IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2320-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 10, Issue 6, June 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.512

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 6, June 2021 ||

| DOI:10.15680/IJIRSET.2021.1006358 |

Enhancing Traceability using Software Development Lifecycle Framework – A Review

Nikitha A P[1], Sunil R Yalamalle[2]

U.G Student, Department of Industrial Engineering and Management, RV College of Engineering, Bengaluru, Karnataka, India[1]

Assistant Professor, Department of Industrial Engineering and Management, RV College of Engineering, Bengaluru, Karnataka, India[2]

ABSTRACT:Many software development procedures need traceability of software artifacts, as it is an essential component long regarded. Traceability aims to increase software systems' overall quality. Traceability improves system acceptability by giving a better and consistent knowledge of the system to the user. Research shows that traceability is helpful, but constructing and maintaining an automated traceability software are costly. There are not many studies have shown, however, that it saves time to develop and use traceability linkages over different software items during the software development stages. The first part of the paper introduces to the traceability topic. The second section examines manual traceability problems in the software artefacts of embedded systems. The third component of the paper provides software-development-lifecycle (SDLC) frame serves to propose a solution for the conception of tools to solve the problems identified in manual traceability. In last section, a generalized cost estimation for developing a traceability software is provided.

KEYWORDS:Manual traceability, automated traceability, software-development-lifecycle, cost estimation

I. INTRODUCTION

The capacity to describe the life of a need, whether forward or backward - from demand to code, and vice versa - is known as "traceability". Throughout the development cycle, traceability helps you to monitor work items such that the code requires them to be implemented so they play an important function in your organization's control structure. Software artifact traceability was recognized as a vital characteristic in the development of a number of software systems activities. Traceability is usually aimed at improving the software system's quality. In the realm of embedded systems, the current practices have to be analyzed first and questions need to be identified to enhance the current situation.

Among the development artifacts used to monitor requirements are test cases or test runs and problems. The source of a need like a stakeholder or a duty to regulate compliance is traced backward. The requirements' traceability is to check if they have been complied with. The development process is also speeded up. This is because your criteria are easy to understand. Traceability for analysis is also necessary. Traceability may be utilized for the evaluation of the impact of a modification if a requirement changes. What the problem is you're going to figure out. You can observe how changes to this criterion also influence other problems or tests.

The complexity of software systems has evolved to a degree without precedent. This has led to extra difficulties for the system builders, but also to new opportunities. These problems exist on all levels of architecture and throughout the life cycle of a system. This article suggests a common procedural framework for describing software engineering's life cycle of the human system. Software Engineering is a cross-disciplinary approach and process in order to develop successful software systems.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 6, June 2021 ||

| DOI:10.15680/IJIRSET.2021.1006358 |

The documentation needs and synthesizing and validation of the system are underlined while studying the entire challenge early in the development stage. It brings together all disciplines and organizations in a coordinated effort forming a systematic development process from ideas through manufacture to maintenance and operation. It takes into account all parties involved in both commercial and technical demands in order to provide a high-quality product that fulfils the demands of users and other stakeholders. Concepts are extended in this life cycle until a system is removed. It covers the purchasing and distribution of systems. It aids in improving communication and cooperation among individuals who build, use, and maintain current software systems in a coordinated, integrated manner. Furthermore, this framework ensures the evaluation and improvement of life-cycle activities.

In section 1 the traceability issue is introduced in the software release. The second section discusses the deficiencies with which software testing may be manually traced. In Section 3, a software development lifecycle architecture is provided with those difficulties as a starting point to design a solution that automates the traceability process. To convey these solutions to industry, it is necessary to assess if the ideas truly function in practice, while offering conceptual answers. This is also addressed in the area where we design an embedded systems traceability management solution based on the needs of industry experts. Our future study will target the objective of strengthening traceability processes. The last part covers the method of cost estimates for the development of a program and demonstrates a calculation of the breakeven sample which compares the manual cost of traceability.

