International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 7 | ISSN : 2456-3307

Heuristic Correlative Features and Least Square Multi-Layer Perception on Software Process Improvement

Dr. A. Saranya
Assistant Professor, Department of Computer Application V.V.V College for Women, Virudunagar, Tamil Nadu, India

ABSTRACT

Successful software process in its own right not only has various favorable inferences for the software industry, but also the broad stakeholder group.As several software processes exists, it becomes difficult for the project managers to select optimal software process model from available software processes. Improper selection of software process not only increases the software development life cycle time, but also reduces the success rate. Therefore, it is necessary to introduce new and efficient technique to reduce the software development life cycle time with minimum user effort. In this paper, a new attribute based recommendation and machine learning technique called, Heuristic Correlative Features and Least Square Multi Layer Perceptron (HCF-LSMLP) is proposed for helping the project managers to select the most suitable software projects among the existing software projects. This technique introduces a heuristic correlation based feature selection that reduces the software process development cycle time by constructing attributes based on top n recommendations without increasing the computational complexity. Moreover, the proposed HCF-LSMLP technique for suitable software project selection provides better performance in terms of true positive rate than the other existing techniques. Besides, the error on the software process selection is also tackled using the Least Square method in an efficient manner by minimizing the sum of squared residuals. The main advantage of the proposed technique is that it minimizes the software development life cycle time and improves the true positive rate.

Keywords: Software Process, Attribute Based Recommendation, Machine Learning, Heuristic Correlative Features, Least Square, Multi Layer Perceptron

I. INTRODUCTION

In spite of its comparatively young age, the sub-field of Software Process Improvement has improved swiftly during the past two decades, with a growth of new research being published specifically through the previous fifteen years. Its value to the project manager community specifically in nurturing perception and consciousness has been immense and the field has become well established in the research literature.

ArchReco, (a Software Architecture Design prototype tool, which supports Context-Aware recommendations for Design Patterns) was investigated in [1] by applying two types of context-aware recommendations to assists users in increasing their skills related to design while training for High Level Software models. ArchReco used Semantic Web technologies, and Content based analysis for the evaluation of non-personalized recommendations for Design Patterns. This non-personalized recommendation in turn helped the users in identifying the most suitable Design Pattern on the

basis of the working context, deriving the meaning, objectives and usages of each Design Pattern.

ArchReco presented a Semantic Modeling of Software Design process and in specific the characterization of the Design Patterns as Ontology model. This in turn helped in improving the precision and recall rate of, with high percentage of relevant recommended Design Patterns and low selected (recall) value. Despite improvement in precision and accuracy while training for high level software models, prior knowledge of the users were not taken into account for evaluating recommendations. This in turn had negative impact on the software process development cycle time and therefore the success rate.

To address the above said issues, in this work, attribute-based top 'n' recommendations are made using the Heuristic Vector Correlation-based Feature Selection. Here, Heuristic Vector Correlation-based Feature Selection is investigated for evaluating recommendations. Based on the recommendations, relevant attributes are selected, therefore minimizing the dimensionality and in addition reducing the software process development cycle time.

On the other hand, most prior software visualization (SV) research has concentrated mainly on building characteristic of abstract software product artefacts. While unquestionably useful, this cornerstone has revealed that software process visualization has experienced far less consciousness. Conceptual Visualization [2] approach, made the design based on the actual developers concepts and intentions using both notable sources of information in terms of software development and maintenance by combining the code artefacts with their actual functional and developmentaspects.

As a result, the Conceptual Visualization approach assisted the developers and managers to investigate multiple characteristics of software product and process. Despite improvement in the most information rich impression software artefacts while visualizing software process, high involvement of developers and quality engineers were required who cannot model or identify to access hidden insights. This in turn affected thetrue positive rate or the rate of sensitivity, that measured the ratio of positives that are correctly identified as such (e.g. the percentage of most suitable software process which are correctly identified as having the condition).

In this work, a machine learning technique called Multi Layer Perceptron updating the weight using Least Square is presented that requires minimal user effort in selecting software process model. Therefore, true positive rate is said to be increased with minimal user effort. The motivation of this work is based on the lack of software process improvement methods that support new designers and project managers in finding and applying software process in high level software models based on recommendations using Vector Correlation-based Feature and taking into account the recommendations made based on correlations.

In the existing literature there are reported attempts to produce recommendations for software process improvement such as Design Patterns ArchReco, a Software Architecture Design prototype tool, that supports Context-Aware recommendations for Design Patterns [1] and Conceptual Visualization [2] approach but none of them was taking into account the recommendations made with minimal user effort which HCF-LSMLP supports.

The rest of the paper is structured as follows: Section 2 presents the motivation behind the selection for the current work by presenting the related work. Section 3 presents the proposed Heuristic Correlative Features and Least Square Multi Layer Perceptron (HCF-LSMLP) technique. Section 4 provides the

experimental setup followed by detailed discussion in Section 5. Finally, the concluding remarks are presented in Section 6.

II. RELATED WORKS

With the development of information technology, a completely new era has taken place in human life. Because of the evolution of informatics that people use software on every social activity, software has become a part of modern human civilization. Careful software process selection has hence become the need of the hour.

