Journal of Engineering Science and Technology
Vol. 14, No. 1 (2019) 073 - 086

FUZZY BASED NOVEL FRAMEWORK FOR USER ORIENTED SOFTWARE ENGINEERING

GURPREET S. SAINI[1,*], SANJAY K. DUBEY[1], SUNIL K. BHARTI[2]

[1]Department of CSE, ASET, Amity University Uttar Pradesh, Noida, India
[2]DCSA, Central University of Haryana, Mahendergarh, India
*Corresponding Author: g.saini4888@live.com

Abstract

A Software Development Life Cycle (SDLC) framework fits the equation of use when developer finds it perfect for the deployment as per the user requisites stated in Software Requirement Specification phase. This phase being the primary step of development can either move it to right or wrong direction resulting into a successful, incomplete or a failure product. The research paper here presents a fuzzy based novel approach on how these software requirements can be prioritized or divided such that an efficient framework of development i.e. a SDLC can be chosen for successful outcome. The research presents a model how fuzzy logic can be employed in weighing the requirements such that they can be serialized or run parallel to each other such that the development time can be reduced and an early product can be prepared using a collective effort of the development team. The matching process between the resources and the module of work will be made efficient using the modified stable marriage algorithm such that the work is done actively and effectively. The collective effort at beginning will lead to a reduced risk of software development process failure at the end. The framework represented here will result in an efficient model which can answer how the resources should be prioritized and who should undertake the development of specific module along with specific timeline.

Keywords: Software development life cycle; Prioritization algorithm; Soft computing; Fuzzy logic; SDLC; Software requirement elicitation & estimation; Stable marriage algorithm.

1. Introduction

Most of the software development projects do not reach a success, as per the reports made available by Schmidt [1], Sirrush Corporation and one key factor stated in those reports point for the failure in prioritizing the software requirements, which is the major step in the early phase of software development. The reports state that almost two-third of the projects never completed or were deficient on the requirements placed by the end-users. This statement highlights the need of appropriate evaluation of requirements at the first step itself. Today, most of the development models are developer oriented and fails to recognize the needs of end-user completely. These development models generate products, which are low on quality as subjected to end-users satisfaction. Even though a developer has worked hard in developing the system, the conditional and perceptual nature of this subjective entity "quality" leads to non-satisfaction of the end-user.

Saini et al. [2] presented the novel framework algorithm in this paper for evaluation of requirements and their prioritization makes use of soft computing technique of Fuzzy logics [3, 4] to prepare dynamic paths resulting into dynamic graphs [5, 6] which can lead the direction of software development cycles. Also, this novel framework algorithm is one step ahead in providing efficient way of generating Dynamic graphs as stated in Dynamic Graph Theory [7]. The idea employed states "Break the requirements into the smallest possible task followed by assigning that task a weight and then prioritize these tasks based on factors of Independence, Time, Functionality and Re-usability. After; weighing of these tasks by each development team member, use the modified stable marriage algorithm to prioritize these tasks and deduce the model of development to be followed."

2. Related Work

Many software development modules have been proposed and they all have failed to reduce the failure rate of the software development projects [8, 9]. These models although answered many important queries of end-users but never answered complete requirements as they were developer oriented.

The first model answering this "chaos" of Software Engineering based on Chaos Theory [10] was developed in view to answer the context relative to future events, which are going to occur, based upon the current future. The official statement of Chaos theory states "When the present determines the future, but the approximate present does not approximately determine the future". This theory was highly used in answering the future states in operations of Non-linear, complex systems wherein Uncertainty and Predictability play a vital role. Based on the same grounds, the field of software engineering relies totally on uncertainty and prediction to drive upon and reach the goals stated during the initial phase of software development life cycle. Initial literature review lead to discovery of only one concrete model, which was based on Chaos Theory strategy for software building, named as "Chaos Model" given by L.B.S. Racoon. The Chaos Model employed simple strategy of dividing the problem state into substantial sized individual and non-conflicting smaller problems each having its own state definition.