II. LITERATURE REVIEW

Gotelet al.[1] define generic traceability as prospective traceability with the stated triple of items consisting of source artifact, destination artifact, and trace connection between the two artifacts. "It has been a mistake. This specification refers specifically to the use of trace links to make links only when used. The lack of this definition is that they are recursive, because they define traceability with the term traces and trace objects with the word traceable. Traceability is a traceable phrase. traceability of the traceable object.

The ability of Spanudakis and Zisman to match artefacts created in software development[2] to depict the system from different perspectives and levels of abstraction, the stakeholders involved in the construction of these artefacts, and the justification for the artefact form is another definition of traceability. As with the preceding definition, the use and use of traceability links are also emphasized; the rationale of artifacts, for example, maybe discussed. But using the description explicitly conveys the sense that these are the only traceability applications, whereas there are plenty more available.

Sujit Kumar Dora's [3] study indicates that software management methods which assess software development through the so-called software development model life cycle. The methodology for the SDLC Program Development Life Cycle (SDLC) means that the software fulfils stated requirements. These techniques place the software development process under different degrees of discipline to make the process more efficient and predictable. This article reviews and explains the heavyweight (traditional SDLC) & the lightweight (agile SDLC) approach and compares the two techniques predictably.

Alexandros Mouzakiti et.al [4] research aims to demonstrate a method to the testing, on Jaguar Land Rover, with the use of ECU diagnostic software (HIL) hardware systems (JLR). Electronic automotive control units (ECUs) are increasingly advanced in embedded software for a comfortable and flexible system test environment. On-site diagnostic software is a major portion of ECU software, making efforts to assess the validity of ECU software time and personnel exorbitant. Tests utilizing real cars typically lead to an on-board diagnostic software testing approach.

P.C. Nissimagoudar et.al [5], a research offers a clear idea of the integration into the automotive electricity channel of these specialized industry developments. The course focuses at realizing the characteristics of an automobile application built using model-based design in HIL-simulation technology (MBD). This methodology enables the design and environmental models of algorithms to be converted into real time applications, connecting to the actual system on specialized real time target hardware. The student learns how to systematically construct and perform test cases for the models or components of his automobile sub-system, check system requirements and may learn to test their ideas in every phase of the design process, including real time testing. The outcomes are assessed by the ability of the students to prototype the industry quickly for their pupils.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 6, June 2021 ||

| DOI:10.15680/IJIRSET.2021.1006358 |

The Navita research [6] describes five SDLC models, including waterfall model, iterative model, v model, spiral model, prototype model. A well-defined and methodical methodology is the software development life cycle (SDLC). It has been used to construct a stable software system of high quality. A lot of SDLC models exist. There are pros and downsides to each development plan. The objective of the presentation of this document is to explore the advantages and disadvantages of software development life cycle (SDLC) and SDLC.

The formalization of the problem of synchronization with co-simulation and hardware in the loop is presented by Van Hoa Nguyen et al[7]. We give synchronization and real-time needs that are properly positioned and synchronized with the formula for co-simulation / hardware in the loop environment. Moreover, the conventional hardware in the loop is easier to utilize in real-time to make it easier. The proposed formulation is displayed in a microgrid, with a secondary control and remote-control integration cross-infrastructure implementation in an intimate co-simulation framework in real-time.

III. TRACEABILITY ACTIVITY

Traceability refers to the ability to express and follow the life of a requirement in both a forward and backward direction - from requirements to code and vice versa. Traceability, when used correctly, allows you to track work items through the development lifecycle, all the way to where requirements are implemented in code, and it plays a critical part in your organization's control system.

The traceability of applicable software development-life cycle (SDLC) criteria is strongly linked to traceability. This ensures that the finished programme meets all standards and is error-free. Traceability is a prerequisite for effective software demand management. Of course, software contains flaws, but traceability may aid in identifying, correcting, and limiting these flaws. Overall, traceability promotes the time and on-budget delivery of company-specific software. Criteria for traceability are an important role in application development.

