# An Efficient Framework for Minimizing Project Complexity using Periodic Software Project Standard based Linear Ranking

NadanaSundaram P V
Research Scholar
Maduraikamaraj University
Madurai-625021

K.Iyakutti, Ph.D.
Professor(Rtd)
Maduraikamaraj University
Madurai-625021.

## ABSTRACT

Application knowledge, skills, and procedure related to software tools have to be examined in a regular basis to identify the complexity related to project management. An Integrated Framework for Risk Response Planning (IF-RRP) was developed for providing support in decision making during project response risk planning. The method, IF-RRP used sequential forward selection greedy algorithm and genetic algorithm and analyzed risk propagation behavior. However, IF-RRP using Structure Matrix though constructed the representative project risk but did not considered the project management on multiple related projects. Speculative Analysis Technique using Awareness Tools (SAT-AT) diagnosed important types of conflicts and risks in the early stage using a tool, Crystal between collaborating team members and classified the risks for maintaining software. Even though, SAT-AT failed in analyzing the complexity of the project in an earlier stage. To reduce the software project complexity on multiple related projects, Periodic Software Project Standard measure based Linear Ranking (PSPS-LR) framework is proposed in this work. Software project on each module is checked to detect the complexity of the software project at an earlier stage. Initially, Periodic Software Project Standard is measured based on the standard matrix and assigning the weight value to the standard matrix. A Deterministic Periodic Quantitative Model based on weighted mean is developed in PSPS-LR framework for the efficient ranking of software project module periodically. Secondly, PSPS-LR framework performs the effective ranking of software project using linear ranking. Linear ranking is performed using Linear algebra $PS^2$-LR framework to assess larger software project with varying module size and therefore to easily measure the complexity ratio. The periodic weight value of the standard matrix together with linear transformation easily computes the project complexity of multiple related projects. Experiment is conducted on factors such as project complexity measure ratio, computational complexity and ranking effectiveness based on the project management level.

**Keywords:** Periodic Software, Linear Ranking, Project Standard measure, Weight Value, Software Project Complexity, Multiple Related Software Module

## 1. INTRODUCTION

Today's competitive software market has evolved as a highly sophisticated environment for software project development. Companies are under sarcastic and immense pressure to structure and provide a competitive provisioning by reducing project complexity and software project latency time while maintaining software project quality. These highly require the software industries to concentrate more on project complexity.

An Integrated Framework for Risk Response Planning (IF-RRP) [1] was designed to provide mechanisms for minimizing and correcting the errors during project management using an efficient matrix representation called structure matrix. Genetic algorithm was also introduced in IF-RRP for large projects. Though risk propagation behavior was analyzed using sequential forward selection greedy algorithm and genetic algorithm, but risk was not considered involving multiple related projects. Speculative Analysis Technique using Awareness Tools (SAT-AT) [2] identified different types of conflicts and risks using Crystal to identify manage and prevent the conflicts. Though computational cost was reduced using SAT-AT, qualitative and quantitative analysis was not provided.

Control Objectives for Information and related Technology (COBIT) [3], designed a framework that helped the organizations to effectively manage and eventually analyze the suppliers in a multi sourced environment. COBIT used key performance indicators (KPIs) to perform quantitative analysis. Though quantitative analysis was included, but was limited to seven business entities.

A new technique to minimize the risk factors with respect to time was analyzed by applying the lean construction principles [4] using last planner system. To minimize the project complexity with respect to software project completion time, Percent Expected Time-overrun (PET) and Percent Plan Completed (PPC) was used. Though very simple and highly efficient, the method was highly complex with respect to cost.

The problem of minimization of automata is considered to be the classic problem in project management, with applications in many areas including efficient processing of text, analysis of image and so on. With the efficient solutions being obtained, interest has still gained in reducing project complexity. To reduce the cost and minimize the project complexity, polynomial-time deterministic finite automaton minimization algorithm [5] was introduced. But the restriction of the work is that it only included deterministic automata resulting in software testing complexity.

Branch Coverage Expectation (BCE) [6] analyzed complexity measures related to program by taking into consideration the most important features of a program. Markov model was used in BCE to estimate the test cases required to reach the coverage level which resulted in the increase of correlation while testing a program by reducing the software testing complexity. However, the software testing complexity was not detected in the early stage. One of the main problems related to the software industry is software project failure. In [7], a new type of log-linear regression model was introduced on the basis of use case point model (UCP) to measure the software project using use case diagrams to measure the software project failure in the early stage.

