A Theoretical Model to Overcome Disruptions of Standard Software Development Models

Nidhi Narula [#1], Saoud sarwar[#2]

[#] *Department of Computer Science & Engineering, Al-Falah School of Engineering & Technology, Faridabad Maharishi Dayanand University, India*

[1]*nidhi.narula08@gmail.com*

***Abstract*— In this paper, we proposed an idea of software development model which tries to overcome the disruptions and discontinuities that can take place during the development of any software project. In this paper we consider the impact(s) they might have on the overall outcome of the project or software and propose a model that can overcome as much disruptions as possible. The disruptions may arise due to transitions between development stages and/or independently of staging. The analysis might investigate individual discontinuities or different categories of discontinuities. Software Development has become a very vital issue in designing and implementing the various software projects.**

Keywords: SDLC, software discontinuities, disruptions.

I. Introduction

Software development has become a very vital issue in designing and implementing the various software projects. Software development is a global activity unconstrained by the bounds of time and space. It is a collaborative effort where groups of developers work together within a global time/space matrix. During such collaboration developers need to keep up their awareness of how well the project is progressing, what the fellow developers are doing, and the current state of resources related to the project [2, 3,4]. The proposed model might not be able to overcome all the disruptions that might take place during the development of a project or software, but at least try to remove as much disruptions as possible. This might increase the complexity of the overall software development model but it will lead us in developing a less risk prone and more efficient project or software. So we must have a better software development process model that will help to overcome these challenges.

II.PAPER ORGANISATION

In section A we will discuss about the problems of Discontinuities in software development. [2, 6]. In Section B, We will discuss the different phases of a rather new and novel software development process model that adopts the modern software development trends and practices CASE STUDY:-[3, 4]

III. SECTION A

1. Software development

Software development may include research, new development, prototyping, modification, reuse, reengineering, maintenance or any other activity that results in software product. The quality control factor during the software development process aims towards a systematic approach to the process of software development

Most of the methodologies used while creating a software or project share some combination of the following stages of the software development:

- Analyzing the problem
- Market research
- Gathering requirements for the proposed business solution
- Design for the software based solution
- Implementation of the software
- Testing the software
- Maintenance and bug fixing

2. *Baselines in Software Development Models*

Baselines are an important part of the systems development life cycle (SDLC) fig 1. These baselines are established after four of the five phases of the SDLC and are critical to the iterative nature of the model. Each baseline is considered as a milestone in the SDLC.

- Functional baseline: established after the conceptual design phase.
- Allocated baseline: established after the preliminary design phase.
- Product baseline: established after the detail design and development phase.
- Updated product baseline: established after the production construction phase.

Fig. 1 Phases involved in Software Development Life Cycle

3. *Disruptions during Software development processes*

The problems associated with the software development process.

I. Larger size.

Ii. Increasing complexity.
Iii. Higher development cost.
Iv. The delivery challenges i.e. Late system delivery.
V. The trust challenge. How much can we trust on system operations?
Vi. Incorrectness: Not satisfying the client needs exactly.
Vii. Poor quality.
Viii. Poor productivity.
Ix. The heterogeneity Challenges i.e. Inter-system coordination problem.
X. Demand of reusability.
Xi. Modularity.
Xii. Maintainability.
Xiii. Integration problem.
Xiv. Scalability.
Xv. Portability.
Xvi. Change Management.
Xvii. Risks associated with software development.

IV. SECTION B

DIFFERENT PHASES OF THE PROPOSED MODEL FOR SOFTWARE DEVELOPMENT PROCESS

After analysing all the disruptions of the existing software development life cycle models, I have proposed a theoretical software development model which will try to overcome as much disruptions as possible in the existing models in fig. 2. As today's software projects require more communication amongst the client and the development team, verification and specification on every major development step, the inter communication among the phases itself and many more; this models tries to incorporate all these factors into it. Different phases of the proposed model are discussed further [5,7,9].

Phase 1: Review and Objective Analysis

This phase documents the market research analysis related with the project and listing all the objectives that are provided by the client which they want to incorporate in their project. This provides a comprehensive but deep study and understanding of the project or software. This provides the client and the development team with a rare level of insight and fact based research into why and how of the project. Review and Objective analysis allows you to document the data that will help the development team to make strategic decisions and to understand the environment.