Three different software development methods were compared in [3] with the objective of implementing project managers to select the right method. However, improper features being selected resulted in the increase in software development cycle time. To address this issue, dimensionality reduction was performed using feed forward artificial neural network [4]. Yet another simplified software process improvement framework was designed in [5] using Capability Maturity Model Integration.

Software engineering and evolution is customary tosuffer from unexpected overtime that in turn results in stress and illness in project managers and can lead to poor software quality with higher bugs, therefore compromising the software being selected. Multi-objective approaches were applied in [6] for software process management. Software development models based on agile methods were investigated in [7] that in turn assisted project managers in executing their roles in software process selection.

A systematic literature review was presented in [8] on predicting faults at an early stage. Yet another systematic literature review on software process improvement was investigated in [9]. A predictive performance method was designed in [10] using machine learning in software defect prediction.

Requirements traceability is widely considered as the main constituent of any meticulous software developmentprocess specifically for constructing high andcritical software systems. In [11], traceability was utilized based on multi level Poisson regression analysis that in turn provided an evidence of decrease in the expected defect rate. A new high powered inheritance metrics for object oriented systems was designed in [12] for software quality assessment. A state of the art method for software process improvement was investigated in [13] based on systematic mapping study. Yet another method based on crowd sourcing approach was presented in [14] based on recommendation system to help software engineers in analyzing process management platform.

In the current scenario, several of the software development organizations are applying the concept of Global Software Development(GSD), primarily due to the remarkable return on investment it initiates. Critical Success Factors and Critical Barriers were analyzed in [15] for making comprehensive understanding of software process improvement. Yet another software reliability model was investigated in [16] based on the Non Homogeneous Poisson Process (NHPP) that resulted in the improvement of fault removal efficiency.

A comparative study of software process improvement was analyzed in [17]. The impacts of organizational culture and top management knowledge were critically analyzed by adopting the statistical technique of Partial Least Squares (PLS) in [18]. The impact of Personal Software Process (PSP) on software quality was analyzed in [19]. Software Testing Defect Corrective Model (STFCM) was designed in [20] with the core objective of increasing the quality without increasing the cost and time.

III. METHODOLOGY

In this work, a new technique named Heuristic Correlative Features and Least Square Multi Layer Perceptron (HCF-LSMLP) is presented. This HCF-LSMLP technique is used to train high level software models based on the recommendation with minimal user or programmer effort by modifying the existing context aware recommendation methods [1] for effective selection of software process.

The HCF-LSMLP technique provides various advantages such as minimum user effort, minimal software development life cycle time, low computational cost and complexity for identifying the software process and improved true positive rate.Figure 1 shows the block diagram of HCF-LSMLP.

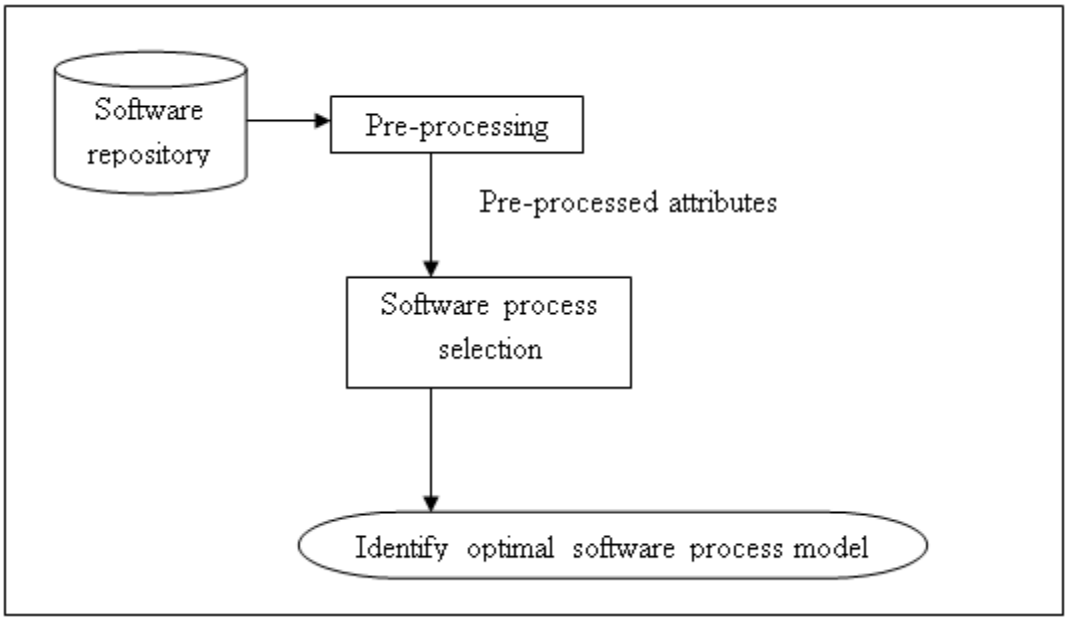

Figure 1. Block diagram of Heuristic Correlative Features and Least Square Multi Layer Perceptron

To start with pre-processing is performed using Heuristic Vector Correlation-based Feature Selection. Here, heuristic functions are used for arriving at the correlation between attributes and to find the top 'n' recommendations based on the selected correlated attributes. Next, a novel machine learning technique, called Multi Layer Perceptron based on the weights using the Least Square is formulated. This HCF-LSMLP has been implemented using JAVA platform and proved that the technique provides better performance when it is compared with the other existing techniques. The elaborate description of HCF-LSMLP technique is given below.