In this work, focus is made on reducing the software complexity using standard matrix and assigning weight value to the standard matrix. The Deterministic Periodic Quantitative Model produces the efficient ranking of software project module using weighted mean. The analyzed result is used for efficient ranking using linear algebra. Efficient ranking is performed to assess larger software project with differing module size and hence to evaluate the complexity ratio. As a result, periodic weight value of standard matrix in addition to linear transformation easily evaluates the project complexity of multiple related projects in an efficient manner.

The structure of paper is as follows. In Section 1, project complexity with respect to software project with existing works is described. In Section 2, literatures related to decision making and project software risk is elaborated by comparing the current history. Section 3 explains about the proposed work Periodic Software Project Standard measure based Linear Ranking (PSPS-LR) framework with near architecture diagram and algorithmic steps to minimize project complexity. Section 4 analyzes the experimental results and Section 5 provides the result analysis using table and graph values. The conclusion is described in Section 6.

## 2. RELATED WORKS

A complex task in software engineering is effective decision-making. With the decision-maker confronted with different actions, and each action related to several consequences, are highly uncertain in nature. Bayesian Networks (BN) in Software Engineering (SE) [8] was introduced to provide effective decision-making in a relatively lesser amount of time. However, reliability and optimization in decision-making was not included.

Optimizing Strategy Software (OSS) [9] was introduced for repetitive software projects which included similar activities that optimized the utilization of resources in an efficient manner. OSS also minimized the complexity involved in resources, but project complexity and scheduling of project remained an open issue. Project Scheduling Problem (PSB) [10] was addressed using evolutionary algorithms by applying fitness function. Though fitness was used as a function to consider cost and completion time, fitness related to project complexity was unaddressed.

Failure in complex projects was analyzed in [11] using non-linear and non-ergodicity as changing is highly unavoidable. Rework has to be accomplished in order to reduce the complexity that was provided using flash cutover conversions. But to enhance the project management development, several other factors like cost involved, behavioral aspects, software estimation were not considered. In [12], the advantages of estimating the software effort using the strategies related to the specific organizational. The advantages of using this method were to improve behavioral aspects, software estimation and so on and accordingly arriving at the best decision. Though software estimation effort resulted in arriving at best decision but issues related to multiple programs was unaddressed.

Existing complexity parameters addressed was highly based on two important factors, namely, code and cognitive metrics. Software Requirement Based Specification [13] was studied to identify the software complexity using Software Development Life Cycle (SDLC) process. Though complexity was addressed, but did not include multi-participant software model.

A new shift in the information system is agile software development that included mechanisms to provide issues related to multi-participant software model using coordination strategy [14] included two components implicit and explicit. Though coordination effectiveness for multi-participant was included, optimization measures were not provided for handling software complexity. In [15], a new mechanism for optimization using fuzzy logic was introduced that included linguistic quantifiers and reasoning based on analogy in order to increase the performance of the software project effort. Cost optimization was provided, but not for real data and was not based on the requirements.

Software development effort estimation refers to the effort of evaluating the highly significant value to develop a method based on certain inputs including software requirements, size usage of software and so on. In [16] software development effort (SDE) based on requirement was provided. Based on the aforementioned techniques, in this work we provide a periodic software project standard measured based linear ranking to reduce the project complexity.

## 3. PERIODIC SOFTWARE PROJECT STANDARD MEASURE BASED LINEAR RANKING

This section presents the relevant theoretical background for minimizing the project complexity using periodic software project standard measure based linear ranking. First, we present the standard matrix and Deterministic Periodic Quantitative Model to efficiently rank the project module. These standard matrix and Deterministic Periodic Quantitative Model form the basis to assess larger software project, which minimizes the project complexity. Second, linear ranking using Linear algebra measure the project complexity ratio.

The effective management of larger software projects with CASE tools in PSPS-LR framework reduces the complexity ratio and attains higher order of ranking. The PSPS-LR framework starts with higher-order conceptualization (i.e.,) ranking with standard measure of weight value for minimizing the software project complexity. The PSPS-LR framework also attempts to facilitate understanding of the scope and nature of project management without any complexity. Larger software projects complexity measure is represented through inverted triangular shape diagram as depicted in Figure 1.

