International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 9, No.5 (Sep-2020)

http://dx.doi.org/10.12785/ijcds/090519

Novel Framework for Software Designing to Create Reliable & Re-Usable Components

Gurpreet Singh Saini[1], Sanjay Kumar Dubey[1] and Sunil Kumar Bharti[2]

[1] Department of CSE, ASET, Amity University Uttar Pradesh, Noida, India
[2] DCSA, Central University of Haryana, Mahendergarh, India

Received 2 Mar. 2020, Revised 31 Mar. 2020, Accepted 2 Jul. 2020, Published 1 Aug. 2020

Abstract: Re-usability & Reliability are eminent factors in development of new software's to control the increasing costs and development period. The suppliers need to customize software as per the customers, however managing the same using re-usable components is a difficult task and determining their reliability is a much tougher job. While using the re-usable code, the focus lies on the coding or low-level design such that it can be merged easily with the new code. Though, it may be one of the options to chalk out a customized solution but focus has to be put on functional requirement during the development period. These components with re-usable need are to be developed with utmost care due to high degree of cohesion and less degree of coupling to other modules with higher reliability factor. The novel framework discussed in this paper describes one such strategy wherein a module can be developed with less coupling to the modules of the software's being developed and reliability as one of the prime focus. The framework possesses the ability to customize the development dependent upon the nature of the product under work. Hence, customizing the re-usable component makes it quick to develop with few modifications at the functional and parametric level. These Re-usable components also lead to short development time-periods along with man hours. Less modifications in modules will also lead to clear testing and hence, less no. of bugs leading to low maintenance costs of the product. The paper also focusses the areas which can use this strategy for development in Future scope section as summary.

Keywords: Software Reuse, Software Re-Engineering, Software Reliability, Software Project Management, Functional Paradigm

1. Introduction

With the growing focus on cost effective software development, the industry has shifted towards process of software reuse, which directly relates to development of those components which could be easily migrated to newer software's [1]. These modules of re-use need to be developed with consideration at all the levels of development i.e, unit, module and even the application level [2] such that they can be relied upon for delivery of service with required functionality. The architecture must have a focus to produce a module which is open to be implemented in multiple software's having similar functionality. Hence, the module must be developed keeping into mind the re-use of functionality [3], Design pattern [4] [5], Architecture [6] and its Economic viability [7] [8].

In order to develop a software module effectively, there should also be concern towards non-technical aspects of the software development and it's re-use such as organizational structure, code reuse, measurement of product lines and also the economic prospects based upon reliability factor [9]. This development means that the module should be free from coupling through parameter passing. Before merging of any module, a feasibility study must be conducted so as to map the variables between the re-usable module and the new software which are to be merged. Any module which could be working fine in System X will not mean that it will work in a similar fashion in System Y [10]. An ideal definition of reliability in such module can be defined as error free working of that module when integrated with a new or existing system for added functionality.

The novel Framework as discussed in series of work [2] [11] [12] [13] focusses on one such methodology, wherein the development is cycle oriented and independent of the previous iterations. This complete process of development is discussed in further sections followed by the comparative results so as to establish the benefits of the framework.

2. Literature Review

Software re-usability and reliability being the need of hour has strong focus of research fraternity. There has been

E-mail: g.saini4888@live.com, skdubey1@amity.edu, sirbharti@gmail.com

numerous finding and discussions on methods and technique's to enhance the re-usability of a component which further results in increased reliability. Initial software development methods delivered the concept of re-use only but were largely dependent upon the coding ability and its change process for the same. [13] These methods were later taken over by OOAD (Object Oriented Analysis & Development) & Value Based Software Engineering [14], putting focus on building product around needs which are present and may occur in future hence adding reliability as a need to consider. The method was taken a step further in form of Architecture Oriented Development [15], which had drawbacks in form of chaotic process management affecting the reliability of the output module. All the development models had software re-usability as a focused outcome but none came near the approach set by Component Based Software Engineering process [16]. The Component model made a leap by developing independent modules which were at the end joined together to give an end product. These modules connected to each other to form a large software catering to demands of users of different domains by change in parameters. However, the Component Based software development failed to answer longer development period, reliability and platform dependency and got limited to products of small scale use like web services.