1.1 Heuristic Vector Correlation-based Feature Selection

In this section, Heuristic Vector Correlation-based Feature Selection is applied to the software repository. This is performedwith the objective of reducing the dimensionality and considering prior knowledge of the users while evaluating recommendations with respect to the relevant attributes being selected.

With the application of more sophisticated user based heuristic analysis obtained on result of vector correlation recommendations and the user based feedback for the obtained recommendations, software process development cycle time is also said to be reduced in a significant manner. Figure 2 shows the

method for measuring the Heuristic Vector Correlation-based Feature Selection.

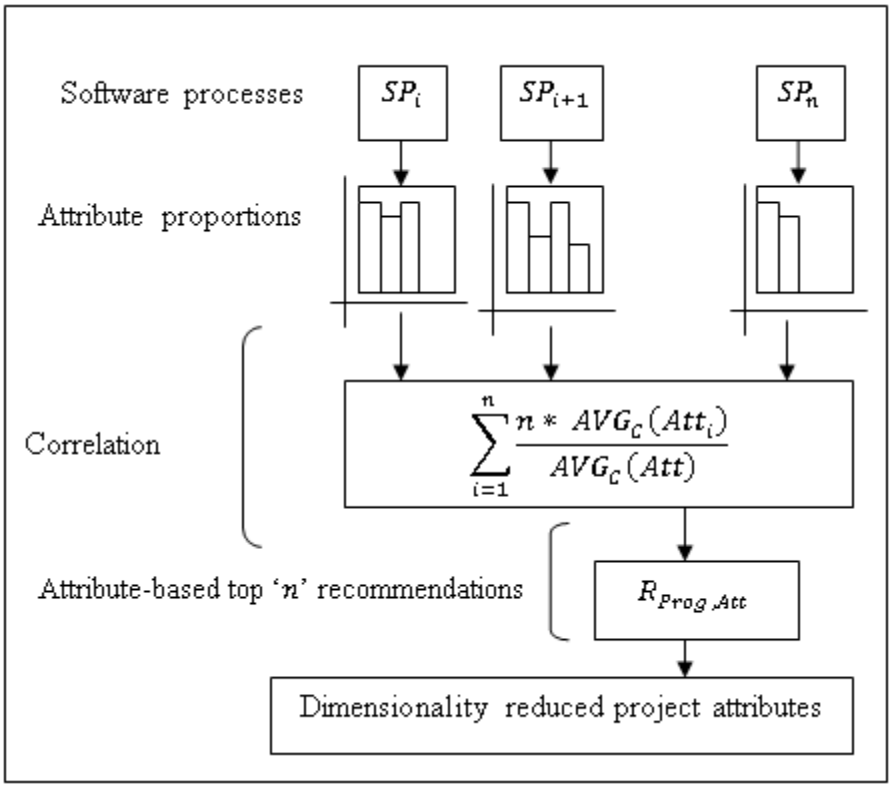

Figure 2. Schematic diagram of Heuristic Vector Correlation-based Feature Selection

The Heuristic Attribute Correlation-based Feature Selection is based on the perception that a useful attribute is one whose values differ consistently with user membership. This relationship between attribute and user is measured through correlation. Besides, an attribute is said to be competent only if it is not redundant. In other words, an attribute is said to be non-redundant only if it is not correlated with another attributes. Based on the above said requirements, a heuristic function is measured using the following formula.

$$H_{SP} = R_{Prog,Att} * \sum_{i=1}^{n} \frac{n * AVG_C(Att_i)}{AVG_C(Att)} \qquad (1)$$

From the above equation (1), the heuristic function for the software process 'H_{SP}' is evolved on the basis of the ' n ' attributes and the ratio of average correlation between each attribute '$AVG_C(Att_i)$' and the average correlation between attributes '$AVG_C(Att)$' respectively.

Besides, from the above equation, greater the relevance value in the numerator and lower the redundant value in the denominator, higher dimensionality reduction is said to be achieved with less effort on the user side. Here, ' $R_{Prog,Att}$ ', represents the Attribute-based top ' n ' recommendations, between programmer and attribute. This measures the similarity between programmers or attributes, with the main objective of making recommendations. The value of '$R_{Prog,Att}$' is measured using the following formula,

$$\sum_{Prog' \in N} R_{Prog,Att} = Sim(Prog, Prog') R_{Prog',Att} \quad (2)$$

From the above equation, 'Prog'' denotes the set of top 'N' programmers (i.e. recommendations made by the programmer) that are most similar to programmer 'Prog' who rated attribute 'Att'. The pseudo code representation of Heuristic Attribute Correlative pre-processing is provided below.

