Global Journal of Computer Science and Technology
Volume 11 Issue 21 Version 1.0 December 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Prototype centric (PC) software development process model: A machine learning based Hybrid Software Development Model

By Nabil Mohammed Ali Munassar, Dr. A. Govardhan
*Jawaharlal Nehru Technological University Hyderabad*

Abstract - Here in this paper we propose a Machine learning technique based Hybrid software development process model called prototype centric, in short can refer as PC. The proposed hybrid model works by considering any one or more traditional models as source models. We also conduct empirical study to analyze the performance of the PC over other traditional models that are most frequently quoted in literature.

Keywords : *Hybrid Software Development Method, Conventional Software Development Methods, Agile Software Development Methods, Empirical Studies, Software Engineering.*

GJCST Classification : *D.2.9*

# Prototype centric (PC) software development process model: A machine learning based Hybrid Software Development Model

Nabil Mohammed Ali Munassar[α], Dr. A. Govardhan[Ω]

*Abstract* - Here in this paper we propose a Machine learning technique based Hybrid software development process model called prototype centric, in short can refer as PC. The proposed hybrid model works by considering any one or more traditional models as source models. We also conduct empirical study to analyze the performance of the PC over other traditional models that are most frequently quoted in literature.

*Keywords : Hybrid Software Development Method, Conventional Software Development Methods, Agile Software Development Methods, Empirical Studies, Software Engineering.*

## I. Introduction

All software, especially large pieces of software produced by many people, should be produced using some kind of methodology. Even small pieces of software developed by one person can be improved by keeping a methodology in mind. A methodology is a systematic way of doing things. It is a repeatable process that we can follow from the earliest stages of software development through to the maintenance of an installed system. As well as the process, a methodology should specify what we're expected to produce as we follow the process. A methodology will also include recommendation or techniques for resource management, planning, scheduling and other management tasks. Good, widely available methodologies are essential for a mature software industry. A good methodology addresses the following issues: Planning, Scheduling, Resourcing, Workflows, Activities, Roles, Artifacts, Education. There are a number of phases common to every development, regardless of methodology, starting with requirements capture and ending with maintenance. During the last few decades a number of software development models have been proposed and discussed within the Software Engineering community. With the traditional approach, you're expected to move forward gracefully from one phase to the other. With the modern approach, on the other hand, you're allowed to perform each phase more than once and in any order. [1,10].

## II. Related Work

Conventional heavyweight, document-driven software development methods can be characterized as extensive planning, codified process, rigorous reuse, heavy documentation and big design up front [3]. The conventional methods were predominant in the software industry up until the mid 1990s. Since then, the conventional methods have been replaced by lightweight agile software development methods mostly in small-scale and relatively simple projects. This phenomenon is mainly due to the conventional methods' shortcomings, including a slow adaptation to rapidly changing business requirements, and a tendency to be over budget and behind schedule [3, 6, 9, 15]. The conventional methods also have failed to provide dramatic improvements in productivity, reliability, and simplicity [9].

Some researchers reported that during their project development experience, requirements often changed by 25% or more [5]. An interesting research mentioned that the conventional methods were not initially designed to respond to requirements change occurring in the middle of the development process, and the ability to take action appropriate to the change often determines the success or failure of a software product. According to the Standish Group report , numerous projects with the conventional methods in various industry and government sectors were completed with fewer features and functionalities than specified in the user requirements. It is also a challenge for the conventional methods to create a complete set of requirements up front due to constant changes in the technology and business environments.

Despite the existing shortcomings, the conventional methods are still widely used in industry, particularly, for large-scale projects. The driving force of this broad utilization of the conventional methods comes from their straightforward, methodical, and structured nature [12], as well as their capability to provide predictability, stability, and high assurance [6].

Agile software development methods focus on iterative and incremental development, customer

*Author [α] : PhD Scholar in Computer Science & Engineering Jawaharlal Nehru Technological University Hyderabad Kuktapally, Hyderabad-500085, Andhra Pradesh, India. E-mail : Nabil_monaser@hotmail.com*
*Author [Ω] : Professor of CSE & School of Information Technology Jawaharlal Nehru Technological University Hyderabad Kuktapally, Hyderabad- 500 085, Andhra Pradesh, India.*
*E-mail : govardhan_cse@yahoo.co.in*

collaboration, and frequent delivery through a light and fast development life cycle. There are many positive benefits of the agile approaches. Shorter development cycles, higher customer satisfaction, lower bug rates, and quicker adaptation to rapidly changing business requirements have been reported [6].

## III. Hybrid Software Development Process Model

The proposed hybrid software development process model works as prototype centric with one or more traditional models as source. In short we there after refer as PC. The fig. 1 describes the proposed risk analysis process that mingles with each stage of the SDLC. Here in PC the risk analysis is strategic and supports to predict the risk that influence the cost and targeted outcomes. This prediction can help the experts involved to change the current action to decrease the severity of the risk predicted. Fig. 2 describe the risk analysis strategy proposed as key aspect of the PC. Here in risk analysis process we opt to machine learning technique called support vector machines in short SVM. The Risk analysis stage of the PC targets the SDLC logs available as input to train the SVM for better predictions. The feature extraction process that is part of SVM training process can be done with support of mathematical model called Quantum particle swarm optimization. The usage of these technologies described in fallowing section.