In practice, several actions are related to traceability. The Development Organization must design the linkage, how it is produced, how it is maintained, how it is used, and how quality is monitored. All these operations must also fit within the company's existing development process, whether agile or planned. It is crucial not only that the organization invests in designing the traceability process for these activities to be performed properly but also that instruments be accessible to assist these operations. This section provides an overview of traceability operations as well as their interdependencies.

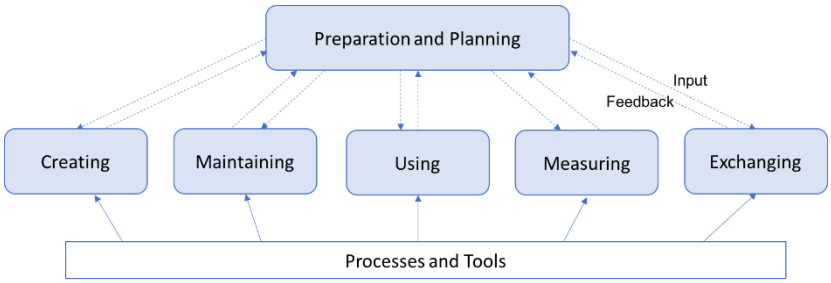

Figure 1: Traceability Model

The traceability model is constructed as shown in Figure 1 to aid understanding of operations. This notion is based on Gotel et al's generic approach to traceability [1], which has been further utilized to explain traceability activities in various software research projects. The model includes four model activities, in which traceability is planned, created, maintained, and used, plus two further activities identified during our study. The model incorporates three activities.

IV. MANUAL TRACEABILITY

Manual reporting places pressure on individuals to always be right in all details, since humans are not flawless, whatever much we desire. The level of service with manual systems depends on individuals and hence management needs ongoing training to be delivered to employees to guarantee that they are motivated and follow the right processes. It can all too easily be changed accidentally and result in a data entry or in hand-written orders that are not consistent. It can be timely and expensive to report and verify that data is robust. This is frequently an area in which

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 6, June 2021 ||

| DOI:10.15680/IJIRSET.2021.1006358 |

large sums may be saved through automation.Traceability is a critical for development of any software. If any of testing feature is neglected or malfunctions, it might have an impact on the functioning of whole software in the future. In software, the characteristics of powertrain components are examined and evaluated. Any bugs discovered are reported and thoroughly investigated. The following points summaries the manual traceability short comes:

- Manual traceability costs may exceed their worth since they are quite time-consuming.
- It is tedious to follow the modifications that the customer has requested and to update the folders.
- Poor documentation immediately reflects product quality since misunderstandings might arise between the customer and the company.
- As the test cases for the vehicle are carried out and reported manually, mistakes are recorded and the vehicle testing scripts will become misleading.

V. AUTOMATION GENERATION OF TRACEABILITY RELATIONS

The study proposes an SDLC model framework for creating a tool that improves the traceability process. In the context of the software development lifecycle, traceability involves the capacity to assure that:

- All requirements have been defined.
- Accepted requirements are divided into development and testing tasks, including cross-references.
- Accepted requirements are broken down into development and testing tasks, complete with cross-references.
- Changes are tracked at any point in the development lifecycle, and cooperation is assured.
- Testing is completed, and the release is ready for deployment on schedule.

51 REQUIREMENT PHASE

The requirement is the first stage of the Software-Development-lifecycle. This phase lays out the broad scope of the project, as well as anticipated challenges, opportunities, and directions. At that stage, the planning and recognition of the dangers related to the quality assurance standards take place. This assists the organization in calculating the time required to complete system work. Teams must satisfy specified and detailed standards at this level.

The goal is to transform the system owner's user-oriented vision of desired capabilities into a technical vision of a solution that meets the user's operational expectations. This process generates a set of measurable system requirements that specify the system's characteristics, attributes, and functional and performance criteria from the standpoint of the provider in order to fulfil stakeholder expectations. The requirements should not include particular implementation to the extent that limitations permit. Underneath the exemplary requirements, the current system and what the system owners expect are compared.