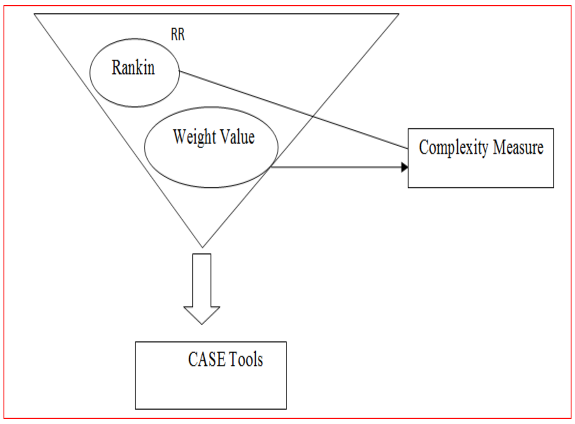


**Figure1 Complexity measure on Software Projects**

The CASE tool at the bottom of the figure is used for the rapid development of software module systems. Each software project module is measured using periodic software action computations. If complexity arises, it is removed during the initial stage using PSPS-LR framework by periodic assessment process. CASE tools are specifically used during the development and execution of larger software project and also during the planning and control processes. The architecture diagram of PSPS-LR framework is depicted in Figure 2.

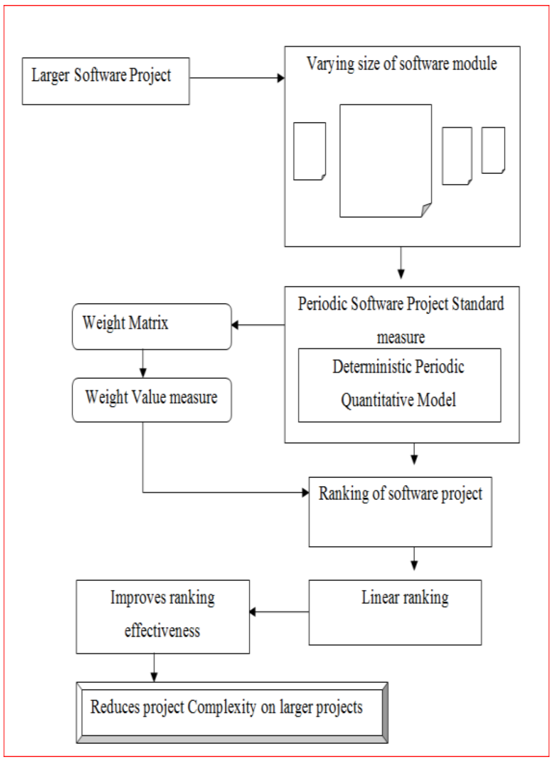

Larger Software Project
Varying size of software module
Periodic Software Project Standard measure
Deterministic Periodic Quantitative Model
Weight Matrix
Weight Value measure
Ranking of software project
Linear ranking
Improves ranking effectiveness
Reduces project Complexity on larger projects

**Figure 2 Architecture Diagram of PSPS-LR framework**

As illustrated in Figure 2, the PSPS-LR framework is initially developed on larger software projects of varying module sizes. Software module with varying sizes is examined periodically to measure the standard. The standard measure then uses the Deterministic Periodic Quantitative Model in PSPS-LR Framework that develops a matrix structure with which it is assigned a weight value to measure the complexity involved in larger software projects. As a result, the software project with weight value helps to design ranking using linear ranking. The linear ranking using the linear transformation handles multiple related software modules. Finally, with the efficient ranking, the complexity measure is reduced.

## 3.1 Periodic Software Project Standard Measure

The PSPS-LR Framework main objective is to depart from the project complexity on the larger software projects. The standard measure is produced with PSPS-LR deterministic periodic quantitative model based on weighted mean for the easier ranking of the larger software projects. The weight of the each standard module is computed and applied that weight mean into the matrix form. The PSPS-LR Framework deterministic periodic quantitative model is described below. The detailed evaluation of periodic software project standard measure using deterministic periodic quantitative model and ranking on PSPS-LR framework using linear ranking algorithm is described in brief in the forthcoming section.

### *3.1.1 Deterministic Periodic Quantitative Model*

Let us consider a matrix for each module assessment with the element 'e' that takes 'y' standard of 'x' model. The matrix for measuring the complexity of each software module is represented as,

$$Standard\ Matrix\ Measure\ (SMM) = \begin{matrix} e_{11} & e_{12} & e_{1m} \\ e_{21} & e_{22} & e_{2m} \\ e_{n1} & e_{n2} & e_{nm} \end{matrix} \quad \ldots\ldots\ldots\ \text{Eqn (1)}$$

The value of project standard matrix measure is placed in the matrix representation. The minimum and maximum value of standard matrix measure obtained during software project assessment is illustrated as,

$$Min\ and\ Max\ of\ Standard\ Matrix\ Measure\ (E_{xy}) = \begin{matrix} E_{min1} & E_{min2} & E_{minn} \\ E_{max1} & E_{max2} & E_{maxn} \end{matrix} \quad \ldots\ldots\ldots\ \text{Eqn (2)}$$

Where Eqn (2), takes $E_{maxn}$ as the maximum value on 'y' standard value to improve the ratio of software project management whereas $E_{minn}$ considers the minimum value on the 'y' standard value attained. The occurrence of complexity is measured periodically using PSPS-LR framework and accordingly rating is performed. The rating of the standard matrix measure is dissimilar for different modules and is therefore determined periodically by substituting two cases.