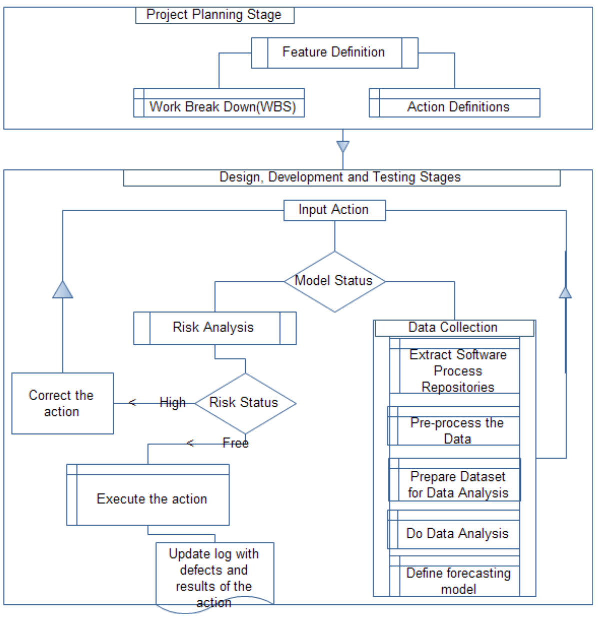


*Fig.1 :* Hybrid Software development process model

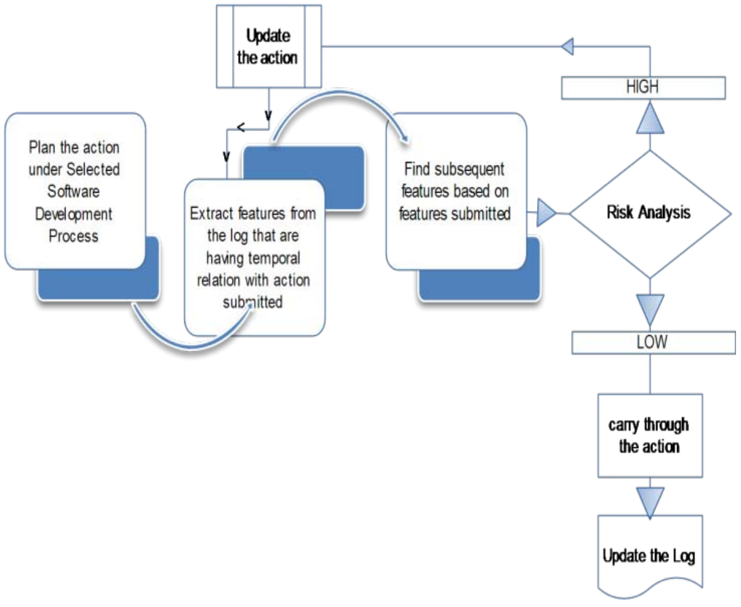

Update the action
HIGH
Plan the action under Selected Software Development Process
Extract features from the log that are having temporal relation with action submitted
Find subsequent features based on features submitted
Risk Analysis
LOW
carry through the action
Update the Log

*Fig. 2 :* Risk Analysis Process

## IV. Risk Analysis Using LS-SVM and Q-QPSO

*a) LS-SVM*

Support vector machine (SVM) introduced by Vapnik[5, 6] is a valuable tool for solving pattern recognition and classification problem. SVMs can be applied to regression problems by the introduction of an alternative loss function. Due to its advantages and remarkable generalization performance over other methods, SVM has attracted attention and gained extensive application[5]. SVM shows outstanding performances because it can lead to global models that are often unique by embodies the structural risk minimization principle[7], which has been shown to be superior to the traditional empirical risk minimization principle. Furthermore, due to their specific formulation, sparse solutions can be found, and both linear and nonlinear regression can be performed. However, finding the final SVM model can be computationally very difficult because it requires the solution of a set of nonlinear equations (quadratic programming problem). As a simplification, Suykens and Vandewalle[8] proposed a modified version of SVM called least-squares SVM (LS-SVM), which resulted in a set of linear equations instead of a quadratic programming problem, which can extend the applications of the SVM. There exist a number of excellent introductions of SVM [8, 9] and the theory of LS-SVM has also been described clearly by Suykens et al[7, 8] and application of LS-SVM in quantification and classification reported by some of the works[10, 11].

In principle, LS-SVM always fits a linear relation $(y = w \quad x + \quad b)$ between the regression (x) and the dependent variable (y). The best relation is the one that minimizes the cost function (Q) containing a penalized regression error term:

$$Q = \frac{1}{2} w^T w + \frac{1}{2} \gamma \sum_{i=1}^{N} e_i^2 \qquad (1)$$

*Subject to*

$$y_{i=w^T\phi(x_i)+b+e_i} \quad i=1,\dots,N \qquad (2)$$

The first part of this cost function is a weight decay which is used to regularize weight sizes and penalize large weights. Due to this regularization, the weights converge to similar value. Large weights deteriorate the generalization ability of the LS-SVM because they can cause excessive variance. The second part of cost function is the regression error for all training data. The relative weight of the current part compared to the first part can be indicated by the parameter 'g', which has to be optimized by the user.

