testing team or professional, this gives a clear picture of what is expected out of the application. The feedback received during the end-user interviews and persona formation helps create concrete scenarios and use cases that needs to be tested. The general tendency is to look at an end-to-end process defined in requirement specification document and create test cases based on the same. The persona-based model of coming up with scenarios helps understand the various test cases that are relevant for each of them who use the system.

7.2 Understanding Stakeholders and testing relevant functionalities

With respect to the previous topic of understanding and defining better use cases and accurate business scenarios, the personas act as proper stakeholder who are somewhere related to the software. The number of end-users for an application can be directly mapped to the personas evolved from design thinking. Covering test cases for all such personas makes it easier to define a finite number of test cases that are finite and relevant in terms of the solution.

There are multiple benefits of using this process. One being the relative way to test all the required functionalities of a solution and check whether it is in sync with what a persona is affected by. The second affects the effort put by a team in terms of writing test cases, which sometimes is just for the sake of covering each line of code and exceeds the exact number of cases to prove that the code is stable. This approach makes it convenient for a management to measure and calculate the amount of Personal Days (PDs) required by the team to test all the functionalities and hence, plan accordingly.

7.3 Understanding end-to-end application to support Manual Testing

With code bases turning into mammoth sized files, the need for manual testing in terms of end-to-end functionality is becoming a mandate. Regression becomes a mouthful word for projects which depends on the code base of multiple teams and results in a confusing situation. With multiple integration scenarios being involved, the closure of a system is often affected by the change made in a component. The availability of a persona-based approach helps a Quality Expert to follow the same and rerun the tests and check for discrepancies, if any. At the same time, certain manual testing tasks can be automated and should be run through scripts as part of build jobs. The same is explained in the upcoming section with respect to user behaviour.

7.4 Automation with respect to user behavior

With operations and delivery coming into the primary run of thoughts, the automation of scripts while transferring code changes and bug fixes to the customer system takes a center stage. The objective is to have apt scripts that checks the functionality of an application while migrating from a development system to a customer landscape. The generation of automated test scripts as per the current behavior will be a major boost in terms of validating existing scripts. There are multiple organizations which are in due course of coming up with multiple scripts that help reduce human workload. And a script based on end-user behavior not only will help evaluate existing use cases but will help operations and delivery team to resolve issues faster considering

common behavior mistakes and typical human errors while development. The generation of recording customer behavior in a customer system is something that needs more evaluation but will go a long way in terms of understanding the end-user and testing the code with respect to the real-time actions.

8. Future Work

With the above thoughts going into the discussion, the next steps would be to dive further into developing a model that takes in user specifications and identifies the persona and generate relevant business scenarios that needs to be tested. The focus should be bringing in best practices to churn out scripts that serves the purpose of checking the right set of functionalities of the solution. Considering cloud development and a huge focus on Agile Software Development techniques, like Scrum and Kanban, the usage of Agile testing keeping the Design Thinking aspects in mind can be looked at. Further study is required to evaluate the best practices available in the industry and how design thinking can help bridge the gap between a customer and a development/testing team. The rise of Artificial Intelligence and Machine Learning is also overseeing multiple researches and is also a possible field of research when it comes to using the same for Software Testing.

9. Conclusion

Testing a software with the right Functional knowledge and to make it technically bug free is a challenge. The way to reduce and identify the bugs early and a robust testing of the software can be done through Design Thinking. This paper emphasizes on providing relevant test cases and understanding the correct business scenarios that are impacted by a software and hence helps the tester to test the software or product in a better way. The above study also hints at a fact that software development and testing activities go hand in hand as the evaluation of personas supports both the development teams to develop the right software and the testing teams to check whether the right software has been developed or not. The quality of a software is most impacted and having a design led process enhances the same.

As mentioned earlier, a quality software is one that consists of the least possible number of bugs, is delivered on time and falls within the stipulated planned budget. The future works for this current field of study has the possible inclusion of Artificial Intelligence based techniques as well as Agile Testing procedures. The involvement of both improves the way we investigate STLC today, but the objective remains the same to understand the requirements better as a tester. To achieve the same, it is most important to understand the psychology of the end-user and the client and provide the best possible solution that helps them solve their business problems. And, Design Thinking is a great approach that helps enterprises deliver quality software to its customers by empathizing towards their actual needs.

References

[1] Emanuel F. Coutinho, George Allan M. Gomes, M.L. Antonio Jose, “APPLYING DESIGN THINKING IN DISCIPLINES OF SYSTEMS DEVELOPMENT”, Telematics and Information Systems (EATIS), 2016 8th Euro American Conference, 10.1109/EATIS.2016.7520123, Cartagena, Columbia.

[2] Tilmann Lindberg, Christoph Meinel, Ralf Wagner, “Design Thinking: A Fruitful Concept for IT Development”, H. Plattner et al. (eds.), Design Thinking: Understand – Improve – Apply, Understanding Innovation, DOI 10.1007/978-3-642-13757-0 1, c Springer-Verlag Berlin Heidelberg 2011.

[3] Peter Newman, Maria Angela Ferrario, Will Simm, Stephen Forshaw, Adrian Friday, Jon Whittle, “The Role of Design Thinking and Physical Prototyping in Social Software Engineering”, ICSE '15 Proceedings of the 37th International Conference on Software Engineering - Volume 2, Pages 487-496,

Florence, Italy — May 16 - 24, 2015.