Algorithm 1. Heuristic Attribute Correlative pre-processing algorithm

Input: Programmer '$Prog_1, Prog_2, \dots, Prog_n$', Project attributes '$Att_1, Att_2, \dots, Att_n$'
Output: Pre-processed dimensionality reduced project attributes '$A_1, A_2, \dots, A_n$'
1: **Begin** 2: **For** each Programmer 'Prog' 3: **For** each Project attributes 'Att' 4: Evaluate the heuristic function using equation (1) 5: Evaluate Attribute-based top 'n' recommendations using equation (2) 6: **End for** 7: **End for** 8: **End**

As provided in the above Heuristic Attribute Correlative pre-processing algorithm, for each software processes and attributes, the objective behind the above algorithm lies in extracting the dimensionality reduced attributes. To do this, recommendation methods according to the recommendations made by the programmer is made.

With this goal, correlation measure is used to obtain the average correlation between each attributes and attributes respectively. Followed by this, a heuristic function is measured to arrive at the top 'n' recommendations made by the programmers. With the top 'n' recommendations arrived at, dimensionality reduced project attributes are obtained. Hence, using the dimensionality reduced project attributes, optimal software process is selected by the project managers at an early stage that in turn reduces the software project development cycle time and therefore resulting in the success rate.

3.2 Least Square Multi Layer Perceptron

In this section, with the pre-processed dimensionality reduced attributes, machine learning technique, i.e. Least Square Multilayer Perceptron that requires minimal user effort in selecting software process model is investigated. The objective behind the design of Least Square Multi Layer Perceptron technique is also to reduce the output errors on a particular set of training data (i.e. software processes) by adjusting the network weights 'W' using Least Square method.

Least Square Multi Layer Perceptron comprises of one or more hidden layers, one input and one output layer. Figure 3 given below, show the Multi Layer Perceptron with three hidden layers namely, 'Layer 1', 'Layer 2' and 'Layer 3'.

As illustrated in the above figure, the input layer or input neurons have 'n' nodes or 'SP_n' software processes. No computation is required in the input layer, therefore, the outputs from software processes in the input layer are, 'SP_i', 'SP_{i+1}', 'SP_{i+2}', 'SP_n' respectively, that are fed into the Hidden Layer.

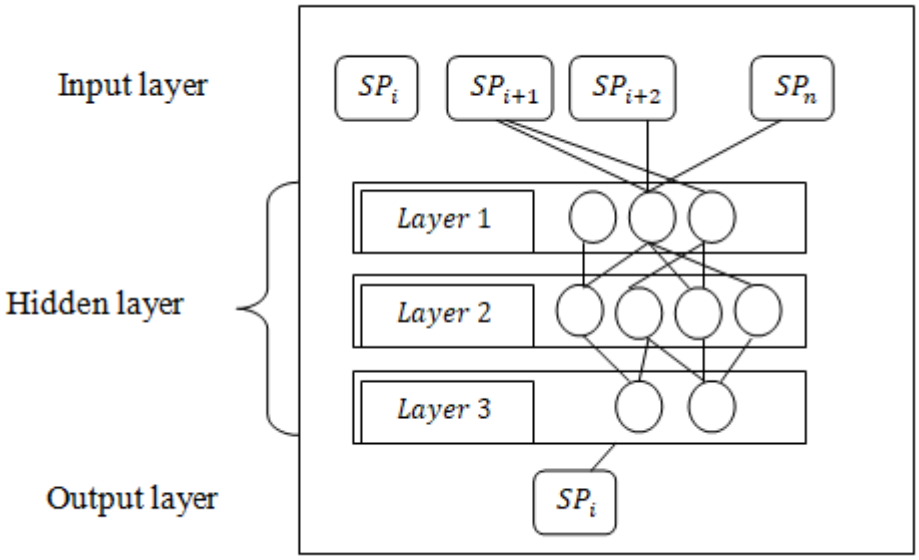

Figure 3. Sample Multi Layer Perceptron with three layers

Hence, the input vector that forms the Multi Layer Perceptron is formulated as given below.

$$\text{fun: } R^p \rightarrow R^q \qquad (3)$$

From the above equation (3), 'p' represents the size of output vector 'D', whereas, 'q' represents the size of vector 'fun(p)'.

$$\text{fun(p)} = A\left[B^{(2)} + W^{(2)}\left(f\left(B^{(1)} + W^{(1)}s\right)\right)\right] \quad (4)$$

From the above equation (4), 'A' represents the activation function, with '$B^{(2)}$' and '$B^{(1)}$' representing the bias vectors and '$W^{(2)}$' and '$W^{(1)}$' representing the weight matrices for the corresponding function 'fun(p)'. Now the hidden layer is evaluated as given below.

$$H(s) = \alpha(s) = f\left(B^{(1)} + W^{(1)}s\right) \qquad (5)$$

From the above equation (5), '$W^{(1)}$', represent the weight matrix connecting input vector to hidden layer. In the proposed work, Learning happens in the perceptron by updating connection weights after each piece of data or software process is processed, based on the error rate in the output layer compared to the anticipated result. For estimating the optimal weights, Least Square method is used.