With the growing need of industry, almost 60 models were put in all together between the year 1990-2018 [17] and all the models failed to answer how the independent modules will be built with multiple constraints under effect of human intervention being the prominent cause of the chaotic nature [2] [11]. The framework suggested by Saini et al [2] [11] tried to answer how intervention can be controlled using the fuzzy methodology and with resource allocation and selection of development team as the heart of process.

The Novel Framework Algorithm [11] has deep focus on answering the key factors of how modules must interact taking into consideration the key factors i.e. dependency (functional & Non-Functional), Output, Non-technical Pre-requisites and also the nature of module to be developed.

The key Focus of the Model lies in division of modules into three major groups [11]:

- SOFT: Requirement with less focus and parametric functional need which require a little or No coding change over pre-developed modules.
- MEDIUM: These requirement are easily to answer as they may be developed as an enhancement to a previous available repository component with little change. They are the re-usable components which require parametric change.
- HARD: These components are newly to be built components and require real focus of the developer as they hold up work if not prepared timely. These have direct effect on the cost of project as they require extensive timelines.

The model has also focus on extensive use of collective prioritization technique which takes into account feedback by each team member on the sequencing of requirements for development. This sequence allows the development of system in a phased manner where each iteration of the development produces an independent module which could be re-used if and when required in the future. Each module developed i.e. iteration output also carries a tag group Soft/Medium/Hard in terms of their coupling requirements and functional dependencies on external parameters. These parameters depict how much reliable a module will be if introduced in a system other than it is developed for. The reliability of each module is tested extensively and same will be elaborated upon in the design section.

The model referenced under novel framework algorithm is focused on answering major three questions of software development and component re-use.

- Prioritizing the requirements, on a collective weighing methodology of fuzzy so as to have a specific timeline, resources and costing for a given module.
- A proper workflow so as to have a pre-recognized matched resources to each module for their independent development and producing an early working model. Developing the system with efficient switching of resources will in turn lower down the cost.
- Efficient and rigorous testing framework which will ensure lowering down the risk of "failures" in the final product.

3. Design And Output

The focus of each model is to develop a product free of defects and same lies at the heart of Novel framework development model. The study of multiple development model [18] [19] [20]shows that numerous factors affect the stability of a system and the component re-use can be clearly achieved if the issues marked in Figure 1 are handled properly at relevant stage of system development.

These issues if not answered properly with right strategy may in-turn even lead to product failure. Hence, Novel framework proposes a development cycle to answer these issues at their point of generation itself. The Novel Framework follows an evolved version of Iterative Development Life Cycle which is more of Function and Need oriented. The requirement prioritization process in Novel Framework algorithm is rigorous in scheduling the requirements and follows keen detailing through means of Dynamic graphs for the dependency marking with use of fuzzy logic for the weighing mechanism as discussed in saini et al [12].

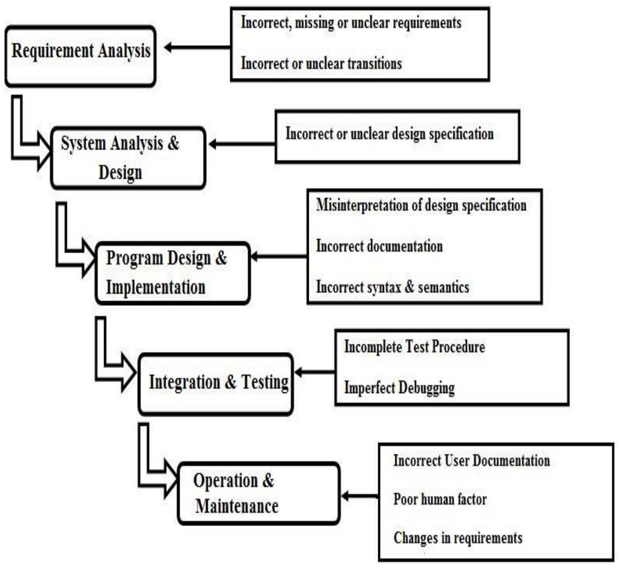
Requirement Analysis
Incorrect, missing or unclear requirements
Incorrect or unclear transitions
System Analysis & Design
Incorrect or unclear design specification
Program Design & Implementation
Misinterpretation of design specification
Incorrect documentation
Incorrect syntax & semantics
Integration & Testing
Incomplete Test Procedure
Imperfect Debugging
Operation & Maintenance
Incorrect User Documentation
Poor human factor
Changes in requirements

Figure 1. Issues at different level of SDLC which damage Component Re-use & Reliability

This complete cycle of development is depicted using Figure 2 which marks an Iterative cycle with multiple outputs as functional modules developed out of prioritized requirements. These cycles of development may be run in a serialized manner or parallel depending upon the Dependency graph. However, utmost care is given to the output of each iteration such that maximum modules developed should fall under Soft or Medium Category. More the no. of HARD modules less re-usable it is for future re-use.