However, so close the "Chaos Model" came in answering many prominent questions related to "what is to be done?", it failed in answering most of the questions relative to "How it is to be done?". To this very point, the framework

stated in this paper will come into effect. The Model had major backdrops in form of answers to the following:

- What requirement or functionality has to be designed at a given time and what resources, time, testing strategy has to be employed over it?
- In defining, how one could break down the system and produce an early working model?
- Finally, to what extent we can minimize the risk of "failures"?

In the following years lot of work was carried out in form of few models proposed in industry. According to Jifeng et al. [11], the second model Component Based Software Engineering model, which was partly developed as a part of research task of E-Macao Project funded by the Macao Government, was implemented but was discarded on the grounds of more emphasis on "structured programming" and "Component re-use" discarding the basic idea of innovation at every building block of software engineering.

The third model Value-Based Software Engineering model [12] proposed by Barry Boehm in which the stress was levied upon dividing the tasks based upon its value in terms of user input, incomplete requirements, changing requirements, lack of resources, unrealistic expectations, unclear objectives, and unrealistic time frames. However, it was later discovered that this approached violated developer of moving ahead with freedom to prioritize the development cycle as per resources at its disposal. But, this approach introduced many important features like user intervention in software development lifecycle, such that it's transparent and makes user fully responsible for the requirements, which are "Hypothetical" in nature but still established by user.

Dyba et al. [13] proposed the fourth model named as Evidence-Based Software Engineering and lot of work was done over it and it focused at finding evidences for deciding upon which requirements must be included and their relevance to the project. The model focused upon raising a question out of stated problem followed by generating background information so as to felicitate answering. Once, the information is found, it is critically analyzed for its validity and applicability. Once this complete process is completed, the module is developed and user inputs are taken to evolve it further.

The strategy employed a very good approach but consumed a lot of initial time in finding the evidences, which incurred great cost at the initial stages itself, which led to development of one more strategy answering Chaos of Software Industry known as "Architecture based Software Engineering". But, this approach was totally non-chaos based strategy as it revolved around the architectural need of software and employed chaos model only at damage control stages occurring at prelim stages of failure known as "Critical Stages".

There have been numerous backdrops of the all the proposed models discussed above. These models did answer few of the backdrops but in the process, they moved away from the basic "chaotic" nature of software modelling which was totally based upon predictive nature of future given the current statuses of the system. Each one of these models were clear in their approach but lacked in answering multiple user valued queries.

In general, situations of Chaos the software development [14] team implies damage recovery procedures based upon any of the above available methods or some classical software development approaches. Chaos strategy generally relates to the situation of urgency and in development cycle, the state of urgency begins when the project is about to cross the schedule timeline or is near failure. In such scenario, the developers look towards the expert help for prioritizing the task and marking the bugs such that they can follow the schedule and look into problems one by one. The chaos strategy drives each development team under such situation to this kind of approach as a standard to be followed [15].

The major answer could be found to such a problem in how the resource allocation algorithms work in an operating system. However, they may not be sufficient enough in answering all the problems as human aspect is part of software development cycle and one major contributor to the chaotic nature of the software. As, the Problem statement here deals with chaotic nature of Software Engineering. A more efficient algorithm is required to eliminate the basic factor which contributes to this chaotic nature i.e. Prioritization of all the requirements in an efficient manner, which can be achieved using the following algorithm.

3. Design

The field of software engineering relies totally on uncertainty and prediction to drive upon and reach the goals stated during the initial phase of software development life cycle. Schweitzer et al. [16] explained that the initial literature review lead to discovery of simple strategy of dividing the problem state into substantial sized individual and non-conflicting smaller problems each having its own state definition. The division of the problem must be done on following parameters as follows:

- The most important fact is to divide modules on the basis of HARD, QUICK, and MUST issues.
- A HARD requirement needs new programming with lot of early dependencies over other requirements and hence provide the required functionality to the work, which has value for the user.
- QUICK requirements are those, which need timely attentions, else they hold up the dependent requirements for a longer time period.
- MUST requirements are re-usable components, which are already developed and could be integrated easily and hence developers can safely focus their attention towards the HARD, QUICK modules.
- To solve means closing a requirement and bringing it to its solved state having all the necessary requirements stated by the user.