Outcomes:

As a result of the System/Software Requirements Specification Process being carried out effectively:

- All systems and services required to develop system/software requirements are available.
- A system description or element description is established for the system solution, which includes interfaces, functions, and boundaries.
- There are identified critical performance measures.
- The needs of system/software and design limitations are established.
- The system/software needs are traceable to stakeholder requirements.
- The needs of the system/software are evaluated.

5.2 ANALYSIS PHASE

The following stage is the analysis phase in the SDLC phases. In the phase, the query is highlighted which in the previous phase is not clarified. This phase contains all that should be created over the life cycle of the project. The answers to the following questions have not been included in the review and analysis of requirements–

- Do the automating tool support multiple languages?
- How frequently the tool is expected to be used?
- How to design the tool so that the use can be generalized across different teams?
- How frequent updates are required?

The system analysis process aims to offer rigorous facts and information to help decision-making throughout the life cycle, for technical comprehension. The process of system analysis covers the generation of information required for any technological evaluation. It can ensure that system requirement, architecture and design have trust in its utility and

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 6, June 2021 ||

| DOI:10.15680/IJIRSET.2021.1006358 |

integrity. The outcome of the systems involves a wide variety of analytical functions, complexity, and rigor levels. It's used to address a wide range of analytical problems, including determining requirement values, resolving conflict requirements, evaluating alternative architectures or system pieces, and evaluating engineering strategy. The formality and rigor of the analysis are determined by the importance of the information demands or supported product, the amount of data/information available, the scope of the project, and the timeliness of the results.

Outcomes:

Following efficient implementation of the system analysis:

- The findings of system analysis for decision-making are offered.
- The system analysis that is necessary is identified.
- There are all enabling systems or services required for system analysis.
- The system analysis' assumptions and conclusions are confirmed.
- The outputs of the system analysis are traceable.

5.3DESIGN PHASE

In line with the requirements documentation, the software design system is prepared in the third stage. This allows the architecture of the whole system to be defined. This design phase is the next step of the framework. The goal of the architecture definition process is to develop system architectural alternatives, choose and articulate a set of consistent points of view one or more alternative(s) that address the interests of stakeholders and fulfill systems requirements. The architecture of the utility sync resync has the following design:

Outcome:

The outcomes of the Design phase's effective implementation:

- Concepts, behaviours, qualities, features, functions, or limitations that are crucial to system design decisions are assigned to architectural entities.
- Perspectives on architecture are being produced.
- The system's context, limits, and external interfaces are specified.
- The architecture addresses the concerns of stakeholders.
- The system's architecture views and models are created.
- The components of the system, as well as their interactions, are identified.
- The architecture is aligned with the needs and design criteria.
- Providing an architectural framework for processes throughout the life cycle is also conceivable.
- Traceability of architectural aspects is built according to stakeholder and system/software needs.
- Any systems or services that are necessary for the definition of architectures are available.

5.4DEVELOPMENT PHASE

The next stage is the design and completion of the tool algorithm. The algorithms are designed for breaking work into units or modules and assigning them to the various developers throughout this phase. This is the longest stage in the software-development-lifecycle. During this phase, developers must follow precise standards of coding. The programming tools, including compilers, interpreters, debuggers, should also be utilized to implement and develop code. The design definition process' goal is to provide sufficient exact data and knowledge about the system and its components so that architectural entities may be implemented according to models and system architecture viewpoints.