This Iterative model of Novel Framework establishes a process which makes this process more reliable in terms of development module's which are independent and free of external inputs making manipulations in its internal functionality. These modules also represent Evolutionary mechanism as one module is added to an older working module and hence providing an early working system which can deliver some or part functionality at the initial cycle of software development. This cycle is efficient in a way that it runs multiple processing at same timeline. For example if one module is being developed at a moment, the previous one might be getting tested at developer end, another one may be deployed at user end for acceptance and a new one being studied for its requirements need. This complete process leads to efficient resource planning as each team will have a process to cater at a given timeline. Hence, lowering the system cost.

The above mentioned process of validating the model can be understood using the Figure 3 which shows how the process will deliver the tested product for re-usability at every iteration.

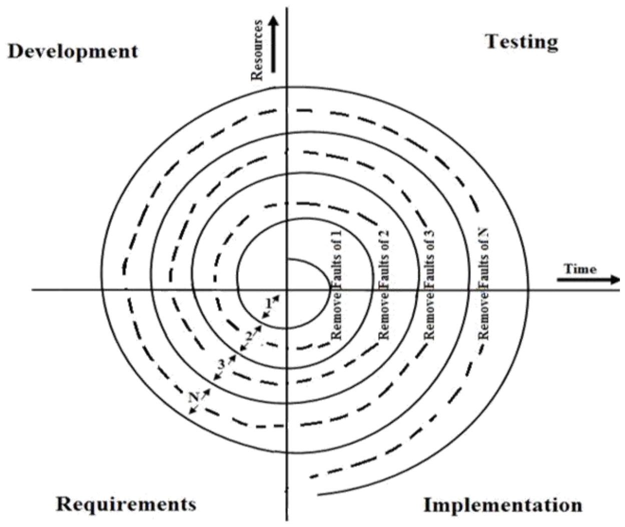

Figure 2. Evolutionary Model of Development removing Faults at each Iteration.

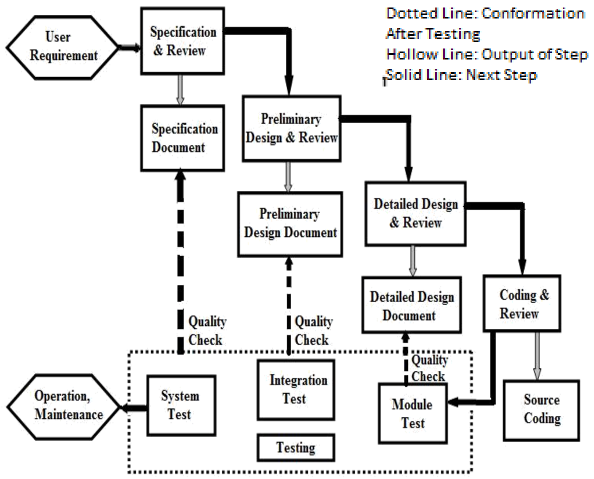

Figure 3. Process of Testing and validating each component for the System

Figure 3 also details how the components are integrated once they are developed. The integration must follow the relevant design documents of each level. The confirmation to the specifications mentioned under each document type will lead to less chances of faults and failures, which in turn will lead to an economically viable system due to less maintenance and for future integration to any other system on requirement basis.

As Reliability and Re-usability are among the key factors of functionality in Novel Framework, their measurement directly marks the functionality of a provided system. Reliability & Re-Usability [21] being a major concern has been quoted in multiple text [20].