Before start of design few basic assumptions [17] are made such that the Novel framework algorithm [2] can answer the delay's dependent upon human ability to perform a task for a given module.

- The algorithm assumes that all the resources have same level of work ability.
- Also, the resource is thoroughly able to perform the task without any malfunction.
- All, the resources are independent in their work approach but are temporal to the throughput to be generated.

The idea states "*Take the requirements providing very critical and uttermost required functionalities based on decisions taken through Dynamic network generated through the applied Novel Framework algorithm (Fuzzy oriented) on stated requirements during earliest phase of software cycle, and build a system around it (Using Modified stable Marriage Algorithm for managing the resources and targets). The leftover requirements should be imparted to the system as an update*". Based on studies by Lane and Gobet [17], Kelly [18], Chong and Lee [19] and Tyagi and Sharma [20], it hold value in using the tools already referred in introduction, which could provide us with a relevant solution to this chaotic nature of Software engineering.

The Novel Framework Approach has following procedure. It can be understood as following steps.

- All requirements of the user should be retrieved first and then divide them on the basis of the priorities given. These priorities are based on the collective weights given by the end users and the experience team, not decided by the individual.
- Formulate a heuristics that assess the weights or priorities of the modules.
- For allocating the resources, stable marriage approach must be applied for every requirement having priorities and then predict the priorities of the task from starting to the end. The iteration of the modules that are supposed to be processed after the initial phase should be given priorities using the methodology of fuzzy logic.
- After the prioritization send the listed and weighed modules for resource allocation.
- Now start processing the modules and construct prototype of the initial working module, then add new functionality to the existing or the older version of the module with each successive iteration.

4. Algorithm

The ideology behind the design of the algorithm is simple and can be described as, generates a dynamic network through use of fuzzy engine [20] followed by stable marriage resource allocation [21] in the modified form as stated later in this section based on the requirements of the initial phase of software cycle. Now gather the requirements that provide critical and most required functionalities based on the decision taken through this dynamic network, and then develop a system. The requirements that are left should be implemented into the system as an update.

Finally, the complete Novel Framework will have the following process flow as depicted in Fig. 1 to generate results, which could lead to design of Dynamic Graphs, which are one of a kind solution in building up the complete strategy for the Software Engineering.

Figure 1 gives a brief outlook of how this algorithm could be implemented over a machine using following steps:

- Identify the Membership Function over which requirements could be put under fuzzy constraint (Functional/Non-Functional/GUI/Cohesion)
- Creation of weights for each Membership Function and preparing rules to prioritize the requirements based upon them.
- Refining the weights while Fuzzy Inference Engine is ON.

- Repeating the process for each individual, which is part of Prioritization Team.
- Accommodation of each requirement dynamically by creating a union between Dynamic Graph Strategy and Fuzzy Output.
- Theoretical verification through Fractal Identification for future projects.
- Then repeating the complete process at each new iteration to be introduced into the first evolution.
- After prioritization step has been conducted, we move to next step of applying modified resource match algorithm to allocate resource to a specific module.
- Once the resource is allocated, processes can be run in conjunction or serialized as per the dependency generated in the beginning after the fuzzy step.