Outcomes:

As a result of the successful execution of the Design Definition process:

- There are specified or refined interfaces between system parts forming the system.
- Each system element's design features are defined.
- The design enablers that are required for design definition are chosen or defined.
- The architectural bodies of the system architecture are traced to the design features.
- The design decisions for system elements are assessed.
- Creation of design artefacts.
- Any design definition enabling systems or services are available.
- The different system components are allocated the system/software requirements.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 6, June 2021 ||

| DOI:10.15680/IJIRSET.2021.1006358 |

5.5TESTING PHASE

Software testing is the process of looking at a software program to see if there are any differences in the input and output. In order to assess the A software item's functionality. Testing is used to assess the product's quality. Software testing is a development procedure that should be performed. Software testing is in other words a method of verification and validation. Enhanced software quality, better users and cheaper maintenance costs, and more precise and dependable findings will help to effective testing. During this stage, the test team will start to test the complete feature of the system. This is done to ensure that the entire application works as expected by the customer. During this phase, test teams might detect bugs/faults communicated by developers. The development team corrects the re-testing problem and returns it to the test team. All stages are watched to verify if the requirements have been met, and the comments are documented in an excellent Microsoft feature.

Outcomes:

As the verification procedure has been successfully carried out:

- All required systems and services for verification are available.
- Verification limitations that influence the requirements, architecture, or design are discovered.
- Information is reported about the provision of remedial action.
- Objective evidence of the actual system complying with requirements, architecture and design is provided.

5.6DEPLOYMENT PHASE

Once the software testing process is completed and no faults or flaws remain, the final deployment phase begins. Based on the project manager's comments, the final program is created and tested for any deployment issues.Operational expenses monitoring or minimizing while maintaining acceptable or improved levels of service are the key purpose of the operation. Although software systems can be specifically infrastructure in scattered contexts, they are generally utilized to activate other software systems and services. The safety, availability, and operating efficiency of the program system in a larger system are therefore concerned. There may also be coordination between current, ongoing, or ongoing services supplied by other systems that provide the same or equivalent services.

Outcomes:

As a consequence of the operating procedure successfully implemented:

- Any operation-specific systems, services and materials necessary.
- Restrictions are discovered which affect the needs of systems/software, architecture or design.
- There are trained, competent operators.
- The performance of the product system is checked during operation.

VI. PROJECT COST ESTIMATE

When an example of the new automated traceability tool is developed by following the software-development-lifecycle framework suggested in the paper a rough estimation can be calculated taking the following parameters. The parameters to be taken into consideration while estimation the tool development cost is:

1. Type of Software Project – Software modification, Web development, Software integration, New Software
2. Size of Software Project – Small, Medium, Large organisation and Enterprise
3. Development Team Size-6 to 8 team members

Table 1. Time required for developing software

	Software Modification	Web Development	Software Integration	New Software
Small (weeks)	1-2	3-5	2-6	3-6
Medium (weeks)	2-6	4-6	3-8	4-12
Large (months)	2-6	8-12	6-14	6-18
Enterprise (months)	4+	6-8+	8+	8+

Roles required in the software development:

- Project Manager
- Developer
- Business Analyst

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 6, June 2021 ||

| DOI:10.15680/IJIRSET.2021.1006358 |

- Quality Assurance Engineer
- QA tester

Average cost for a team:
2000/day for a developer
25,000/week for a team
Formula: Project Resource Cost x Project time = Project cost
On an average the Project cost turns out to be: 100k -300k

Work cost estimation:

Table 2. Percentage of Cost reduction calculation

	Labor cost per hour	Total number of labor	Average labor hours per folder	Total labor cost (Rupees)
Manual Traceability	100	8	63	509
Automated Traceability	100	3	2	540
Cost Reduction				50372
Percentage of cost reduction				98.94

Breakeven point =

$$\frac{\text{Total cost of tool development} * \text{Number of days required for manual traceability of per folder}}{\text{Total cost of manual traceability of one folder}}$$

$= (250000 \times 64) \div 50912$
$= 315$ days
As a result, when working on manual traceability, the breakeven point arrives after 315 days. Whatever money is spent after 315 days results in a profit for the firm.