The basic formula for the calculation of Reliability "(1)" and Re-usability "(2)" are:

$$MTBF = MTTR + MTTF \quad (1)$$

$MTBF = Mean\ Time\ Between\ failure$

$MTTR = Mean\ Time\ To\ Repair$

$MTTF = Mean\ Time\ To\ failure$

$$Reusability\ Factor\ (Rf) = \frac{Lines\ Of\ Code\ Used\ Without\ Modification\ (Lc)}{Lines\ Of\ Code\ present\ in\ system\ (L)} \quad (2)$$

As, the study of Novel framework has focus on removing faults on the parallel cycle. There is a bound timing constraint for removal of each fault for a given module. The removal of concerned fault must finish before the finish of next cycle as the new output module needs to be integrated over the previous module. Hence, the testing strategy implemented in the Novel Framework Algorithm is a cyclic version as depicted in Figure 4 for Integration and Acceptance testing. This only leads to focus of team over one kind of testing at the last iteration output.

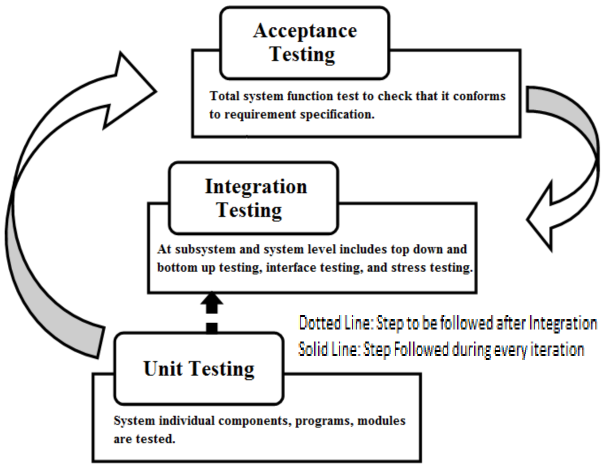
Acceptance Testing
Total system function test to check that it conforms to requirement specification.
Integration Testing
At subsystem and system level includes top down and bottom up testing, interface testing, and stress testing.
Unit Testing
System individual components, programs, modules are tested.
Dotted Line: Step to be followed after Integration
Solid Line: Step Followed during every iteration

Figure 4. Testing Strategy Deployed In Novel Framework Algorithm

Since, the faults are removed at every single iteration, they are reduced to N times, where N represents the no. of iterations made. Now, (1) changes to:

$$MTBF(new) = \frac{MTTR+MTTF}{N} \quad (3)$$

On careful examination of (3), we can assume that the faults in the complete development cycle are evenly distributed over the N no. of iterations and when the last Iteration is delivered, the no. of Faults left to manage are only from two sources:

1. The current Iteration.
2. Merging of Output of Current iteration to the previous existing product.

This relates to a very evenly distributed fault management graph as shown in Figure 5. The solid line depicts the no. faults which occurred in the previous cycle of the Novel framework Development and dotted line depicts the no. of faults solved in the Current cycle. It shows that the same no of faults (previous iteration) are solved before the faults of current iteration comes into picture.

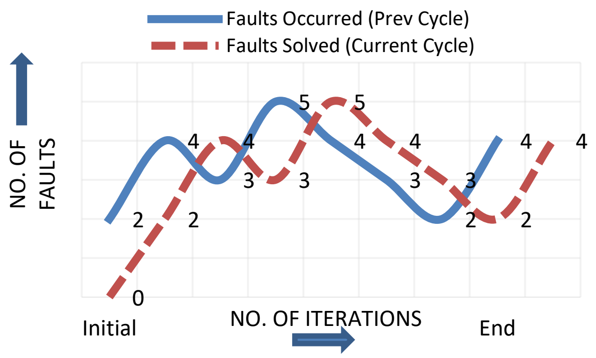

Figure 5. No. Of Faults Managed During Development Cycle taken from Table 1 (Given in Section Result).

The Novel Framework algorithm has focus on four key factors while the determination of re-usability factor.

1. Language of System: It is the key factor which justifies whether the movement of code between two systems can be made or not?
2. Architecture of System: This relates to specialized software's intended to deliver some Domain related functionality and hence their architecture for flow of data must resemble.
3. Algebra: This factor considers the constraints and limitations of the component movement. Each Component must adhere to algebra for flow of data amongst itself and new system.
4. Abstraction & Refinement: This factor will be determined if only the first three are satisfied. This factor relates to decisions available and decisions made over a component to make it compliable in the new system.

These four key factors have a different graphs conduct during the development period as shown in Figure 6 & 7.