Similar to other multivariate statistical models, the performances of LS-SVMs depends on the combination of several parameters. The attainment of the kernel function is cumbersome and it will depend on each case. However, the kernel function more used is the radial basis function (RBF), a simple Gaussian function, and polynomial functions where width of the Gaussian function and the polynomial degree will be used, which should be optimized by the user, to obtain the support vector. For the RBF kernel and the polynomial kernel it should be stressed that it is very important to do a careful model selection of the tuning parameters, in combination with the regularization constant g, in order to achieve a good generalization model.

### *b) Q- QPSO*

Millie Pant et al[12] attempt to optimize the QPSO by replacing least good swarm particle with new swarm particle. An interpolate equation will be traced out by applying a quadratic polynomial model on existing best fit swarm particles. Based on emerged interpellant, new particle will be identified. If the new swarm particle emerged as better one when compared with least good swarm particle then replace occurs. This process iteratively invoked at end of each search lap.

The computational steps of optimized QPSO algorithm are given by :

*Step 1 :* Initialize the swarm.
*Step 2 :* Calculate mbest
*Step 3 :* Update particles position
*Step 4 :* Evaluate the fitness value of each particle
*Step 5 :* If the current fitness value is better than the best fitness value (Pbest) in history Then Update Pbest by the current fitness value.
*Step 6 :* Update Pgbest (global best)
*Step 7 :* Find a new particle
*Step 8 :* If the new particle is better than the worst particle in the swarm, then replace the worst particle by the new particle.
*Step 9 :* Go to step 2 until maximum iterations reached.

The swarm particle can be found using the following.

| | |
|---|---|
| $t_i = \sum_{k=1}^{3} p_i^2 - q_i^2) * f(r)$ | $p=a, q=b, r=c$ *for* $k=1$; $p=b, q=c, r=a$ *for* $k=2$; $p=c, q=a, r=b$ *for* $k=3$ |
| $t1_i = \sum_{k=1}^{3} p_i - q_i) * f(r)$ | $p=a, q=b, r=c$ *for* $k=1$; $p=b, q=c, r=a$ *for* $k=2$; $p=c, q=a, r=b$ *for* $k=3$ |

$$x_i = 0.5 * (\frac{t_i}{t1_i})$$

In the above math notations 'a' is best fit swarm particle, 'b' and 'c' are randomly selected swarm particles $x_i$ is new swarm particle.

### *c) LS-SVM Regression and QPSO based hyper parameter selection*

Consider a given training set of N data points $\{x_t, y_t\}_{t=1}^{N}$ with input data $x_t \in R^d$ and output $y_t \in R$. In feature space LS-SVM regression model take the form

$$\mathrm{y\ (x) = w^T}\varphi\ \mathrm{(x) + b} \qquad (1)$$

Where the input data is mapped $\varphi(.)$.

The solution of LS-SVM for function estimation is given by the following set of linear equations:

$$\begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & K(x_1,x_1)+1/C & \dots & K(x_1,x_1) \\ . & . & . & . \\ . & . & . & . \\ 1 & K(x_1,x_1) & & K(x_1,x_1)+1/C \end{bmatrix} \begin{bmatrix} b \\ \alpha_1 \\ . \\ . \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} 0 \\ y_1 \\ . \\ . \\ y_1 \end{bmatrix} \qquad (2)$$

Where $\mathrm{K(xi\ ,xj\ )} = \phi(x_i)^T \phi(x_j)^T$ for i, j =1...L

And the Mercer's condition has been applied.

This finally results into the following LS-SVM model for function estimation:

$$f(x) = \sum_{i=1}^{L} \alpha_i K(x, x_i) + b \qquad (3)$$

Where $\alpha$ , b are the solution of the linear system, K(.,.) represents the high dimensional feature spaces that is nonlinearly mapped from the input space x. The LS-SVM approximates the function using the Eq. (3).

In this work, the radial basis function (RBF) is used as the kernel function:

$$k(x_i, x_j) = \exp(-\| x - x_t \|^2 / \sigma^2)$$

In the training LS-SVM problem, there are hyper-parameters, such as kernel width parameter σ and regularization parameter C, which may affect LS-SVM generalization performance. So these parameters need to be properly tuned to minimize the generalization error. We attempt to tune these parameters automatically by using QPSO.

*d) Hyper-Parameters Selection Based on Q-QPSO*

To surpass the usual L2 loss results in least-square SVR, we attempt to optimize hype parameter selection.

There are two key factors to determine the optimized hyper-parameters using QPSO: one is how to represent the hyper-parameters as the particle's position, namely how to encode [13,14]. Another is how to define the fitness function, which evaluates the goodness of a particle. The following will give the two key factors.

i. *Encoding Hyper-parameters*

The optimized hyper-parameters for LS-SVM include kernel parameter and regularization parameter. To solve hyper-parameters selection by the proposed Q-QPSO, each particle is requested to represent a potential solution, namely hyper-parameters combination. A hyper-parameters combination of dimension m is represented in a vector of dimension m, such as $x_i = (\sigma, C)$.The resultant Hyper-parameter optimization under Q-QPSO can found in fallowing the Eq. (4).