[4] Olga Liskin, Christoph Hermann, Eric Knauss, Thomas Kurpick, Bernhard Rumpe, Kurt Schneider, "Supporting Acceptance Testing in Distributed Software Projects with Integrated Feedback Systems: Experiences and Requirements", 2012 IEEE Seventh International Conference on Global Software Engineering, Puerto Alegre, Brazil, pp.84-93, 2012.

[5] Geir K. Hanssen, Børge Haugset, "Automated Acceptance Testing: a Literature Review and an Industrial Case Study", Agile 2008 Conference, 978¬0¬7695¬3321¬6/08 $25.00 © 2008 IEEE DOI 10.1109/Agile.2008.82.

[6] Ken Pugh, "Introduction to Acceptance Test-Driven Development", New Objectives Essential White Paper.

[7] Grigori Melnik, Kris Read, Frank Maurer, "Suitability of FIT User Acceptance Tests for Specifying Functional Requirements: Developer Perspective", Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.296.5178&rep=rep1&type=pdf, Time Accessed: 29.05.2018.

[8] Dudekula Mohammad Rafi, Katam Reddy Kiran Moses, Kai Petersen, Mika V. Mantyla, "Benefits and limitations of automated software testing: Systematic Literature review and practitioner survey", Automation of Software Test (AST), 2012 7th International Workshop, 10.1109/IWAST.2012.6228988, 2-3 June 2012.

[9] Sarbjeet Singh, Sukhvinder singh, and Gurpreet Singh, "Software Testing", International Journal of Advanced Research in Computer Science, 1 (3), Sept –Oct 2010, 403-406.

[10] Archana Magare, Madonna Lamin, "Cognitive evolution in software development life cycle through Design Thinking", COMPUTER MODELLING & NEW TECHNOLOGIES 2017 21(3) 31-34.

[11] Peter Newman, Maria Angela Ferrario, Will Simm , Stephen Forshaw, Adrian Friday, Jon Whittle, "The Role of Design Thinking and Physical Prototyping in Social Software Engineering", Available: http://eprints.lancs.ac.uk/73487/1/ICSE15_SEIS_10_008_Newman.pdf. Time Accessed: 29.05.2018

[12] José P. Miguel, David Mauricio, Glen Rodríguez, "A REVIEW OF SOFTWARE QUALITY MODELS FOR THE EVALUATION OF SOFTWARE PRODUCTS", International Journal of Software Engineering & Applications (IJSEA), Vol.5, No.6, November 2014.

[13] International Organization for Standards, "Quality management systems -- Fundamentals and vocabulary", Available: https://www.iso.org/standard/45481.html, 03.120.10 Quality management and quality assurance 01.040.03 Services. Company organization, management and quality. Administration. Transport. Sociology. (Vocabularies) 03.100.70 Management systems.

[14] Software Testing Life Cycle (STLC), Available: http://www.softwaretestingclass.com/software-testing-life-cycle-stlc/, Date Accessed: 12 August, 2019.

[15] Try Quality Assurance, Available: http://tryqa.com/what-is-software-quality/, Date Accessed: 12 August 2019.

[16] Test Coverage in Software Testing, Available: https://www.softwaretestinghelp.com/test-coverage/, Date Accessed: 12 August 2019.

[17] James A. Whittaker, "What is Software Testing? And why is it so hard?", IEEE Software, Available: https://profinit.eu/wp-content/uploads/2016/03/HardSwTesting.pdf, Date Accessed: 12 August 2019.

[18] M. Palacin-Silva, J. Khakurel, A. Happonen, T. Hynninen and J. Porras, "Infusing Design Thinking into a Software Engineering Capstone Course," 2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T), Savannah, GA, 2017, pp. 212-221. doi: 10.1109/CSEET.2017.41.

Authors' Profiles

Suman De is a student at BITS Pilani and is in his 4th year of his MTech. degree in Software Engineering. He has also been working at SAP for 4 years now and is currently designated as an Associate Developer for Enterprise Project and Portfolio Management Team (Project Systems). He has also worked in Tech Core Gateway & S4 Master Data Management, followed by a phase as a Solution Architect for startups. He has experience of working with SAP UI5, Node JS, JavaScript and is a full stack developer on SAP Cloud Platform with focus on Fiori elements and Cloud Foundry. He is an enthusiastic scholar with 2 International journal papers to his name and has worked as the team lead in couple of projects during his stay at SAP Labs India.

Dr. Vinod Vijayakumaran has 15 years of IT industry experience and currently employed as Technical Project Manager with HANA Enterprise Cloud, Partner Centre of Expertise, at SAP Labs India Pvt. Ltd. He holds doctorate from Karunya University, Coimbatore in the field of Computer Applications. His research area is on processes and this thesis was on a combined software development model incorporating LEAN, Agile and Six Sigma methodologies. Vinod holds a patent from USPO on his tool 'Downtime Calculator'.

How to cite this paper: Suman De, Vinod Vijayakumaran, " A Brief Study on Enhancing Quality of Enterprise Applications using Design Thinking ", International Journal of Education and Management Engineering(IJEME), Vol.9, No.5, pp.26-38, 2019.DOI: 10.5815/ijeme.2019.05.04