The purpose of using the Least Square method in measuring the value of weight in the proposed work is that it is entirely free from personal prejudice of the programmer as it is very impartial in nature. The design of Least Square method comprises of altering the parameters 'u_i' and 'v_i', with 'u_i' representing the independent variable and 'v_i' representing the dependent variable so that error minimization is said to be achieved.

The objective is to identify the parameter values for the Least Square Multi Layer Perceptron that results in error minimization. It is measured by its residual, defined as the difference between the actual value of the dependent variable and the predicted value and is mathematically represented as given below.

$$Res_i = v_i - f(u_i, \beta) \qquad (6)$$

Also in the above equation (5) and (6), 'f' represent the tan h function. The purpose of using tan h function in the proposed work is the advantage of being computationally efficient or the computation time for training the test data converges at a faster rate. Hence the use of tan h function is made in the proposed work. The mathematical formulation for tan h is as given below.

$$\tan h(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \qquad (7)$$

Finally, the optimal parameters are identified by minimizing the sum 'LS' of squared residuals:

$$LS = \sum_{i=1}^{n} Res_i^2 \qquad (8)$$

The pseudo code representation of Least Square Multi Layer Perceptron is given below. The following section shows how the outlined method improved the performance of Multi Layer Perceptronthat was trained using the Least Square Multi Layer Perceptron algorithm.

Algorithm 2. Least Square Multi Layer Perceptron algorithm

Input: pre-processed dimensionality reduced attributes '$A_1, A_2, \dots, A_n$'
Output: Computationally efficient software process selection
1: **Begin**
2: **For** each pre-processed dimensionality reduced attributes '$A_1, A_2, \dots, A_n$'
3: Obtain the input vector that forms the Multi Layer Perceptron using equation (3)
4: Obtain the hidden layer using equation (5)
5: Measure the residual value using the equation (6)
6: Compute the tan h function using equation (7)
7: Measure the least square using the equation (8)
8: **End for**
9: **End**

To start with, initially values are selected using the pre-processed dimensionality reduced attributes. Next, the input vector is determined based on the size of output vector ' D '. Then, the activation function in the hidden layer is determined using the tan h function. Finally, the networks are learnt using the Least Square method. This in turn reduces the error while selecting the software process. As a result, the true positive rate is improved, therefore improving the success rate of the software process being selected by the programmers.

IV. EXPERIMENTAL SETUP

The following five open-source programs from http://sourceforge.net are used to ensure optimized selection of software process by the project managers with minimal user effort based on the recommendations. The experiments are conducted using the following information provided in Table 1.

Table 1. Characteristics of software program code

S. No	Program	Description
1	faqforge	a tool for creating and managing documents
2	webchess	an online chess game
3	schoolmate	a solution for administering elementary, middle and high schools
4	time clock	a web-based time clock system
5	phpsysinfo	an utility for providing the system information like CPU time, memory and so on

An effective software validation system using Heuristic Correlative Features and Least Square Multi Layer Perceptron (HCF-LSMLP) technique is developed to experiment the software process selection. The experiment is carried out in JAVA platform using the open-source programs extracted from http://sourceforge.net. The open-source programs from http://sourceforge.net are used to perform the experiment on the factors such as software process development cycle time, true positive rate, computational complexity and success rate.

Software process development cycle time for HCF-LSMLP technique measures the time taken to extract the relevant and dimensionality reduced attributes 'Time(A_i)' and the size of the software program code ' Size(SPC) ' using the heuristic function. It is mathematically formulated as given below.

$$PDCT = \sum_{i=1}^{n} Time(A_i) * Size(SPC) \quad (9)$$

From the above equation (9), the software process development cycle time 'PDCT' is measured in terms of milliseconds. Lower the time taken to extract the dimensionality reduced attributes, lower the software process development cycle time. Therefore, the software process is selected amongst the available software processes at a faster rate.

True positive rate or sensitivity of a test is defined as the ratio of software process model properly selected by the project manager. In other words, a highly sensitive or true positive test is one that correctly identifies the software process to be used.

$$TPR = \sum_{i=1}^{n} \frac{NTP}{Tot(SP_i)} \quad (10)$$

From the above equation (10), the true positive rate 'TPR' is arrived at, using the number of true positives made 'NTP' to the total number of software processes ready for testing 'Tot(SP_i)'. Higher the true positive rate, greater is the possibility for correct software process being selected from the available software processes. It is measured in terms of percentage (%).

Computational complexity or computational time refers to the time taken to perform software process selection. It is mathematically formulated as given below.

$$CT = \sum_{i=1}^{n} SP_i * Time(Sel[SP_i]) \quad (11)$$

From the above equation (11), the computational time 'CT', is measured on the basis of number of software processes ' SP_i ' and the time taken for

software process selection ' Time(Sel[SP_i]) '. It is measured in terms of milliseconds (ms).

Discussion

The result analysis of Heuristic Correlative Features and Least Square Multi Layer Perceptron (HCF-LSMLP) technique using software programs extracted from http://sourceforge.netis compared with existing ArchReco [1] and Conceptual Visualization [2] approach to select optimal software process with minimal user effort. The first set of experiments is performed for software process development cycle time.