*a. Fitness function*

The fitness function is the generalization performance measure. For the generation performance measure, there are some different descriptions. In this paper, the fitness function is defined as:

$$fitness = \frac{1}{RMSE(\sigma, \gamma)} \qquad (4)$$

Where RMSE $(\sigma, \gamma)$ is the root-mean-square error of predicted results, which varies with the LS-SVM parameters $(\sigma, \gamma)$. When the termination criterion is met, the individual with the biggest fitness corresponds to the optimal parameters of the LS-SVM.

There are two alternatives for stop criterion of the algorithm. One method is that the algorithm stops when the objective function value is less than a given threshold ε; the other is that it is terminated after executing a pre-specified number of iterations. The following steps describe the Q-QPSO-Trained LS-SVM algorithm:

1) Initialize the population by randomly generating the position vector iX of each particle and set iP = iX;
2) Structure LS-SVM by treating the position vector of each particle as a group of hyper-parameters;
3) Train LS-SVM on the training set;
4) Evaluate the fitness value of each particle by Eq.(4), update the personal best position iP and obtain the global best position gP across the population;
5) If the stop criterion is met, go to step (7); or else go to step (6);
6) Update the position vector of each particle according to Eq.(7), Go to step (3);
7) Output the gP as a group of optimized parameters.

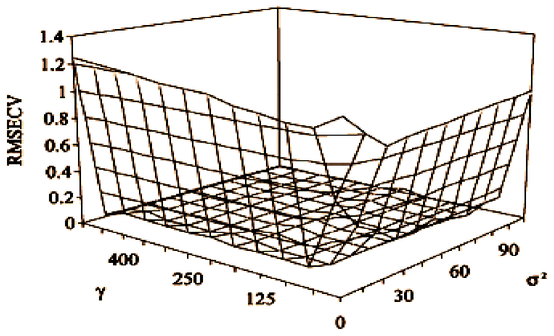


*Fig.3 :* Hyper-Parameter optimization response surface under Q-QPSO for LS-SVM

## V. Risk Analysis Method Proposed

This section explains the algorithm for proposed risk analysis in various stages of SDLC, where the feature extraction can be done under LS-SVM regression and Q-QPSO.

- The SDLC log considered into multitude blocks of SDLC stages.
- Collect the resultant approximate and details features of each block.
- Apply LS-SVM regression under Q-QPSO on each feature matrix that generalizes the training data by producing minimum support vectors required.
- Estimate the features determined levels.
- Apply the risk analysis process by comparing the features of the assigned action and subsequent related actions of the current SDLC stage.
- Identify the risk status

## VI. Empirical Study and Results Discussion

The performance analysis of the proposed Software development process model is carried by conducting empirical study on various projects development process logs. We opted to different logs that belong to applications of different sizes from low to high and enterprise level.

*a) Empirical analysis of the small size software development process logs*

We opted to a small size off the shelf application development process log to analyze the performance of the proposed hybrid software development process model that can referred as prototype centric in short PC. This selected off the shelf product actually developed under waterfall model. We conducted some empirical analysis for waterfall prototyping.

Empirical analysis has been conducted by considering the features of each individual action of each SDLC stage and applied risk analysis process as discussed in section IV. And then we conducted a comparative study between risk status identified and actual impact available in the log. The results that we observed are interesting and concluded that this model is having much influence in SDLC stages

1. Development
2. Testing

Table 1 represents the actual deviation ratio of waterfall model and predicted possible deviation ratio for PC with waterfall model as source. The Fig. 4 indicate the accuracy in risk analysis approach proposed in PC with waterfall as source model. We can observe that the proposed model is impressive at prediction particularly in development testing stages. Therefore we can conclude that PC with Waterfall model as source can minimize the cost and involvement of the high risk.

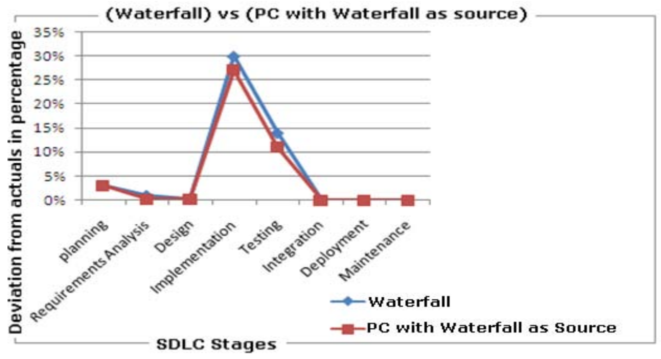
*Table 1:* performance and deviation analysis of PC with Waterfall as source