**Case-I**

When bigger value of the standard represents appropriate result to the actual data (i.e.) best value,

$$Standard\ Rating\ (SR) = \frac{S\ Max\ value\ in\ Storage\ point - Attained\ Standard\ Value}{S\ Max\ value\ in\ Storage\ point - S\ Min\ value\ in\ Storage\ point}$$

**Case-II**

When smaller value of the standard represents appropriate result to the actual data

$$Standard\ Rating\ (SR) = \frac{Attained\ Standard\ Value - S\ Min\ value\ in\ Storage\ point}{S\ Max\ value\ in\ Storage\ point - S\ Min\ value\ in\ Storage\ point}$$

The standard rating '$SR$' of the software holds either the smaller or bigger value which helps to measure the weight mean periodically. The weighted mean on each software module structure is represented as,

$$W_{xy} = 1 - SR \qquad \text{……. Eqn (3)}$$

The weighted mean on the PSPS-LR framework using the CASE tools in the form of weight matrix is represented as,

$$Weight\ Matrix\ (W) = \begin{matrix} W_{11} & W_{12} & W_{1m} \\ W_{21} & W_{22} & W_{2m} \\ W_{n1} & W_{n2} & W_{nm} \end{matrix}$$

…….. Eqn (4)

Then the value of weighted mean, using PSPS-LR framework is computed as,

$$Weighted\ Mean\ (WM) = \frac{\sum_{i=1}^{m} E_{xy}}{\sum_{i=1}^{m} W_{xy}} \qquad \text{……… Eqn (5)}$$

The PSPS-LR Framework measure the value of weighted mean $WM$ by dividing $'E_{xy}'$ and $'W_{xy}'$. The obtained weighted mean is used for efficient ranking of larger software project and measure the complexity of project. With this, the value of higher order of ranking reduces the complexity rate. The implementation of linear ranking using Linear algebra is clearly described in section 1.2.1

## 3.2 Ranking on $PS^2$-LR Framework

Once the weighted mean value is obtained, ranking is performed using Linear algebra to minimize the project complexity. With the weighted mean value varying on each module of larger software project, the software project ranks the system and attains higher feasibility ratio. The algorithmic step of linear ranking in PSPS-LR framework is described as,

### *3.2.1 Linear Ranking Algorithmic Step*

**Input:** Software project of 'n' modules

**Output:** Higher Order Ranking attained with lesser computational project complexity

```
For Module (i=1 to n)
Begin
  Step 1:  Weighted Mean Matrix Value measured
  Step 2: Uses Deterministic Periodic Quantitative model
  Step 3: If exists higher order weighted mean
  Step 4: Then return higher linear ranking result
  Step 5: Else
  Step 6: Lesser ranking result attained on each module
End
```

The PSPS-LR framework linear ranking improves the rate of software project management without any complexity. The weighted mean value of each module helps to easily reduce the latency time in PSPS-LR Framework.

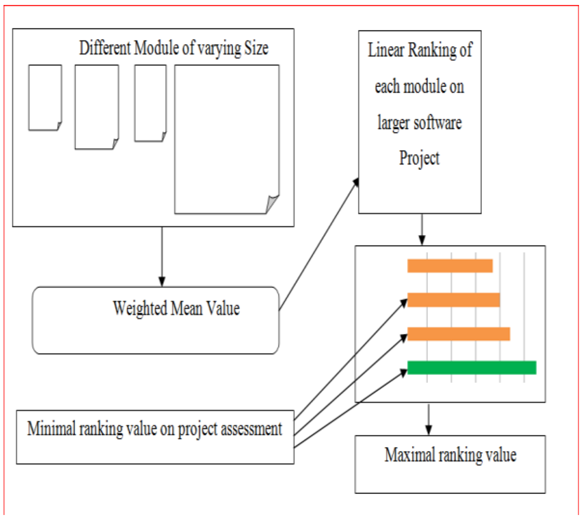

Different Module of varying Size
Linear Ranking of each module on larger software Project
Weighted Mean Value
Minimal ranking value on project assessment
Maximal ranking value