The software process development cycle time using HCF-LSMLP refers to the time taken to perform software process development with respect to the software program code. It is represented in terms of milliseconds (ms). The sample calculation for HCF-LSMLP and the existing ArchReco [1] and Conceptual Visualization [2] is provided below.

Sample calculation

- **Proposed HCF-LSMLP**: With size of the software program code being '500MB' and time taken to extract dimensionality reduce attributes being '0.035ms', the software process development cycle time is measured as given below.

$$PDCT = 500MB * 0.035ms = 17.5ms$$

- **ArchReco**: With size of the software program code being '500MB' and time taken to extract dimensionality reduce attributes being '0.049ms', the software process development cycle time is measured as given below.

$$PDCT = 500MB * 0.049ms = 24.5ms$$

- **Conceptual Visualization**: With size of the software program code being '500MB' and time taken to extract dimensionality reduce attributes being '0.058ms', the software process development cycle time is measured as given below.

$$PDCT = 500MB * 0.058ms = 39.5ms$$

Figure 4 reports the software process development cycle time with software program code size in the range of 500MB to 5000MB from webchessextracted from http://sourceforge.net.

As illustrated in the figure, the horizontal 'x' axis represents the software program code and the vertical 'y' axis represents the software process development cycle time. The x axis is represented in terms of megabyte (MB), whereas the y axis is represented in terms of milliseconds (ms). As depicted in the figure, the software process development cycle time is reduced in HCF-LSMLP due to the identification of correlation between the attributes from different software process.With this identified correlations, attribute-based top 'n' recommendations were made. This in turn extracted the dimensionality reduced attributes at a faster rate.

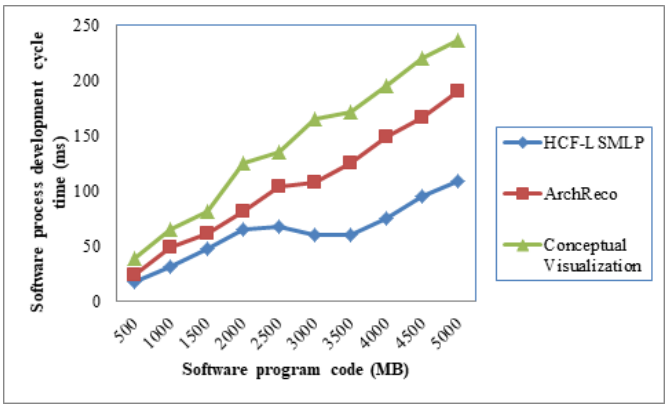

Figure 4. Comparison of software process development cycle with respect to software program code using HCF-LSMLP, ArchReco [1] and Concept Visualization [2]

Due to this, the software process development cycle time is reduced by 37% compared to ArchReco [1]. Besides, by applying the heuristic function based on the recommendations, with greater relevance value

and lesser redundant value, the in the software process development cycle time is reduced by 55% compared to Conceptual Visualization [2].

The second set of experiments is conducted for true positive rate. The true positive rate here refers to the correct identification of the software process by the project managers at an early stage with minimal user effort. It is represented in terms of percentage (%). The sample calculation for HCF-LSMLP and the existing ArchReco [1] and Conceptual Visualization [2] is provided below.

Sample calculation

- **Proposed HCF-LSMLP**: With the number of software processes ready for testing being '10' and the number of true positives being '9', the true positive rate is measured as given below.

$$\text{TPR} = \frac{9}{10} * 100 = 90\%$$

- **ArchReco**: With the number of software processes ready for testing being '10' and the number of true positives being '8', the true positive rate is measured as given below.

$$\text{TPR} = \frac{8}{10} * 100 = 80\%$$

- **Conceptual Visualization**: With the number of software processes ready for testing being '10' and the number of true positives being '7', the true positive rate is measured as given below.

$$\text{TPR} = \frac{7}{10} * 100 = 70\%$$

Figure 5 reports the true positive rate with software process in the range of 10 to 100 using webchess extracted from http://sourceforge.net.

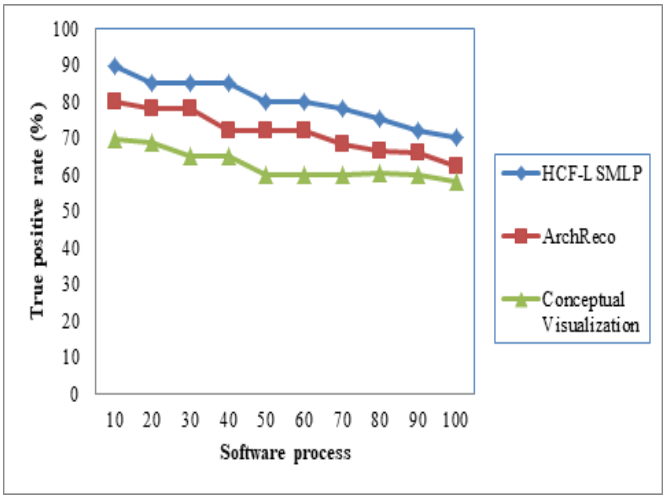

Figure 5. Comparison of true positive rate with respect to software process using HCF-LSMLP, ArchReco [1] and Concept Visualization [2]