*(a)* deviation analysis

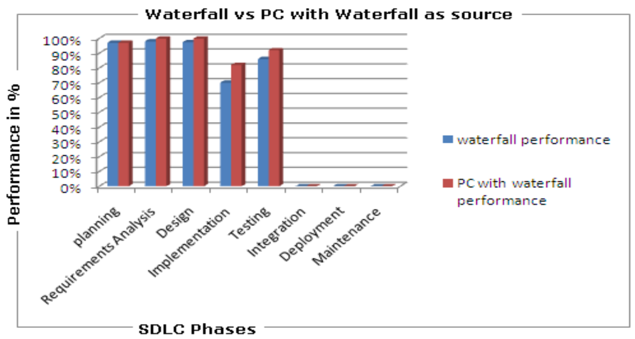
| Waterfall vs PC with Waterfall as source | planning | Requirements Analysis | Design | Implementation | Testing | Integration | Deployment | Maintenance |
|---|---|---|---|---|---|---|---|---|
| Actual deviation | 3% | 1% | 0.30% | 30% | 14% | 0% | 0% | 0% |
| Predicted deviation | 3% | 0.20% | 0.15% | 27% | 11% | 0% | 0% | 0% |

*(b)* Performance analysis

| Waterfall vs PC with Waterfall as source | planning | Requirements Analysis | Design | Implementation | Testing | Integration | Deployment | Maintenance |
|---|---|---|---|---|---|---|---|---|
| waterfall performance | 97% | 98% | 97.40% | 70% | 86% | 0% | 0% | 0% |
| PC with waterfall performance | 97% | 99.80% | 99.85% | 82% | 92% | 0% | 0% | 0% |



*(a)* Deviation Comparison chart



*(b)* Performance Comparison chart

*Fig. 4 :* Deviation ratio and performance ratio of waterfall and PC with Waterfall as source

*b) Empirical analysis of the mid size software development process logs*

We opted to a mid size work flow engine application development process log to analyze the performance of the PC. This selected product actually developed under spiral model with less expertise resources. We conducted some empirical analysis for Spiral prototyping as described in Section IV. And then we conducted a comparative study between risk status identified and actual impact available in the log. The results that we observed are interesting and concluded that this model is having much influence in SDLC stages

1. Design
2. Development
3. Testing

Table 2 represents the actual deviation ratio of spiral model and predicted possible deviation ratio for PC with spiral model as source. The Fig. 5 indicate the accuracy in risk analysis approach proposed in PC with spiral as source model. We can observe that the proposed model is impressive at prediction particularly in design, development and testing stages. Therefore we can conclude that PC with spiral model as source can minimize the cost and involvement of the high risk even under less expertise resource availability. Where in the case of spiral model high expert resources are must to minimize the cost and risk involvement.

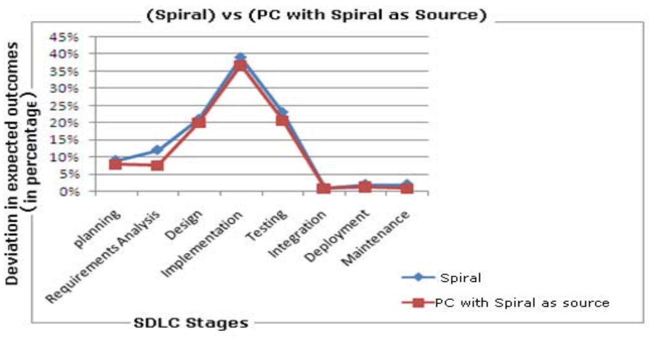
*Table 2 :* Deviation ratio and performance ration of PC with spiral as source

*(a)* Deviation ratio

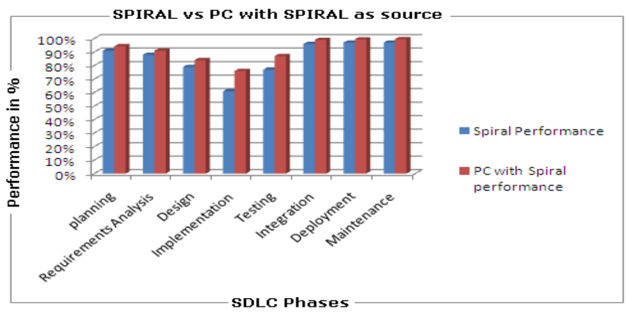
| Spiral Vs PC with spiral as source | planning | Requirements Analysis | Design | Implementation | Testing | Integration | Deployment | Maintenance |
|---|---|---|---|---|---|---|---|---|
| Actual deviation | 9% | 12% | 21% | 39% | 23% | 1% | 2% | 2% |
| Predicted deviation | 7.90% | 7.60% | 20.10% | 36.78% | 20.67% | 0.90% | 1% | 0.90% |

*(b)* Performance ratio

| Spiral Vs PC with spiral as source | planning | Requirements Analysis | Design | Implementation | Testing | Integration | Deployment | Maintenance |
|---|---|---|---|---|---|---|---|---|
| Spiral Performance | 91% | 88% | 79% | 61% | 77% | 96% | 97% | 97% |
| PC with Spiral performance | 94.30% | 91.00% | 84.00% | 76.00% | 87% | 99.00% | 99% | 99.60% |



*(a)* Deviation ratio chart



*(b)* Performance ratio chart

*Fig. 5 :* Deviation ratio and Performance ratio of Spiral and PC with spiral as source

*c) Empirical analysis of the big size software development process logs*

We opted to a big size tailor made java bean framework development process log to analyze the performance of the PC. This selected product actually developed under Incremental model. We conducted some empirical analysis for incremental prototyping as described in section IV. And then we conducted a comparative study between risk status identified and actual impact available in the log. The results that we observed are interesting and concluded that this model is having much influence in SDLC stages

1. Design
2. Development
3. Testing
4. Integration

Table 3 represents the actual deviation ratio of incremental model and predicted possible deviation ratio for PC with incremental model as source. The Fig. 6 indicate the accuracy in risk analysis approach proposed in PC with incremental model as source. We can observe that the proposed model is impressive at prediction particularly in design, development, testing and integration stages. Therefore we can conclude that PC with incremental model as source can minimize the cost and involvement of the high risk. This becomes practical because the proposed model prediction ability of deviations in requirement analysis. Where in the case of incremental model, risk involvement is high since requirement analysis not done in beginning that reflects as high cost and risk involvement.