Phase 2: Requirements Planning, Verification & Specification

This phase includes the software as well as system requirements for the project or software. During the requirement gathering the development team not only establishes the components for building the system, software tools, and other necessary components; but also establishes the expectations for software functionality and identifies which system requirements the software affects.

The main objective of this phase is to collect all the system and software requirements from the user and to list them in a separate document, get it verified from the user and maintain it for future references. This phase also determines the interactions needed with other applications and databases, performance requirements, user interface requirements and so on.

Phase 3: Feasibility Analysis, Risk Assessment & Reduction

The objective of this phase is to take into account the feasibility study of the project as well as to assess if there is any major risk involved in the project and tries to reduce those risks. The main purpose of feasibility study is to check that if the proposed model is economically viable or not. The feasibility study should cover three areas under it: market issues, technical and organisational requirements, and financial overview. The feasibility study allows project managers to investigate the possible negative and positive outcomes of a project before investing too much time and money on it. Risks are assessed and activities are put in place to reduce the key risks. Risks are to be identified, analysed and an alternative plan has to be decided upon for future use. Finally the risk specification document and the feasibility report is maintained for the future references.

Phase 4: System & Software Design, Verification & Specification

This phase determines the system and software framework to meet the specific requirements. This framework defines the major components and the interaction of those components; also it produces specifications of how each component is implemented. This phase involves both high level and low level design activities, i.e.; an abstract design of the overall project as well as the implementation issues related to the design or architecture. The result of this phase will be a finalised design which can be directly implemented in the next phase. After the designing and implementation issues are dealt with, the design is documented and gets verified by the user or the client which is then further used in the next phase.

Phase 5: Coding

The objective of this phase is to write the coding for all the components specified in the previous phase. Each and every component module should be coded properly keeping in mind the standard coding guidelines mentioned by the user or the client or are in the development team's guidelines.

Phase 6: System Building: Automated Testing

After all the components have been coded, it's now the time to build the system by integrating all the components of the system together. While building the system we perform testing. We have chosen automated testing as our testing for the project or software. We

have selected automated testing for the following reasons:

- Using this, it is possible to record test suites and replay them as and when required.
- No human intervention will be needed.
- The speed of test execution is increased.
- Increased test coverage.
- Reduce number of test cases.

The complete build up system is being tested here using automated testing to check the correctness of the system functionality. After the testing is done, a test report is prepared which helps in the validation and maintenance of the system further [1,7,6].

Phase 7: System Validation

The main objective of this phase is to validate all the requirements laid down by the user or the client recorded on the document. The user should be fully satisfied with the working of the project as the project gets validated and user is constantly involved with the project. Each and every function should be exactly implemented as per the user's requirements. The document on which the requirements and objectives of the project are laid should firmly be followed or referred at this point.

After the system validation is completed a validation report has to be prepared for future references.

Phase 8: System Implementation & Verification

Once the system is validated, it is then given to the user for use. The objective of this phase is to safely deliver the working system to its user. The use of verification at this point is that if in case there is a change required by the user in the system then the developer should try and incorporate that change into the system. Delivery of the working system should not just be formality by the developer. This phase goes on till the user gets comfortable with the working of the system efficiently and effectively.

Also its the developer's duty to train the persons whosoever they are within the user's organisation, so that they can work with the system efficiently and effectively.

Phase 9: system maintenance & specification

The maintenance of any system is a lifetime process, which goes on till the system is in effective use within the organization. As the time passes by and the working environment changes the system also needs some or the other changes at some point of time. Moreover, Lehman's first law related to software says," Software product must change continually or become progressively less useful" [5]. The main objective of this phase is to provide constant post-delivery support to the user. The maintenance may be of different type's i.e. Corrective maintenance, adaptive maintenance, perfective maintenance, and preventive maintenance [5, 6]. Every time the maintenance activity is performed on the system, a maintenance report is maintained for future references.