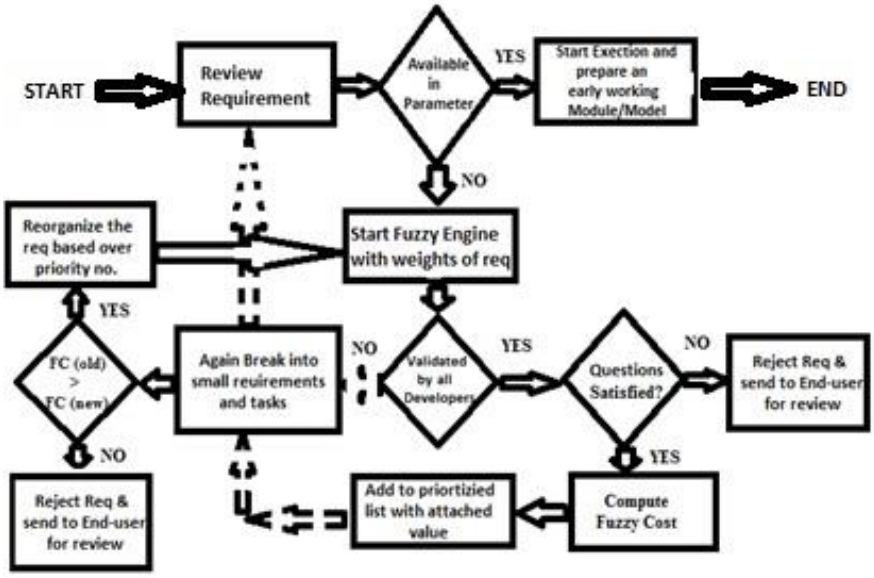

Fig. 1. Fuzzy process of prioritizing the requirements.

This complete process will help reduce the timing of the complete development cycle and hence reducing the cost and the wastage of resources over MUST task by channeling them towards the QUICK and HARD task. This complete speedup of the process could be understood using the following mathematical expression.

After, the problem has been reduced to sub modules to be worked upon, an optimum resource allocation process is carried out using the modified Stable marriage algorithm wherein, the matches are made between two distinct entities based upon the following factors:

- The match is to be made if the resource is free at the given moment.
- The match cannot be altered until the current task has been performed and reached the point of stability
- To come to stability means developing an output which could be valuable to project and can be validated.

The algorithm can be presented for the same as follows keeping in consideration that the algorithm is applicable only after the resources have been also marked as per the stated division & nature of the entity or requirement. The Modified Stable marriage algorithm [2] is used in context of managing the resources where the right development team is allocated the requirement (on the basis of its nature and requisite) along with the right no. of resources (Software, Hardware or Team) to finish the task within the stipulated time. Also, the algorithm has dynamic nature so as to create new matches if in case any dynamic change occurs.

ALGORITHM FOR MATCH MAKING

Set each requirement as free

While (requirement = = free)

Do

Check the requirement demanding the resource, which is not allocated yet;

If (requirement ==free) && (resource == free)

Then assign resource to requirement.

Else

If (resource! = free)

Then continue providing resource to old requirement.

Else // (resource = hold)

Resource rejects joining requirement.

End.

Once the allocation of the resources have been done, we now deduce the working mathematical model to validate our design efficiency and ability in bringing down the failure rate. Let us assume the development cycle had

$$T_{old} = T \tag{1}$$

But, as the Novel framework algorithm works with division of modules between the resources to develop. It will lead to two divisions. One being the dependent modules, which will require serialized development and other ones being the independent modules, which could be run parallel. Hence, the time gets divided between two kinds as follows:

$$Tnew = T(S) + T(P) \tag{2}$$

Now, to compare the speedup between the two systems of development, we can draw the following inference with the guidance of Amdahl's law of speedup.

$$S = \frac{Told}{Tnew} \tag{3}$$

Hence, the speedup will take up the final deduction as follows:

$$S = \frac{T}{\{T(S) + T(P)\}} \tag{4}$$

$$T(P) = \frac{(T - T(S))}{N} \tag{5}$$

As, the Novel framework algorithm works upon the division of complete modules and building an early model and then imparting the changes as an update to the system. Therefore, if we assume M is the no. of updates developed to complete the product development, the speedup has also multiplied M no. times.