**Figure 3 Linear ranking on PSPS-LR Framework**

The linear arithmetic in PSPS-LR framework ranks the module linearly using the mathematical model as given below. The linear ranking for total size of n, using Linear algebra mathematical function is defined as,

$$\rho = 1 - \frac{6\,(E_{xy} - W_{xy})}{n(n^2 - 1)}$$

$$\rho(x) = \begin{cases} 1, if\ there\ exisits\ high\ Weighted\ Mean\ Value \\ 0, otherwise \end{cases}$$

………Eqn (6)

$\rho(x)$ Measure the linear ranking value. The linear mathematical expression over each module maps the results with effective operations.

## 4. EXPERIMENTAL EVALUATION

Periodic Software Project Standard Measure based Linear Ranking (PSPS-LR) framework is developed in JAVA using Computed Aided Software Engineering (CASE) tools.

The PSPS-LR framework uses JAVA platform by applying Cylinder Bands Data Set extracted from UCI repository where the information measures the complexity level on larger software projects. CASE tools provide the significant measure for module assessment and therefore efficiently identify the complexity level. The software process also ranks the project to identify the order of complexity for different user requests.

The PSPS-LR framework transforms the knowledge acquisition task in which the modules are directly elicited from an expert to identify the project complexity level. The

PSPS-LR framework compares the work with the existing Integrated Framework for Risk Response Planning (IF-RRP) [1] and Speculative Analysis Technique using Awareness Tools (SAT-AT) [2]. The experiment is conducted on the factors such as latency time, project complexity measure ratio, ranking effectiveness and feasibility ratio based on the project management level.

The latency time using PSPS-LR framework measures the interval between the simulation and response to measure the software project complexity given as below which is the summation of optimistic time $OT$ (i.e., the best possible time for ranking software project), pessimistic time $PS$ (i.e., the worst possible time for ranking software project) and the most possible time $MPT$ (i.e., the most possible time required for ranking software project). The latency time is measured in terms of milliseconds (ms).

$$LT = \frac{(OT + PS + [4 * \ MPT\ ])}{6}$$

The project complexity measure ratio in PSPS-LR framework evaluates the complexity using (6) with the help of linear algebra over each module with effective operations. It is measured in terms of percentage (%). Ranking effectiveness using PSPS-LR framework is the ratio of min and max of standard matrix measure and the overall weight matrix as given in (5). Ranking effectiveness is derived in terms of percentage (%).

Given a new software productivity rate as $\beta$, the effort required to complete $\delta$ code is

$$Productivity_{pre} = \frac{\delta}{\beta}$$

With a rework rate of $x$, and the speed required to complete the rework is $\gamma$

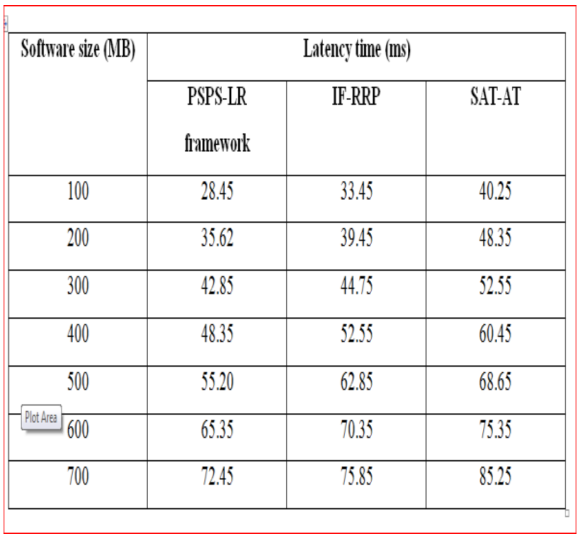
$$Productivity_{post = \frac{x\delta}{\gamma}}$$

Then the feasibility ratio $FR$ (measured in terms of percentage) is the ratio of speed required to complete the rework $\gamma$ and the sum of speed to perform the rework with rework rate $x$ time the new software productivity rate $\beta$ which is given as

$$FR = \frac{\gamma}{\gamma + x\beta}$$

## 5. RESULTS ANLAYSIS OF PSPS-LR FRAMEWORK

The result analysis of Periodic Software Project Standard measure based Linear Ranking (PSPS-LR) framework using Cylinder Bands Data Set extracted from UCI repository is compared with existing Integrated Framework for Risk Response Planning (IF-RRP) [1] and Speculative Analysis Technique using Awareness Tools (SAT-AT) [2]. The table 1 represents the latency time obtained using JAVA and comparison is made with two other methods, namely IF-RRP [1] and SAT-AT [2].