VII. CONCLUSION

This paper presents a study to assess the effect of the presence of traceability links during software maintenance tasks. The software system is a complex set of components that together work to provide the expected outcome. If anyone of the components is not tested or has issues, it may influence the sequence of the performance of the components. In the paper, a software-development lifecycle framework is provided to automate manual traceability by designing software. The framework presented in the paper has a brief introduction to each phase in SDLC. The report also discusses the consequences of each phase. A model can be constructed when the proposed SDLC framework is utilized to design a software system for automating the traceability link between the activities. The report includes a cost estimate for developing a software model. In the nutshell, the study gives an overview of the traceability topic, tackles manual traceability difficulties, and presents a SDLC framework for designing an automated traceability software. Finally, the study gives software development cost estimates using an example of medium-sized software development which showed a breakeven point to be attended within two and a half months.

REFERENCES

[1] O.Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Gr¨unbacher, A. Dekhtyar, G. Antoniol, J. Maletic, and P. M¨ader, \Traceability fundamentals," in Software and Systems Traceability. Springer, 2012, pp. 3-22.
[2] \Ieee recommended practice for software requirements specifications," *IEEE Std 830-1998*, pp. 1-40, Oct 1998.
[3] E. Bouillon, P. M¨ader, and I. Philippow, \A survey on usage scenarios for requirements traceability in practice," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2013, pp. 158-173.
[4] Dora, S. K., & Dubey, P. (2013). Software Development Life Cycle (SDLC) Analytical comparison and survey on Traditional and agile methodology. *Abhinav National Monthly Refereed Journal of Research in Science & Technology*.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 6, June 2021 ||

| DOI:10.15680/IJIRSET.2021.1006358 |

[5] Mouzakitis, A., Copp, D., Parker, R., & Burnham, K. (2009). Hardware-in-the-loop system for testing automotive ECU diagnostic software. *Measurement and control*, *42*(8), 238-245.
[6] Nissimagoudar, P. C., Mane, V. R., Iyer, N. C., Ramakrishna, S., Kiran, M. R., Uma, K. M., ... &Shettar, A. (2015). A Virtual industry platform for Course projects in Automotive Electronics: A case study. *Journal of Engineering Education Transformations*, *28*(2 & 3), 145-152.
[7]Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8-13.
[8] Nguyen, T. L., Tran, Q. T., &Besanger, Y. (2019). Synchronization conditions and real-time constraints in co-simulation and hardware-in-the-loop techniques for cyber–physical energy system assessment. *Sustainable Energy, Grids and Networks*, *20*, 100252.
[9] Sundaram, S. K., Hayes, J. H., Dekhtyar, A., & Holbrook, E. A. (2010). Assessing traceability of software engineering artifacts. *Requirements engineering*, *15*(3), 313-335.
[10] Spanoudakis, G., & Zisman, A. (2005). Software traceability: a roadmap. In *Handbook Of Software Engineering And Knowledge Engineering: Vol 3: Recent Advances* (pp. 395-428).
[11] Boehm, B. (1988). A Spiral model of software development and enhancement. IEEE Computer, May 1988, pp. 61–72.
[12] Boehm, B. (2000). Spiral development: experience, principles and refinements (Special Report CMU/SEI- 2000-SR-008). Carnegie Mellon University.
[13] Cleland-Huang, J., Gotel, O. C., Huffman Hayes, J., Mäder, P., & Zisman, A. (2014). Software traceability: trends and future directions. In *Future of Software Engineering Proceedings* (pp. 55-69).
[14] Kamalabalan, K., Uruththirakodeeswaran, T., Thiyagalingam, G., Wijesinghe, D. B., Perera, I., Meedeniya, D., &Balasubramaniam, D. (2015, April). Tool support for traceability of software artefacts. In *2015 Moratuwa Engineering Research Conference (MERCon)* (pp. 318-323). IEEE.
[15] Karim, N. S. A., Albuolayan, A., Saba, T., & Rehman, A. (2016). The practice of secure software development in SDLC: an investigation through existing model and a case study. *Security and Communication Networks*, *9*(18), 5333-5345.
[16] Pandey, S. K., & Batra, M. (2013). Formal methods in requirements phase of SDLC. *International Journal of Computer Applications*, *70*(13), 7-14.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.512

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर
NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details