Hence, the new speedup generated by the algorithm is

$$Snew = S_1 + S_2 + S_3 + S_4 \ldots . + S_m \quad (6)$$

As, every new iteration will have a different speedup, the above equation can be rewritten as:

$$Snew = \sum S_m \quad (7)$$

This speedup in development cycle could become evident when the first iteration of the product is made available. This kind of approach is almost applicable in every scenario. However, this developed iteration may or may not be acceptable to the end-user dependent upon the nature of the product to be developed. This cycle could be easily implemented in Web-based architecture's, Standalone solutions and also for developing solutions, which need up-gradations over a longer period of time.

5. Results

The Novel framework algorithm [2] helps in reduce the complete project development life cycle along with answers to three major questions of the complete life cycle.

- How the prioritization of the tasks should be done?
- How to insure reduction in risk of failure?
- What should be mode of development?

The framework [2] is designed to have multiple membership functions over which the prioritization task could be performed. For the given result the inputs to the fuzzy engine were Dependency, Nature of Requirement (HARD, MUST, QUICK), Feasibility and the ratio of resources available. The membership functions are allocated to each requirement on the scale of 1 to 10. The inference engine follows a Vukic and Velagic [22] commented that type inference model is for a singleton output and hence allows for an easy prioritization. However, Sugeno type [23] inference model can also be used wherever, the output is similar or constant for the different input types.

Also, one can easily understand the process of speedup using the following sample data Table 1. The table contains data from Serialized development of a software using Component based Engineering (given by T_{old}) and the time taken by Novel Framework (T_{new}).

Table 1. Sample data feed taking serial development = 50 % of the total development time.

Iteration	*N*	T_{old}	$T_{(S)}$	$T_{(P)}$	T_{new}	*S*
1	**2**	**5**	**2.5**	**1**	**3.8**	**1.33**
2	**4**	**10**	**5**	**1**	**6.3**	**1.60**
3	**3**	**15**	**7.5**	**3**	**10**	**1.50**
4	**5**	**20**	**10**	**2**	**12**	**1.67**
5	**4**	**25**	**13**	**3**	**16**	**1.60**
6	**3**	**30**	**15**	**5**	**20**	**1.50**
7	**2**	**35**	**18**	**9**	**26**	**1.33**

$\sum S_M$	**10.53**

The sample data used here is further taken up to draw multiple inferences in stating how this novel framework procedure is more efficient and reliable in terms of handling the chaotic nature of the software development cycle. The resources taken are quite dynamic and so is the planning conducted.

Figure 2 illustrates the difference between the two approaches clearly. The series 1 is plotted for the iterations made during the standard software development cycle and series 2 is plotted for the iterations made during the follow of Novel approach based algorithm. It clearly indicates the time saved at each iteration.

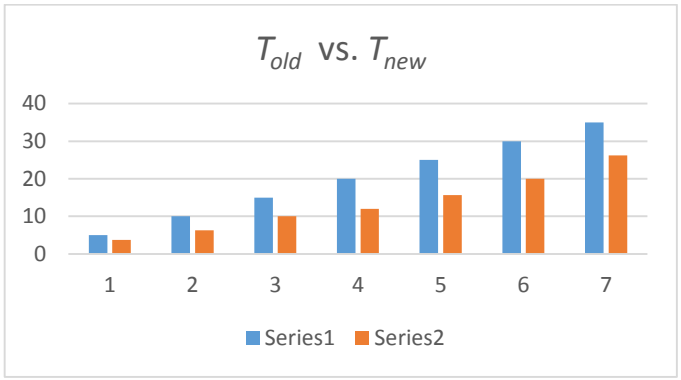

Fig. 2. T_{old} / T_{new}

The process will enable us to speed up the complete development cycle and as evident from the above data table, the speedup will be directly proportional upon the identified parallel tasks and the no. of resources executing them. This could be easily represented using the following speedup graph as presented in Fig. 3.