| Software size (MB) | Latency time (ms) | | |
|---|---|---|---|
| | PSPS-LR framework | IF-RRP | SAT-AT |
| 100 | 28.45 | 33.45 | 40.25 |
| 200 | 35.62 | 39.45 | 48.35 |
| 300 | 42.85 | 44.75 | 52.55 |
| 400 | 48.35 | 52.55 | 60.45 |
| 500 | 55.20 | 62.85 | 68.65 |
| 600 | 65.35 | 70.35 | 75.35 |
| 700 | 72.45 | 75.85 | 85.25 |

**Table 1: Tabulation for latency time**

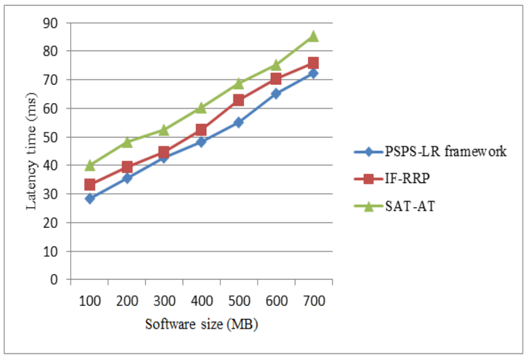


**Figure 4 Measure of latency time with respect to software size**

Figure 4 show that the proposed PSPS-LR framework provides lesser latency time when compared to IF-RRP [1] and SAT-AT [2]. This is because of the application of weighted mean value for each module with respect to the software size in PSPS-LR framework that obtains the values of standard rating based on the attained and maximum standard value. As a result it provides more elaborated information and measure the weight mean periodically improving the latency time by 4 – 17 % when compared to IF-RRP. In addition to that with the use of standard matrix measure based on the minimum and maximum value, the latency rate gets improved by 15 – 41 % than the SAT-AT [2].

**Table 2: Tabulation for Project Complexity Ratio**

| Software size (MB) | Project Complexity Ratio (%) | | |
|---|---|---|---|
| | PSPS-LR framework | IF-RRP | SAT-AT |
| 100 | 41.12 | 45.82 | 50.67 |
| 200 | 54.45 | 59.55 | 65.45 |
| 300 | 61.15 | 65.25 | 69.38 |
| 400 | 56.25 | 61.35 | 71.27 |
| 500 | 63.35 | 67.45 | 77.33 |
| 600 | 58.35 | 62.45 | 72.38 |
| 700 | 66.25 | 70.35 | 76.25 |

The comparison of project complexity ratio is presented in table 2 with respect to the different software size in the range of 100 – 700 MB. Elaborate comparison is made with two other methods, IF-RRP [1] and SAT-AT [2].

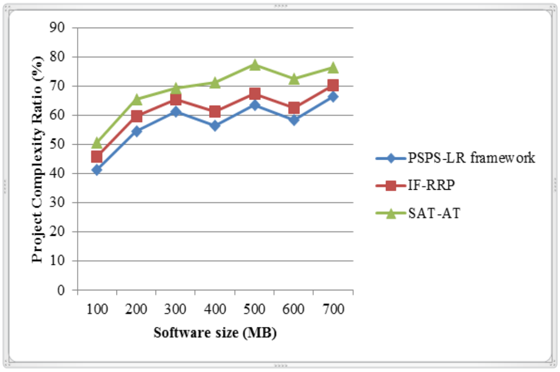


**Figure 5 Measure of Project Complexity Ratio with respect to software size**

To ascertain the performance of the project complexity ratio, comparison is made with two other existing works Integrated Framework for Risk Response Planning (IF-RRP) [1] and Speculative Analysis Technique using Awareness Tools (SAT-AT) [2]. In figure 5, the size of software is varied between 100 and 700 MB for experimental evaluation. From the figure 5 it is illustrative that the project complexity ratio is decreased using the proposed PSPS-LR framework when compared to the two other existing works. This is because with the effective ranking being performed by applying linear algebra which helps in reducing the complexity for larger software project with varying sizes. As a result, the project complexity ratio is improved by 6 – 11% when compared to IF-RRP [1]. Furthermore, by ranking the software project using Deterministic Periodic Quantitative model, the project complexity ratio is decreased using the proposed PSPS-LR framework by 13 – 26 % when compared to SAT-AT [2].