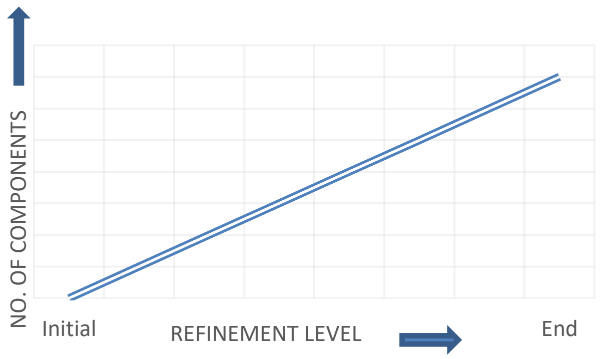

Figure 6. Components Vs Refinement Level

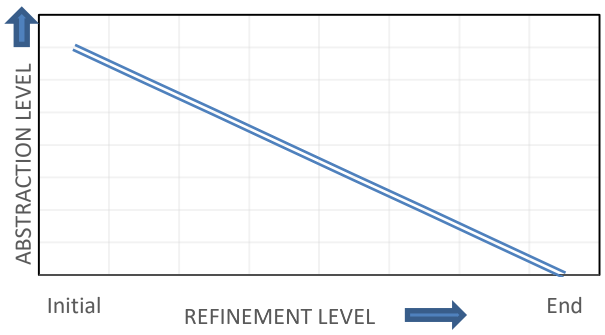

Figure 7. Abstraction Level Vs Refinement Level

The Figure 6 depicts that the as the no. of re-usable components increase in the system to be developed, the Refinement level increases with the same rate in the general system. However, the Figure 7 states that if the abstraction level is high amongst the modules to be re-used, less is the refinement level required by them. These two are well taken care in the Novel Framework development cycle which is also evident from the fact that each module in the iterations are developed independently and only the parameters are passed for the combined system output. This can only be achieved once the system is developed with global functions which are independent of internal module coupling.

The Reusability Factor R_f will increase with the high the level of abstraction taken into consideration in Novel framework Algorithm. The Refinement is reduced with global functions taking less parameters from class and only passing the necessary parameters to the other module for the functionality purpose. This passing will also in-turn reduce the complexity of the code and make it more efficient in terms of processing and throughput.

4. Result And Analysis

The novel framework Algorithm is a multi-tasking evolutionary model which has multiple constraint satisfying ability and for this it relies heavily upon its ability to develop a Quality Module which is reliable and re-usable. This is evident from the fact that every iteration of Novel framework Algorithm delivers an early working module which is tested rigorously and is independent of intake from its predecessors.

The Novel Framework has major features in the development cycle which conform its process flow and adhere to its quality standard:

1. Software modules are developed independently, it's possible that two modules may be worked upon in parallel keeping focus on software re-use. These modules may or may not have similar timelines, however the module with shorter timeline is delivered followed the ones with next higher timeline and so on.

2. The Novel Framework employs a rigorous mechanism of requirement prioritization and resource allocation to them. Hence, the faults which occur are only of code. These coding faults are generic and are easily traceable.

3. As the faults are detected once the module is on working mode, the Novel Framework assures the removal of these faults before the next iteration is integrated to the current one.

4. As all the iterations are developed independently with only variable and their types as common point of flow between two different modules, the coupling is very low.

5. The testing strategy as shown in figure 3 ensures a multi-level strategy for removal of the defaults, this ensures that quality is not compromised and a reliable product is delivered at the user site.

The following Table 1 is comparative analysis of multiple models of software development with key attributes which directly affect the re-use and fault removal. Each entry is descriptive of the approach taken by the development model regarding the fault management.

TABLE I. COMPARATIVE ANALYSIS OF MULTIPLE MODELS FOR FAULT OCCURRENCE AND SOLUTION

Novel Framework			Value Based Development			Architecture Based Development			Component Based Development		
Iteration	Faults Occur	Faults Solved	Iteration	Faults Occur	Faults Solved	Iteration	Faults Occur	Faults Solved	Iteration	Faults Occur	Faults Solved
1	5	0		5	0		5		1	5	0
2	4	5		4	0		4		2	4	2
3	3	4	1	3	0	1	3	16	3	3	4
4	4	3		4	0		4		4	4	3
5	0	4		0	16		0		5	0	5

Novel Framework Algorithm clearly steers past the other three popular models by two major factors:

1. Every Iteration is delivered only when the previous faults are answered. Hence keeping each timeline and module of development independent. This enhances the re-usability. The component Based Development model does not take care of this and hence the deliverables are more prone to delay.

