HYBRID SOFTWARE DEVELOPMENT MODEL -PROTOTYPE CENTRIC (PC)

Nabil Mohammed Ali Munassar,
University of Science & Technology - Yemen

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, Software Engineering.

I. INTRODUCTION

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. 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 describes 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.

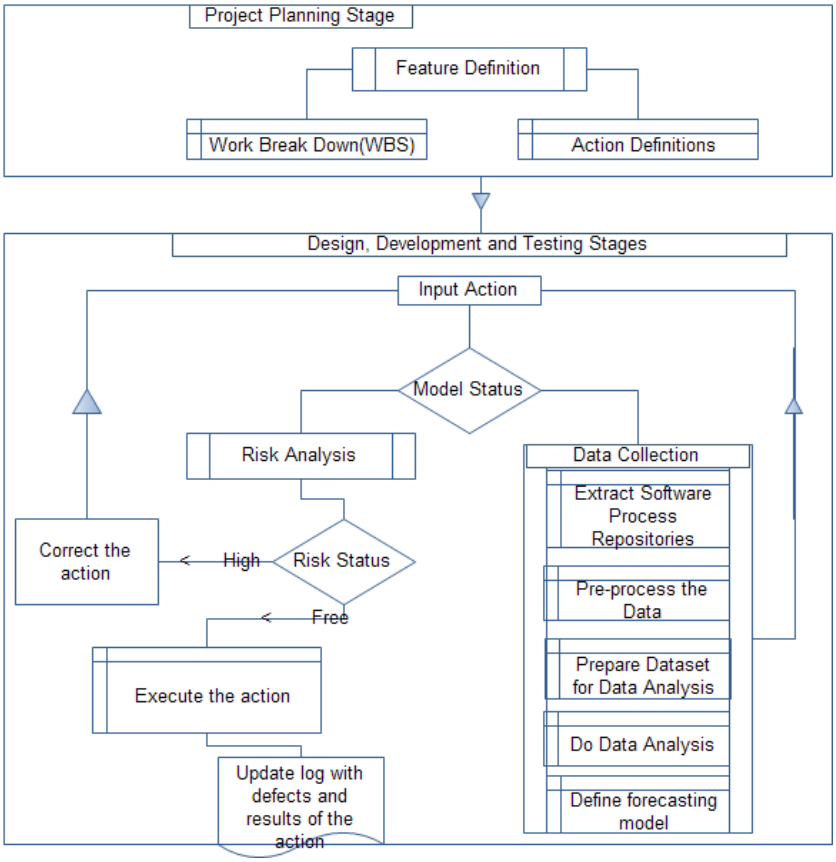

Fig.1 : Hybrid Software development process model

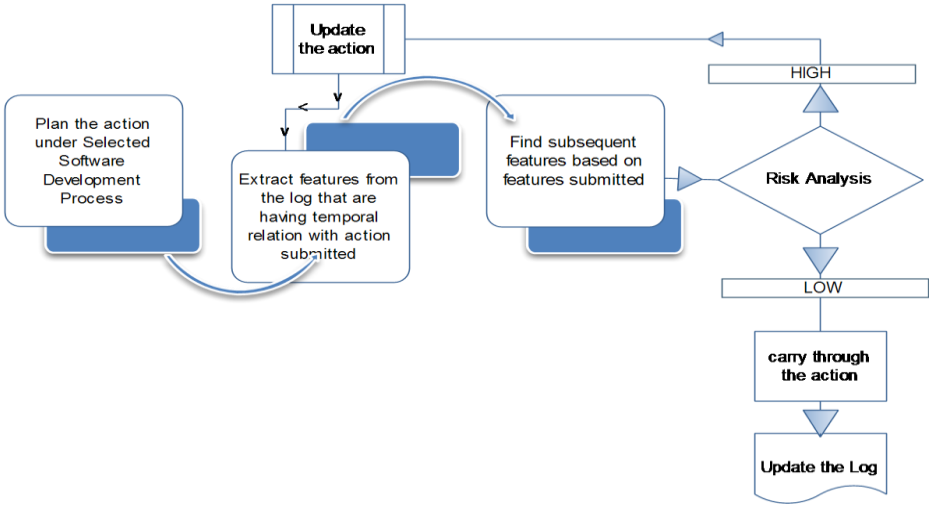

Fig. 2: Risk Analysis Process

III. EMPIRICAL STUDY AND RESULTS DISCUSSION.

- Feature wise performance analysis of existing and proposed software development process models

Here in the Table 1 we described our observations emerged as results to qualitative and quantitative methods.

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

Table 1: Comparison report of the existing and proposed Software development process Models

- 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, while the agile model is unsuitable because of complex nature. Also because of its complexity, more time and money is required to complete a software project [5].
- 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.
- 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.
- 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 [6].
- 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.

- 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 [7].
- 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.
- 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.

IV. CONCLUSION:

As of the reports emerged as results to the empirical analysis, 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 fallowed to perform risk analysis, it is stable regardless of the software application size.

REFERENCES

[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 Proceedings of the 10th annual conference on Genetic and evolutionary computation (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.