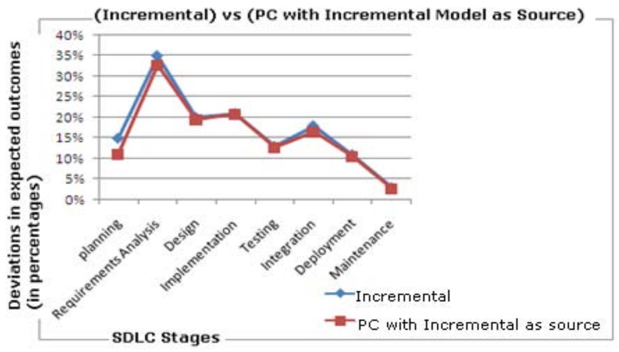
*Table 3 :* Deviation Ratio and performance ratio of Incremental model and PC with incremental model as source

*(a)* Deviation Ratio

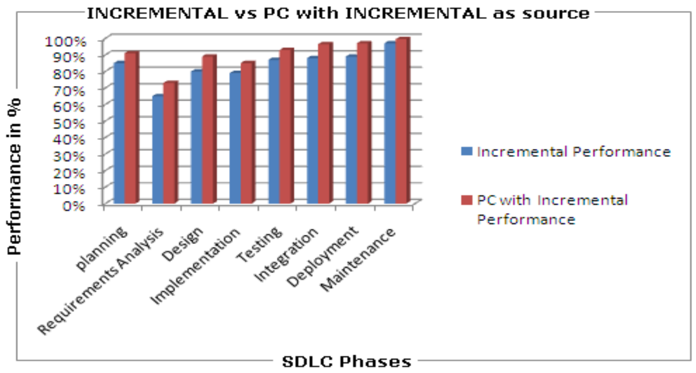
| Incremental Vs PC with incremental as source | planning | Requirements Analysis | Design | Implementation | Testing | Integration | Deployment | Maintenance |
|---|---|---|---|---|---|---|---|---|
| Actual deviation | 15% | 35% | 20% | 21% | 13% | 18% | 11% | 3% |
| Predicted deviation | 11% | 32.70% | 19.40% | 20.80% | 12.70% | 16.50% | 10.60% | 2.70% |

*(b)* Performance Ratio

| Incremental Vs PC with incremental as source | planning | Requirements Analysis | Design | Implementation | Testing | Integration | Deployment | Maintenance |
|---|---|---|---|---|---|---|---|---|
| Incremental Performance | 85% | 65% | 80% | 79% | 87% | 88% | 89% | 97% |
| PC with Incremental Performance | 91% | 73.00% | 89.00% | 85.00% | 93% | 96.50% | 97.00% | 99.50% |



*(a)* Deviation ratio chart



*(b)* Performance Ratio Chart

*Fig. 6 :* Deviation ratio and Performance Ratio of Incremental and PC with Incremental model as source

*d) Empirical analysis of the big enterprise software development process logs*

We opted to a big enterprise MVC based media sharing web application development process log to analyze the performance of the PC. This selected product actually developed under Agile. We conducted some empirical analysis for agile prototyping as described in section IV. And then we conducted a comparative study between risk status identified and actual impact available in the log. The results that we observed are interesting and concluded that this model is having much influence in SDLC stages.

1. Design
2. Development
3. Testing
4. Integration
5. Maintenance

Table 4 represents the actual deviation ratio of incremental model and predicted possible deviation ratio for PC with incremental model as source. The Fig. 7 indicates the accuracy in risk analysis approach proposed in PC with agile model as source. Since agile is combination of iterative and incremental models, so that he advantages of PC with incremental model as source those we observed in earlier section are applicable as it is. We can observe that the proposed model is impressive at prediction particularly in design, development, integration, testing and maintenance stages. Therefore we can conclude that PC with incremental model as source can minimize the cost and involvement of the high risk. This becomes practical because the proposed model prediction ability of deviations in requirement analysis. Where in the case of agile model, risk involvement is high since requirement analysis not done in beginning and that reflects as high cost and risk involvement. It is obvious in agile model that expert resources are must to avoid the project to deviate from the expected outcome. Because of risk analysis and prediction strategy introduced in PC, an interesting issue about PC with agile as source model is that risk involvement can be minimized even under resources with moderated expertise.