2. As it is Iterative in nature, the requirements are justified and could be run parallel to maximize the use of each functional team. This will lead to early deliverables which can be tested and deployed at user site for limited functionality. The other functionalities are later added as enhancements.

Now, assuming that each fault occurrence refers to change in 2 LOC per fault, then the Table 2 comes into role.

Using the values in Table 2 and the Re-use Factor Table 3 is generated and it confirms the Novel Framework delivers an early to use re-usable component which can be deployed easily without and dependency other than Hardware oriented.

Table 3 indicates that each model delivers a re-usable component at some stage of development cycle if it's presumed that the platform remains same for the system development with evidently no coupling between the modules which is dependent directly on the process flow. These models deliver a MEDIUM OR HARD re-usable software module; but, Novel Framework due to its process flow and strong resource prioritization technique delivers an early working independent re-usable module which has SOFT module development feature as its goal.

TABLE II. COMPARATIVE ANALYSIS OF FAULTS FOR RE-USE FACTOR

Novel Framework			Value Based Development			Architecture Based Development			Component Based Development		
Iteration	Faults Occur LOC	Faults Solved LOC	Iteration	Faults Occur LOC	Faults Solved LOC	Iteration	Faults Occur LOC	Faults Solved LOC	Iteration	Faults Occur LOC	Faults Solved LOC
1	10	0		10	0		10		1	10	0
2	8	10		8	0		8		2	8	4
3	6	8	1	6	0	1	6	32	3	6	8
4	8	6		8	0		8		4	8	6
5	0	8		0	32		0		5	0	10

TABLE III. RE-USE FACTOR TABLE

Novel Framework		Value Based Development		Architecture Based Development		Component Based Development	
Iteration	Re-Use Factor	Iteration	Re-Use Factor	Iteration	Re-Use Factor	Iteration	Re-Use Factor
1	0		0		0	1	0
2	1		0		0	2	0.4
3	1	1	0	1	0	3	0.57
4	1		0		0	4	0.5
5	1		1		1	5	1

5. Conclusion

The results drawn through comparison amongst multiple software modules depict that Novel Framework has one strong feature of dividing the requirements and development of each requirement as an independent functional SOFT module. This independent module passes through a rigorous testing framework to ensure the quality of the system and enhanced reliability in terms of code stability and value passing.

The Novel framework algorithm is a complex software development module with evolutionary iterative strategy to isolate the faults from module affecting the successor. This strategy is also clear in answering the issues which might occur at different stages of development cycle due to Language, Algebra, Architecture and the Refinement of modules at each level.

Removal of these all hindrances result in a goal oriented module which is effective and developed in a shorter time frame with highest quality and low cost.

6. Future Scope

The Novel Framework algorithm is a generalized approach and it could be refined for use in multiple environments. However, few modifications can be done to suit the platform of each product being developed such that crisper handling or process flow can be derived. Each step can itself be modified as per the need of the user and being an easily understandable process, it can take AI tools as decision enhancing means and can give relevant results which could be deployed in Business Intelligence.

References

[1] D. Manjhi and A. Chaturvedi, "Software Component Reusability Classification in Functional Paradigm," *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, 2019, pp. 1-7.

[2] G. S. Saini, S. K. Dubey, S. K. Bharti, “Fuzzy Based Algorithm for Resource Allocation”, Springer- Advances in Intelligent Systems and Computing, Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications, FICTA 2016, Volume 1 – 515, pp 69-78

[3] F. Lanubile and G. Visaggio, “Extracting Reusable Functions by Flow Graph Based Program Slicing”, IEEE Trans. Software Eng. vol. 23, no. 4, pp. 246-259, Apr. 1997.

[4] N. Soundarajan, J. O. Hallstrom, “Responsibilities and Rewards: Specifying Design Patterns”, Proceedings of 26th International Conference on Software Engineering, pp. 666-675, May 2004.

[5] J. K. H. Mak, C. S. T. Choy, and D. P. K. Lun, “Precise Modeling of Design Patterns in UML”, Proceedings of 26th International Conference on Software Engineering, pp. 252-261, May 2004.