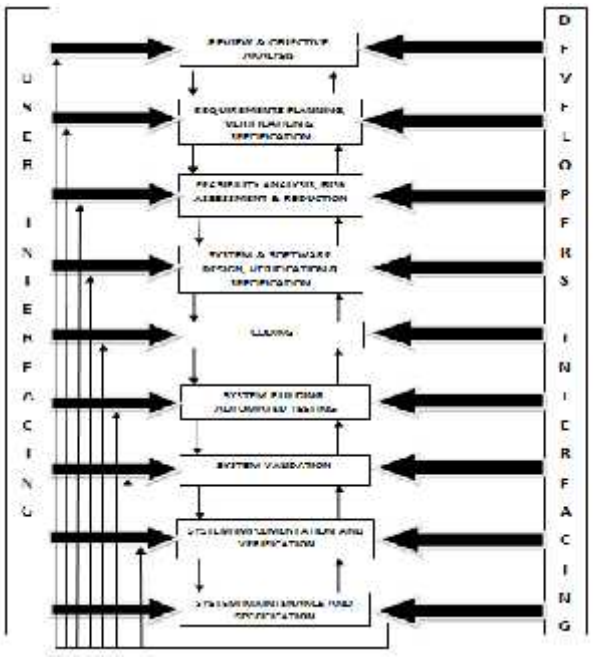

Fig.2 The Proposed software development process model

IV. CONCLUSIONS

During the development of this thesis i expanded my knowledge of the software development field in general. In the early stages of the project i learned a lot about how research works and how to read research papers efficiently. The study process was extremely enjoyable as it was the first real opportunity that i have had to think something based on cutting-edge research. Overall this thesis has been extremely beneficial to me both in technical terms and software engineering terms. The disruptions in the fundamental model of software development at any time of the software project can be easily seen when the software project is tested or environment is dynamically changing or there are some self-adaptive applications that are running in parallel to the software project. We tried to detect some disruptions or discontinuities in the fundamental models of software development employed for making the software projects and proposed a model that will help overcome these disruptions in the software development model. If these disruptions or discontinuities are taken care off then the software projects can be saved from a major software crisis which will enable us to deliver a fully functional system with better quality, further it will help us deliver the software project within time which will serve the right goals. The basic need for removing these disruptions is to make the software development process more useful and yield more benefits.

V. FUTURE SCOPE

This study concluded that the capabilities in the theory of the software development models are weak as compared to the practical processes used while making a software project. This suggests that the future research should be guided such that usefulness of all the capabilities of software development models is considered while developing any software project. As the innovations in the software development processes will take place, the

software projects will give a better return in terms of time, quality; risks involved and will cut on the cost of the project also. Thus the process of innovation becomes more crucial in order to gain more in less from the software projects. The idea of the future research is basically to understand and motivate the goals at each step and of each person involved in the making of software projects.

REFERENCES

[1] Hevner, A., R., S. March, T., et al. (2004). Design Science in Information Systems

[2] Antonia Bertolino, "Software testing research: Achievements,challenges, dreams," In Future of softwareEngineering, 2007.

[3] Brooks, F. P. (1987). "No Silver Bullet: Essence and Accidents of Software Engineering." IEEE Computer 20(4): 10-19.

[4] Research. MIS Quarterly. 28.

[5] Mall R., *Fundamentals of Software Engineering*, PHI, Second Edition, 2008.

[6] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, Mc-Grawhil, Sixth Edition, 2005.

[7] Basili, V. R., R. W. Selby, et al. (1986). "Experimentation in software engineering." IEEE Trans. Softw. Eng. 12(7): 733-743.

[8]http://www.researchgate.net/publication/224247671_Automated_Detection_of_Discontinuities_in_Models_Inferred_from_Execution_Traces/file/d912f510a34654318b.pdf

[9] http://onlinelibrary.wiley.com/doi/10.1002/978047016511.app1/pdf

[10] http://arxiv.org/pdf/1109.1648.pdf

[11] http://hrcak.srce.hr/file/69727

[12] http://www.utwente.nl/mb/iebis/staff/amrit/

[13] http://www.utwente.nl/mb/iebis/staff/amrit/phdthesis.pdf