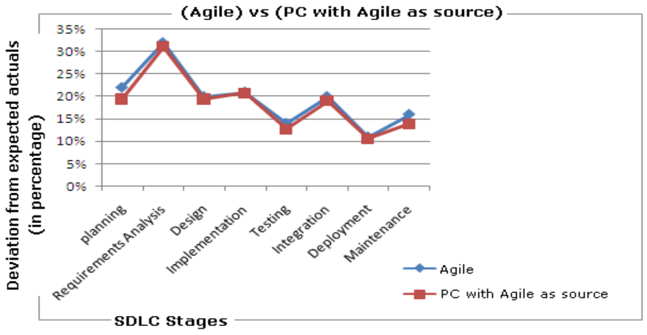
*Table 4 :* Deviation and performance analysis of PC with Agile model as source

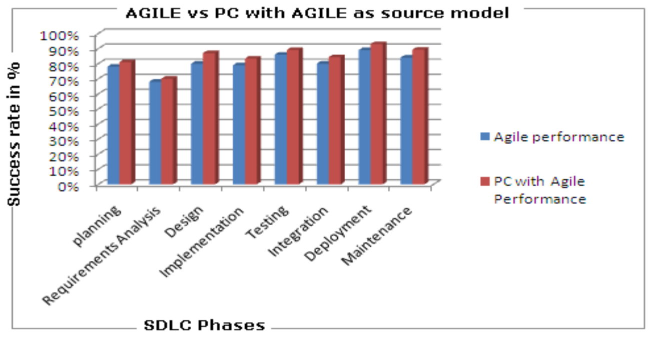
*(a)* Deviation Ratio

| Agile vs PC with Agile as source | Planning | Requirements Analysis | Design | Implementation | Testing | Integration | Deployment | Maintenance |
|---|---|---|---|---|---|---|---|---|
| Actual deviation | 22% | 32% | 20% | 21% | 14% | 20% | 11% | 16% |
| Predicted deviation | 19.40% | 31.10% | 19.40% | 20.80% | 12.70% | 19.04% | 10.60% | 13.90% |

*(b)* Performance Ratio

| Agile vs PC with Agile as source | planning | Requirements Analysis | Design | Implementation | Testing | Integration | Deployment | Maintenane |
|---|---|---|---|---|---|---|---|---|
| Agile Performance | 78% | 68% | 80% | 79% | 86% | 80% | 89% | 86% |
| PC with agile performance | 80.60% | 69.90% | 81.40% | 80.80% | 88.40% | 19.04% | 91.10% | 88.00% |





*(a)* Risk prediction ratio between Agile and PC with Agile as source

*(b)* SDLC phases level Success ratio between Agile and PC with Agile as source

*Fig. 7 :* Risk prediction ratio and SDLC phase level success ratio of PC with Agile model as source

*e) Feature wise performance analysis of existing and proposed software development process models*

*Table 5 :* Comparison report of the existing and proposed Software development process Models

| Feature | Waterfall Model | Prototype Model | Spiral Model | Iterative Model | Agile Model | Prototype Centric(PC) |
|---|---|---|---|---|---|---|
| Requirement Specifications | Beginning | Frequently Changed | Beginning | Beginning | Frequently Changed | Dependent of Risk Analysis report |
| Understanding Requirements | Well Understood | Not Well understood | Well Understood | Not Well understood | Well understood | Well understood |
| Cost | Low | High | Intermediate | Low | Very high | Moderate |
| Guarantee of Success | Low | Good | High | High | Very high | Very high |
| Resource Control | Yes | No | Yes | Yes | No | Yes |
| Cost Control | Yes | No | Yes | No | Yes | Sure |
| Simplicity | Simple | Simple | Intermediate | Intermediate | Complex | Moderate |
| Risk Involvement | High | High | Low | Intermediate | Moderate | Dependent of Source Model |
| Expertise Required | High | Medium | High | High | Very high | Dependant of source model |
| Changes Incorporated | Difficult | Easy | Easy | Easy | difficult | Moderate |
| Risk Analysis | Only at beginning | No Risk Analysis | Yes | No | yes | On each Stage of source model |

| | | | | | | |
|---|---|---|---|---|---|---|
| User Involvement | Only at beginning | High | High | Intermediate | high | Dependent of Risk Analysis report |
| Overlapping Phases | No | Yes | Yes | No | yes | Dependant of source model |
| Flexibility | Rigid | Highly Flexible | Flexible | Less Flexible | highly flexible | Highly Flexible |

i. *Simplicity*

Data was obtained for a cost driver value of 'multi-skilled and experienced'. The data indicates that the waterfall and prototype models are most suitable for projects in which simplicity is the main factor. The spiral and iterative models have limited impact because they have intermediate with regard to simplicity factor and agile is not feasible[15], while the Prototype Centric model is most optimal because of its ability to minimize the complex nature of the source model. But due to modular evaluation, more time and money is required to complete a software project.

ii. *Risk Involved*

The data indicates that the Spiral model is most suitable for projects because software projects using this model involve low risk, where as waterfall model is unsuitable because high risk is involved in software projects. But Prototype centric can be optimal regardless of the source model to minimize the risk.

iii. *Expertise Required*

Data was obtained for a cost driver value of 'range of development experience' The Prototyping models are most appropriate where only developers with a range of experience are available. The waterfall, spiral and iterative models are slightly less suitable because they require personnel with high level of expertise, whereas the agile process model is inappropriate because it requires personnel with very high expertise and experience. The strong positive value for the Prototyping model may suggest the developers, instead of managers, are performing objective setting and evaluation. The proposed Prototype centric PC can improvise the other models performance even under resources with less expertise.