As illustrated in the figure, the horizontal 'x' axis represents the software process available and the vertical 'y' axis represents the true positive rate measured using HCF-LSMLP, ArchReco [1] and Concept Visualization [2]. From the figure it is evident that with the increase in the number of available software process, the software process development cycle time is increased and therefore true positive rate is said to be decreasing. Comparatively better performance improvement observed when applied with HCF-LSMLP than [1] and Concept Visualization [2]. This is because of the application of Least Square Multi Layer Perceptron algorithm that initially, performs multi layer perceptrons using the pre-processed attributes. As a result, the size of attributes present in each software process is reduced. This in turn helps in identifying the correct software process amongst the available software processes using HCF-LSMLP by 12% compared to ArchReco [1]. Also, the convergence graph is not said to be linear using all the three techniques. This is because of the differing software process size and therefore non-linearity of convergence graph. The weights in the heuristic function are updated by applying the least square that possess the advantage of identifying the best software

process for a dataset providing a visual demonstration between the software processes. As a result, the true positive rate using HCF-LSMLP is improved by 28% compared to Concept Visualization [2].

The third set of experiments is conducted for computational complexity or computational time. The computational time here refers to the time taken to select optimal software process with minimal user effort. It is represented in terms of milliseconds (ms). The sample calculation for HCF-LSMLP and the existing ArchReco [1] and Conceptual Visualization [2] is provided below.

Sample calculation

- **Proposed HCF-LSMLP**: With the number of software processes ready for testing being '10' and the time of selecting one software process being '0.075ms', the computational time is measured as given below.

$$CT = 10 * 0.075 = 0.75ms$$

- **ArchReco**: With the number of software processes ready for testing being '10' and the time of selecting one software process being '0.089ms', the computational time is measured as given below.

$$CT = 10 * 0.089 = 0.89ms$$

- **Conceptual Visualization**: With the number of software processes ready for testing being '10' and the time of selecting one software process being '0.093ms', the computational time is measured as given below.

$$CT = 10 * 0.093 = 0.93ms$$

Figure 6 reports the computational time with respect to software process in the range of 10 to 100 using webchess extracted from http://sourceforge.net.

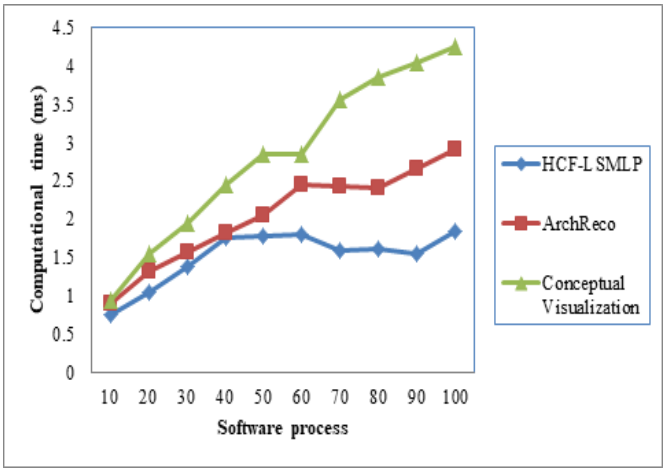

Figure 6 .Comparison of computational time with respect to software process using HCF-LSMLP, ArchReco [1] and Concept Visualization [2]

As illustrated in the figure, the computational time first increases with the increase in the number of software process. By applying HCF-LSMLP, with the software process from 10 to 50, there was an increase in the computational time for obtaining the optimal software process with minimal user effort. On the other hand, by applying ArchReco, with the software process from 10 to 60, there was an increase in the computational time and similarly when applied with Concept Visualization, with the software process from 10 to 40, an increase in the computational time was said to be observed. A swift decrease was then observed using all the three techniques. This is because, with the increase in the software process size, software process development cycle time increases and as a result, the computational time involved in identifying the software process amongst the available software processes also increases. However, a swift improvement is observed by applying HCF-LSMLP. This is because by updating the weights by applying the least square while training multi layer perceptrons, the computation time for training the test data converges at a faster rate. This in turn reduces the computational time by applying HCF-LSMLP with 24% compared to ArchReco and 42% compared to Concept Visualization.

V. CONCLUSION

Software process selection amongst the available software processes with minimal user effort is one of the most pre requisite for project managers for making an effective software processes model in software engineering. Indeed, many techniques have been regularly being investigated by several research persons. However, while investing with high level software models during software process selection hinders the overall performance. In this work, Heuristic Correlative Features and Least Square Multi Layer Perceptron (HCF-LSMLP) technique is presented. The main contribution of the work is that it reduces the software process development cycle time using the Heuristic Vector model. Here, the relevant attributes for any software process is selected on the basis of the correlation between software processes. The design of Heuristic Attribute Correlative pre-processing algorithm in HCF-LSMLP technique top 'n' recommendations made by the programmers were used. Dimensionality reduced attributes was obtained by applying Heuristic Attribute Correlative pre-processing algorithm. Finally, the Multi Layer Perceptron extended with Least Square for updating the weight improves the true positive rate and therefore the success rate resulting in effective outcome. Experiments conducted using JAVA reveals the efficiency of the proposed technique in terms of software process development cycle time, true positive rate and computational time compared to the state-of-the-art works.