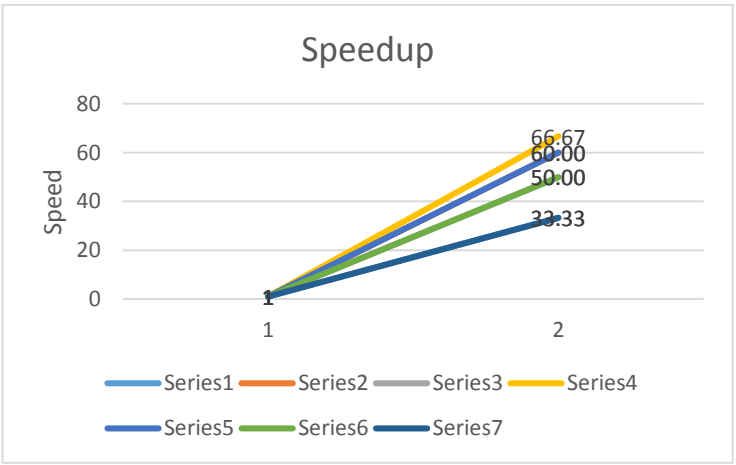

Fig. 3. Speedup of iterations.

The algorithm and results also conform to the standard stating the 67.23 % faster product development for the given data. This is also validated by the standard

speedup graph generated when the system is under parallel core processing or threading mode. As stated by Amdahl [24] and Parhami [25], as shown in Fig. 4, it can clearly outline the standard curves of speedup.

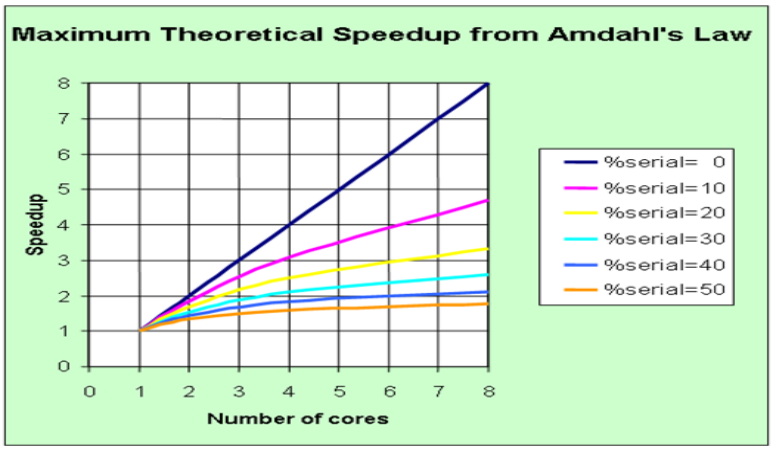

Fig. 4 Standard speedup graph of Amdahl's law.

If we draw comparison between the outputs generated by the algorithm which states 67.23 % speedup with varied no. of resources & 50% process's being serializes one's; we can easily relate it to the lowest line in the standard graph which is almost in the same region of speedup marking same averages. This directly validates the results generated by the algorithm.

With the efficient deployment of fuzzy logic and stable marriage algorithm for computation of appropriate number of resources and their allocation, the Novel Framework algorithm has given the solution to all the backdrops. This has resulted with speedup of 10.53 times for the sample data as visible in Table 1 and given boost in iterations up to 60% for the given sample data. The use is not compulsory limited to implementation within the IT projects as long as alterations and customizations with respect to relevant domains are done.

With the outcome of the literature survey, some factors should also be considered such as every iteration requires certain amount of time, i.e., the time period of the iteration, man power required, adding more functionalities, integrating new updates into the older modules, testing required and training of the end-user for better understanding of the developed system.

We can use this approach at any phase of the project development life cycle or it could be used at the beginning of the project cycle or whenever a development team faces situations, which are challenging to handle.

This solution also allows for ability of software cycle to adapt to the needs of the user who can either shrink it or extend it by allowing for dynamic exchange of user needs and cost related investments, which he can incur over the complete project.

6. Algorithmic Outcomes

The algorithm has presented crisp solutions for efficient planning and draws down a standard equation using which one can easily keep a check on the kind of work,

which needs to be undertaken along with the resources, which are needed for the efficient development. The algorithm will easily produce following deliverable to the planning team.