iv. *Changes Incorporated*

From the analysis of data, it is observed that the prototype, spiral and iterative models are most suitable of all as they requires less changes to be incorporated after the project is complete. Because if model needs more changes during usage, software projects takes more cost and also time for its updating etc. While the Waterfall model and agile models are totally inappropriate because if it requires the changes to be incorporated, then many difficulties do arise while incorporating changes in the software project [16].

v. *Risk Analysis*

Data was obtained for a cost driver value of 'risk involvement (expressed as 'complex, difficult or challenging to implement' or 'very complex or novel algorithm'). Data shows waterfall model have risk involved only at beginning, while the prototype model and iterative model don't involves any risk analysis while being used in any software projects. While on the contrary the spiral model and agile process model have risk analysis being used in any software project.

vi. *User Involvement*

Data was obtained and it is observed that waterfall model has very less involvement of the users because it requires user involvement only at the beginning of project. Iterative model needs intermediate user involvement, whereas spiral model and agile process models require high user involvement as a requirement of these models [17].

### *M. Overlapping Phases*

From the research it was seen that Waterfall model and iterative model have no overlapping phases while the prototype model, spiral model process models requires overlapping phases. In the point of prototype centric it is obvious that the behavior of source model need to be considered.

vii. *Flexibility*

Data was obtained for a cost driver value of 'range of flexibility'. Data shows that PC process model and prototype models are highly flexible and are most appropriate, spiral and waterfall models also performs much better when those considered as source process models for PC. As an individual Waterfall model is rigid but as a source model of PC performs better.

## VII. Conclusion

Based on the results of the empirical analysis conducted in section VI, we can conclude that regardless of the source model the Prototype Centric is modest in all desired features, particularly in terms of cost, resource utilization and balanced SDLC. It helps to work with any one or more traditional models as source under any circumstances such as resource availability with less expertise. As the methodology we allowed to perform risk analysis, it is stable regardless of the software application size.

## References References Referencias

1. Molokken-Ostvold et.al, "A comparison of software project overruns - flexible versus sequential development models", Volume 31, Issue 9, Page(s): 754 – 766, IEEE CNF, Sept. 2005.
2. Boehm, B. W. "A spiral model of software development and enhancement", ISSN: 0018-9162, Volume: 21, Issue: 5, on page(s): 61-72, May 1988.

3. Abrahamsson P. et.al, “Agile Software Development Methods: Review and Analysis”, ESPOO, VTT Publications 478, VTT Technical Research Centre of Finland. http:/www.fi/pdf/publications/2002/ P478 .pdf, 2002.
4. Vapnik, V.; Statistical Learning Theory, John Wiley: New York, 1998.
5. Cortes, C.; Vapnik, V.; Mach. Learn. 1995, 20, 273.
6. Sun J, Xu W, Feng B, A Global Search Strategy of Quantum- Behaved Particle Swarm Optimization. In Proc. of the 2004 IEEE Conf. on Cybernetics and Intelligent Systems, Singapore: 291 – 294, 2004.
7. Suykens, J. A. K.; Vandewalle, J.; Neural Process. Lett. 1999, 9, 293.
8. Suykens, J. A. K.; van Gestel, T.; de Brabanter, J.; de Moor, B.; Vandewalle, J.; Least-Squares Support Vector Machines, World Scientifics: Singapore, 2002.
9. Zou, T.; Dou, Y.; Mi, H.; Zou, J.; Ren, Y.; Anal. Biochem. 2006, 355, 1.
10. Ke, Y.; Yiyu, C.; Chinese J. Anal. Chem. 2006, 34, 561.
11. Niazi, A.; Ghasemi, J.; Yazdanipour, A.; Spectrochim. Acta Part A 2007, 68, 523.
12. Millie Pant, Radha Thangaraj, and Ajith Abraham. 2008. A new quantum behaved particle swarm optimization. In <em>Proceedings of the 10th annual conference on Genetic and evolutionary computation</em> (GECCO '08), Maarten Keijzer (Ed.). ACM, New York, NY, USA, 87-94. DOI=10.1145/1389095.1389108 http://doi.acm .org/ 10.1145/1389095.1389108.
13. Liu J, Sun J, Xu W, Quantum-Behaved Particle Swarm Optimization with Adaptive Mutation Operator. ICNC 2006, Part I, Springer-Verlag: 959 – 967, 2006.
14. M. Barni, F. Bartolini, and A. Piva, "Improved Wavelet- Based Watermarking Through Pixel-Wise Masking," IEEE Transactions on Image Processing, Vol. 10, No. 5, IEEE, pp. 783-791, May 2001.
15. Dennis, A., Wixom, B. H. and Tegarden, D. (2002), Systems Analysis and Design: An Object-Oriented Approach, John Wiley & sons, New York.
16. Roger S. Pressman, “Software Engineering a practitioner’s approach”, McGraw-Hill, 5th edition, 200.
17. M M Lehman,”Process Models, Process Programs, Programming Support”, ACM, 1987.