VI. REFERENCES

[1]. George A. Sielis, Aimilia Tzanavari and George A. Papadopoulos, "ArchReco: a software tool to assist software design based on context aware recommendations of design patterns", Journal of Software Engineering Research and Development, Springer, May 2017

[2]. Mujtaba Alshakhouri, Jim Buchan, Stephen G. MacDonell, "Synchronised visualisation of software process and product artefacts: Concept, design and prototype implementation", Information and Software Technology, Elsevier, Jan 2018

[3]. Suryanto Nugroho, Sigit Hadi Waluyo, Luqman Hakim, "Comparative Analysis of Software Development Methods between Parallel, V-Shaped and Iterative", International Journal of Computer Applications (0975 – 8887) Volume 169 – No.11, July 2017

[4]. D. Peteiro-Barral, V. Bolón-Canedo, A. Alonso-Betanzos, B. Guijarro-Berdiñas, N. Sánchez-Maroño, "Toward the scalability of neural networks through feature selection", Expert Systems with Applications, Elsevier, June 2012

[5]. Delroy A. Chevers, Annette M. Mills, Evan W. Duggan & Stanford E. Moore, "Toward a Simplified Software ProcessImprovement Framework for Small SoftwareDevelopment Organizations", Journal of Global Information Technology Management, Taylor and Francis Group, June 2017

[6]. Federica Sarro, Filomena Ferrucciy, Mark Harman, Alessandra Mannay, Jian Ren, "Adaptive Multi-objective Evolutionary Algorithmsfor Overtime Planning in Software Projects", IEEE Transactions on Software Engineering, Volume 43, Issue 10, Oct. 1 2017

[7]. Rashina Hoda, James Noble, and Stuart Marshall, "Self-Organizing Roles onAgile Software Development Teams", IEEE Transactions on Software Engineering, VOL. 39, NO. 3, MARCH 2013

[8]. Tracy Hall, Sarah Beecham, David Bowes, David Gray, and Steve Counsell, "A Systematic Literature Review onFault Prediction Performancein Software Engineering", IEEE Transactions on Software Engineering, VOL 38, NO 6, NOVEMBER/DECEMBER 2012

[9]. Michael Unterkalmsteiner, Tony Gorschek, A.K.M. Moinul Islam, Chow Kian Cheng, Rahadian Bayu Permadi, and Robert Feldt, "Evaluation and Measurementof Software Process Improvement—A Systematic Literature Review", IEEE Transactions on Software Engineering, VOL. 38, NO. 2, MARCH/APRIL 2012

[10]. Martin Shepperd, David Bowes, and Tracy Hall, "Researcher Bias: The Use of Machine Learningin Software Defect Prediction", IEEE Transactions on Software Engineering, VOL. 40, NO. 6, JUNE 2014

[11]. Patrick Rempel and Parick Mader, "Preventing Defects: The Impact of RequirementsTraceability Completeness on Software Quality", IEEE Transactions on Software Engineering (Volume: 43, Issue: 8, Aug. 1 2017)

[12]. Neha Gehlot and Jagdeep Kaur, "Dynamic inheritance coupling metric-design andanalysis for assessing reusability", Int. J. Software Engineering, Technology and Applications, Vol. 1, No. 1, 2015

[13]. Marco Kuhrmann, Philipp Diebold and Jürgen Münch, "Software process improvement: asystematic mapping study on the stateof the art", Peer J Computer Science, May 2016

[14]. Mushtaq RazaJoão Pascoal Faria, Luis AmaroPedro Castro Henriques, "WebProcessPAIR: Recommendation System for Software ProcessImprovement", ACM, July 2017

[15]. Arif Ali Khan, Jacky Keung,Shahid Hussain, Mahmood Niazi, Muhammad Manzoor IlahiTamimy, "Understanding Software Process Improvement in GlobalSoftware Development: A Theoretical Framework ofHuman Factors", ACM Symposium on AppliedComputing, Copyright 2017

[16]. Qiuying Li, Hoang Pham, "A testing-coverage software reliability modelconsidering fault removal efficiency and errorgeneration", PLOS ONE | https://doi.org/10.1371/journal.pone.0181524 July 27, 2017

[17]. Mahmood Niazi, "A comparative study of software process improvementimplementation success factors", John Wiley & Sons Limited, Aug 2015

[18]. Jung-Chieh Lee, Yih-Chearng Shiue, Chung-Yang Chen, "Examining the impacts of organizational culture and top management support of knowledge sharing on the success of software process improvement", Computers in Human Behavior, Elsevier, Sep 2015

[19]. Fernanda Grazioli, Diego Vallespir, Leticia Pérez, and SilvanaMoreno, "The Impact of the PSP on Software Quality: Eliminatingthe Learning Effect Threat through a Controlled Experiment", Hindawi Publishing CorporationAdvances in Software EngineeringVolume 2014

[20]. K. Karnavel and R. Dillibabu, "Development and Application of New Quality Model forSoftware Projects", Hindawi Publishing Corporation, The Scientific World JournalVolume 2014