- An overview by creating a difference amongst requirements by stating them under HARD, QUICK & MUST divisions.
- Enabling the plan of disposal of resources, which could be temporal, and division specific.
- Plan for the prioritization of requirements.
- Generating approximate time schedule for delivery of iterations.
- Measuring the complete speed up using the equations.

This helps in eliminating certain issues such as

- Prioritizing the requirements by an individual only, but now it is done with the experienced team or group of end users.
- Extension in timeline for the complete development of the system.
- Early working system, we can now construct prototype with initial module.
- Clarity in project development cycle for the end user.
- Independence to explore new techniques or ways for finding a solution.
- Interchangeable components that can be reused in terms of software or civil construction, as they are already tested at individual levels and can be combined easily.

7. Backdrops

The designing process of the algorithm contains some key points that cannot be neglected and improve to some extent:

- History cannot always give adequate structure to build upon future projects, which are dynamic in nature and could generate complex equations. Hence, one can only estimate how to proceed in a particular direction by using historical data but can never instruct about how to use resources efficiently.
- Values, which are not predicted easily and are an issue of concern, due to the dynamic nature of the problem one cannot define all the constraints and cannot initialize all the variables while planning to construct a complex system. This changing environment always consist some instances where non-predictive values appear.
- When algorithm is used deliberately that causes it to fail: After practice has been done with some operations, people have recognized the intended meaning of the algorithm and its protocols; they can use it by doing some alterations as per their personal needs that violates its significance.

8. Conclusion

The proposed work aims at answering allocation of resources in an efficient manner that reduces the risk of failure and beneficial in specifying how work has to done using those resources. It will give importance in formulating a model with proper observation of consequences so that it could be invoked at the time of chaos and failure. The evolution of the model will be done by considering the perspective of

the developer during development process accompanied by the views of the customer regarding the needed functionalities.

The complete design will answer three major problems of the chaotic nature:

- Prioritization of resources, which will come out in form of a dynamic graph, which will further indicate the dependency, and the order of the module to be undertaken.
- The resource allocation, which will be produced on a collective weight measuring by complete team.
- Making re-usable components and hence delaying time cycle of the further projects indirectly.

9. Future Scope

A framework is developed from the proposed algorithm, which can be used in various environments such as software project management, operating system, construction models etc. The future actions that may be taken into account is developing a framework, which is specific to the environment by doing changes or converting this generalized framework so that it could become the real time project management solution to the problem whose nature can be clearly identified.

Nomenclatures

T_{old}	Time taken by SDLC prior to use of algorithm
T_{new}	Time taken by SDLC after use of algorithm
$T(P)$	Time taken by Parallel Modules for development
$T(S)$	Time taken by serialized modules for development
S	Speedup of development work when divided amongst multiple resources
N	No. of resource Instances
S_m	Speedup generated after implementation of algorithm
S_{new}	Speedup at every single iteration
$FC(old)$	The old cost given in terms of weighing the requirement stated.
$FC(new)$	The new cost in terms of Fuzzy weighing of the requirement stated.

Abbreviations

FC	Fuzzy Cost
SDLC	Software Development Life Cycle

References

1. Schmidt, R.F. (2012). *Software engineering: Architecture-driven software development*. Waltham, Massachusetts: Morgan Kaufman.
2. Saini, G.S.; Dubey, S.K.; and Bharti, S.K. (2016). Fuzzy based algorithm for resource allocation. *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications*. Bhubaneswar, India, 69-77.
3. Kumar, G.; and Bhatia, P.K. (2015). Neuro-fuzzy model to estimate and optimize quality and performance of component based software engineering. *ACM SIGSOFT Software Engineering Notes*, 40(2), 1-6.