**Table 3: Tabulation for ranking effectiveness**

| Software projects (n modules) | Ranking effectiveness (%) | | |
|---|---|---|---|
| | PSPS-LR framework | IF-RRP | SAT-AT |
| 5 | 75 | 61 | 58 |
| 10 | 77 | 63 | 61 |
| 15 | 74 | 61 | 60 |
| 20 | 79 | 66 | 64 |
| 25 | 82 | 62 | 59 |
| 30 | 84 | 64 | 61 |
| 35 | 81 | 61 | 59 |

The ranking effectiveness for PSPS-LR framework is elaborated in table 3. We consider the method with software projects of size 5 – 35 modules for experimental purpose using JAVA.

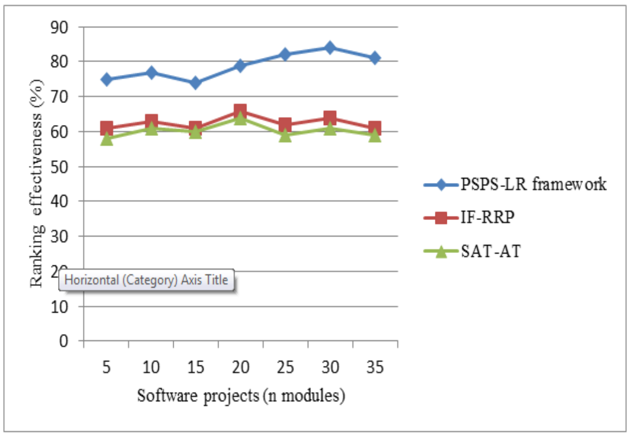


**Figure 6 Measure of ranking effectiveness with respect to software projects**

In figure 6, we depict the ranking effectiveness attained using the software modules of size 5 to 35 for experimental purposes. From the figure 6, the value of ranking effectiveness achieved using the proposed PSPS-LR framework is higher when compared to two other existing works Integrated Framework for Risk Response Planning (IF-RRP) [1] and Speculative Analysis Technique using Awareness Tools (SAT-AT) [2]. Besides we can also observe that by increasing the size of the software modules, the ranking efficiency value is increased using all the methods. But comparatively, it is higher in PSPS-LR framework because of the application of standard matrix that assigns the weight value using Deterministic Periodic Quantitative Model. With the identified weight values, the ranking efficiency of software project improved by 16 – 24 % when compared to IF-RRP [1] and 18 – 27 % than SAT-AT [2] respectively.

**Table 4: Tabulation for Feasibility ratio**

| Software projects (n modules) | Feasibility ratio (%) | | |
|---|---|---|---|
| | PSPS-LR framework | IF-RRP | SAT-AT |
| 5 | 55.35 | 45.25 | 40.05 |
| 10 | 58.25 | 53.15 | 48.15 |
| 15 | 63.55 | 58.45 | 53.35 |
| 20 | 55 | 50 | 48 |
| 25 | 68.45 | 63.35 | 59.45 |
| 30 | 70.05 | 65.05 | 61.35 |
| 35 | 74.55 | 69.45 | 63.75 |

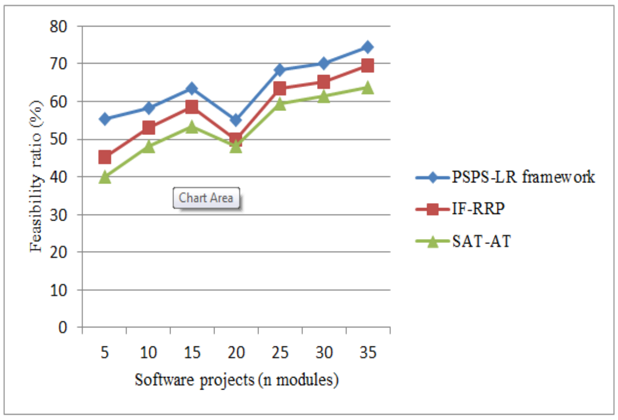


**Figure 7 Measure of feasibility ratio with respect to software projects**

Table 4 and Figure 7 illustrate the feasibility ratio versus the number of software project modules measured in terms of percentage (%) for experimental purpose conducted using JAVA. From the figure we can note that the feasibility ratio attains 9.09 % improvement for the software modules of size 20 when compared to IF-RRP [1] and 12.72 % improvement when compared to SAT-AT [2] which shows that there is a significant gain in terms of feasibility ratio using the proposed PSPS-LR framework. In addition, the project complexity of multiple related software projects is measured with linear algebra that attains higher feasibility ratio by 18 % when compared to IF-RRP and 27.64 % when compared to SAT-AT.