[6] Y. Peng, Y. Shi, J. Xiang-Yang, Y. Jun-Feng, L. Ju-Bo, Y. Wen-Jie. “A Reflective Information Model for Reusing Software Architecture”, ISECS 2008 International Colloquium on Computing, Communication, Control, and Management. 2008.

[7] J. Sametinger, “Software Engineering with Reusable Components”, Springer-Verlag Berlin Heidelberg. 1997.

[8] N. Padhy, S. Satapathy, R. P. Singh, “State-of-the-Art Object-Oriented Metrics and Its Reusability: A Decade Review”, S. C. Satapathy et al. (eds.), Smart Computing and Informatics, Smart Innovation, Systems and Technologies, Springer, 2018

[9] Y. Xu, J. Ramanathan, R. Ramnath, N. Singh, S. Deshpande. “Reuse by Placement: A Paradigm for Cross-Domain Software Reuse with High Level of Granularity”, ICSR 2011, LNCS 6727, pp. 69–77, 2011.

[10] G. S. Saini, S. K. Dubey, S. K. Bharti, “Fuzzy Based Novel Framework for User Oriented Software Engineering”, Journal of Engineering Science and Technology, Vol. 14, No. 1, 2019

[11] G. S. Saini, S. K. Dubey, S. K. Bharti, “Novel algorithm for software planning & development” ACM - International Conference Proceeding Series, Proceedings of ICAICR-2019.

[12] B. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. McLeod, M. Merritt, “Characteristics of Software Quality”, North Holland, 1978.

[13] T. Dybå, B. A. Kitchenham, M. Jørgensen, “Evidence-Based Software Engineering for Practitioners”, IEEE Software, Published by the IEEE Computer Society. January-February, 2005

[14] R. Schmidt, “Software engineering: Architecture-driven Development”, NDIA 15th Annual Systems Engineering Conference, October-2012

[15] M. Yalhali, A. Chouarfia, “Towards a software component assembly evaluation”, IET Software, 2015, Vol. 9, Issue 1, pp. 1–6

[16] Y. F. Perez, A. F. Estrada, C. Cruz, J. L. Verdegay, “Fuzzy Multi-criteria Decision Making Methods Applied to Usability Software Assessment: An Annotated Bibliography”, C. Berger-Vachon et al. (eds.), Complex Systems: Solutions and Challenges in Economics, Management and Engineering, Studies in Systems,Decision and Control, Springer, 2018

[17] P. K. Kapur, H. Pham, A. Gupta, P. C. Jha,, “Software Reliability Assessment with OR Applications”, Springer Series in Reliability Engineering, 2011

[18] D. Kelly, “Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software” The Journal of Systems and Software, Elsevier, Vol. 109, 2015

[19] C. Y. Chong, S. P. Lee, “Analyzing maintainability and reliability of object oriented software using weighted complex network”, The Journal of Systems and Software, Elsevier, Vol. 110, 2015

[20] B. M. Goel, P. K. Bhatia, “Analysis of reusability of object-oriented systems using object-oriented metrics”, ACM SIGSOFT software engineering notes, ACM; 2013. p. 1–5. Issue 4

[21] K. Tyagi, A. Sharma, “A rule-based approach for estimating the reliability of component-based systems”, Advances in Engineering Software, Elsevier, Vol 54, 2012

Gurpreet Singh Saini
Gurpreet Singh Saini is a doctoral candidate at Amity University Uttar Pradesh, Noida working in the field of Software Resource Planning and Reliability. He has keen interest in developing solutions for the software industry which can reduce the costing and efforts put in a software project. Also, currently he is working as an Instructor of ICT in Directorate of Education, Government of NCT of Delhi, India.

Sunil Kumar Bharti
Dr. Sunil Kumar Bharti is currently associated with Galgotias University Uttar Pradesh, Noida and Central University Of Haryana, Mahendergarh as Associate Professor for Computer Science and relevant streams. His Areas of interest Include Big Data, Software Engineering, Algorithm Design and Neural Architecture.

Sanjay Kumar Dubey
Dr. Sanjay Kumar Dubey is currently associated with Amity University Uttar Pradesh, Noida as Associate Professor and Head of Research Division for Amity School of Engineering And Technology for Computer Science and relevant streams. His Areas of interest Include Algorithm Analysis, Software Engineering and Big Data Analysis.