4. Mishra, S.; and Sharma, A. (2015). Maintainability prediction of object oriented software by using adaptive network based fuzzy system technique. *International Journal of Computer Applications*, 119(9), 24-27.
5. Konig, M.D.; Battiston, S.; Napoletano, M.; and Schweitzer, F. (2008). On algebraic graph theory and the dynamics of innovation networks. *Networks and Heterogeneous Media*, 3(2), 201-219.
6. Shai, O.; and Preiss, K. (1999). Graph theory representations of engineering systems and their embedded knowledge. *Artificial Intelligence in Engineering*, 13(3), 273-285.
7. Toffetti, G.; and Pezze, M. (2013). Graph transformations and software engineering: Success stories and lost chances. *Journal of Visual Languages and Computing*, 24(3), 207-217.
8. Viet, H.H.; Trang, L.H.; Lee, S.; and Chung, T. (2016). A bidirectional local search for the stable marriage problem. *Proceedings of the International Conference on Advanced Computing and Applications (ACOMP)*. Chan Tho, Vietnam, 18-24.
9. Konig, M.D. (2010). *Dynamic R&D network : The efficiency and evolution of interfirm collaboration networks*. Ph.D. Thesis. Vienna University of Technology, Wien, Austria.
10. Boccaletti, S.; Grebogi, C.; Lai, Y.C.; Mancini, H.; Maza, D. and Lai, Y.-C. (2000). The control of chaos: Theory and applications. *Physics Reports*, 329(3), 103-197.
11. Jifeng, H.; Li, X.; and Liu, Z. (2002). Component-based software engineering - The need to link methods and their theories. *Lecture Notes in Computer Science*, 26 pages.
12. Boehm, B.W. (2005). Value-based software engineering: Overview and agenda. *ACM SIGSOFT Software Engineering Notes*, 28(2), 4.
13. Dyba, T.; Kitchenham, B.A.; and Jorgensen, M.(2005). Evidence-based software engineering for practitioners. *IEEE Software*, 22(1), 58-65.
14. Attri, R.; Grover, S.; and Dev, N. (2013). A graph theoretic approach to evaluate the intensity of barriers in the implementation of total productive maintenance (TPM). *International Journal of Production Research*, 52(10), 3032-3051.
15. Saini, D.K.; and Ahmad, M. (2012). Software failures and chaos theory. *Proceedings of the World Congress on Engineering*. London, United Kingdom, 6 pages.
16. Schweitzer, F.; Fagiolo, G.; Sornette, D.; Vega-Redondo, F.; and White, D.R. (2009). Economic networks: What do we know and what do we need to know? *Advances in Complex Systems*, 12(4), 407-422.
17. Lane, P.C.R.; and Gobet, F.; (2012). A theory-driven testing methodology for developing scientific software. *Journal of Experimental and Theoretical Artificial Intelligence*, 24(4), 421-456.
18. Kelly, D. (2015). Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software. *Journal of Systems and Software*, 109, 50-61.

19. Chong, C.Y.; and Lee, S.P. (2015). Analyzing maintainability and reliability of object oriented software using weighted complex network. *Journal of Systems and Software*, 110, 28-53.
20. Tyagi, K.; and Sharma, A. (2012). A rule-based approach for estimating the reliability of component-based systems. *Advances in Engineering Software*, 54, 24-29.
21. Iwama, K.; and Miyazaki, S. (2008). A survey of the stable marriage problem and its variants. *Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society*. Kyoto, Japan, 131-136.
22. Vukic, Z.; and Velagic, J. (1999), Comparative analysis of mamdani and sugeno type fuzzy autopilots for ships. *Proceedings of the 1999 European Control Conference (ECC)*. Karlsruhe, Germany, 1369-1374.
23. Jassbi, J.; Alavi, S.H.; Serra, P.J.A.; and Ribiero, R.A. (2007). Transformation of a Mamdani FIS to first order Sugeno FIS. *Proceedings of the 2007 International Fuzzy Systems Conference*. London, United Kingdom, 1-6.
24. Amdahl, G.M. (2013). Computer architecture and Amdahl's law. *Computer*, 46(12), 38-46.
25. Parhami, B. (2015). Amdahl's reliability law: A simple quantification of the weakest-link phenomenon. *Computer*, 48(7), 55-58.