## 6. CONCLUSION

Minimizing the project complexity on multiple related projects has become the key for software project maintenance, to achieve higher ranking effectiveness and improve the level of feasibility ratio with relatively lesser amount of latency time. In this work, we investigate the performance effects of multiple related projects and to minimize the project complexity by proposing a framework, Periodic Software Project Standard measure based Linear Ranking. The PSPS-LR framework based on standard matrix and weight value using Deterministic Periodic Quantitative Model provides an efficient means of software ranking and improves the ranking efficiency. First, we study the use of Deterministic Periodic Quantitative Model that measures the complexity of each software model and propose to use the standard matrix representation for measuring standard matrix measure in PSPS-LR framework. Second, linear ranking is performed using linear algebra to assess larger software project with varying module size to minimize the project complexity using Cylinder Bands Data Set extracted from UCI repository. The experiment conducted using Cylinder Bands Data Set shows that the PSPS-LR framework achieves up to 17.32 percent improvement on project complexity compared to the existing methods.

## 7. REFERENCES

[1] Chao Fang., Marle, F., Min Xie ; Zio, E., “An Integrated Framework for Risk Response Planning Under Resource Constraints in Large Engineering Projects,” Engineering Management, IEEE Transactions on (Volume:60 , Issue: 3 ), 2013

[2] Yuriy Brun, Reid Holmes, Michael D. Ernst, and David Notkin.,“Early Detection of Collaboration Conflicts and Risks,”IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 39, NO. 10, OCTOBER 2013

[3] Thomas Ph. Herz, Florian Hamel, Falk Uebernickel, Walter Brenner,” Toward a model of effective monitoring of IT application development and maintenance suppliers in multi sourced environments”, International Journal of Accounting Information Systems, Elsevier, Jan 2013

[4] Usama Hamed Issa,” Implementation of lean construction techniques for minimizing the risks effect on project construction time”, Alexandria Engineering Journal, Elsevier, July 2013

[5] Manuel Vazquez de Parga, Pedro Garcia, Damian Lopez,” A polynomial double reversal minimization algorithm for deterministic finite automata”, Theoretical Computer Science, Elsevier, July 2013

[6] Javier Ferrer, Francisco Chicano, Enrique Alba,” Estimating software testing complexity”, Information and Software Technology, Elsevier, July 2013

[7] Ali Bou Nassifa, Danny Hob, Luiz Fernando Capretza,” Towards an early software estimation using log-linear regression and a multilayer perceptron model”, The Journal of Systems and Software, Elsevier, Aug 2012

[8] Ayse Tosun Misirli, and Ayse Basar Bener,” Bayesian Networks For Evidence-Based Decision-Making in Software Engineering”,IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 40, NO. 6, JUNE 2014

[9] Remon Fayek Aziz,” Optimizing strategy software for repetitive construction projects within multi-mode resources”, Alexandria Engineering Journal, Elsevier, April 2013

[10] Leandro L. Minku, Dirk Sudholt, and Xin Yao, “Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 40, NO. 1, JANUARY 2014

[11] Kaitlynn M. Whitney, MEM, Charles B. Daniels,” The Root Cause of Failure in Complex IT Projects: Complexity Itself”, Procedia Computer Science 20, Elsevier, Sep 2013

[12] Prabhakar Rao & P Seetharamaiah,” Organizational Strategies and Social Interaction Influence in Software Development Effort Estimation, IOSR Journal of Computer Engineering, Volume 16, Issue 2, Apr 2014

[13] Olabiyisi S.O.1, Adetunji A.B2, Olusi T.R3,” Using Software Requirement Specification as Complexity Metric for Multi-Paradigm Programming Languages”, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 3, March 2013)

[14] Diane E. Strode, Sid L. Huff, Beverley Hope, Sebastian Link,” Coordination in co-located agile software development projects”, The Journal of Systems and Software, Elsevier, Feb 2012

[15] S.Malathia, Dr.S.Sridharb,” OPTIMIZATION OF FUZZY ANALOGY IN SOFTWARE COST ESTIMATION USING LINGUISTIC VARIABLES”, International Conference on Modeling, Optimization and Computing (ICMOC -2012)

[16] Ashish Sharmaa, Dharmender Singh Kushwaha,” Estimation of Software Development Effort from Requirements Based Complexity”, Elsevier, C3IT-2012