Research Journal of Applied Sciences, Engineering and Technology 7(13): 2724-2728, 2014
DOI:10.19026/rjaset.7.592
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2014 Maxwell Scientific Publication Corp.
Submitted: September 09, 2013 Accepted: October 03, 2013 Published: April 05, 2014

Research Article

A New Model for Software Engineering Systems Quality Improvement

Ahmad A. Al-Rababah, Taghreed AlTamimi and Najat Shalash
Department of Computer Science and Software Engineering, Ha'il University, KSA

Abstract: In the continuing effort to improve the system analysis and design process, several different approaches have been developed. This study will propose a new process methodology solves some problems in traditional system development methodologies it will study the strength and limitation of existing system development methodologies from traditional waterfall to iterative model including (Prototyping, Spiral, Rapid Application Development, XP and RUP) to Agility. Propose a new methodology focus on produce a high quality product and suitable for all kind of project. Compare the new methodology with others to view some features that is differentiating it from previous methodologies.

Keywords: Multimedia technology, quality improvement, software process model, software quality, software systems

INTRODUCTION

No model is universally superior. But it is more focus on solving some problems that exist in previous model by proposing new methodology. Waterfall model effect from obligation in consistencies and system obligations "locked in" after being very resolute (can't change) and restricted the user engagement (Aranda *et al*., 2007; Weinberg, 2008). On the other hand, methodology leaps that by using visualize prototyping which gives the user more involvement and this leads to accurate and clear requirement and can be changed by using feedback between communication and prototyping (Janzen and Saiedian, 2005; Aggarwal and Singh, 2005).

RAD methodology: Produces low quality software whereas this methodology produces high quality software because it concentrates in loath analysis and design not only on design at express of detailed analysis as in RAD (Aggarwal and Singh, 2005; Pressman, 2005; Thayer and Christensen, 2005).

Spiral methodology: There is no deadline and cycles continue with no clear termination condition, whereas this methodology has a milestone after each phase and deadline and this will determine accurately the budget and skills which is needed (Dan and Russ, 2008; Wiegers, 2005). Also the spiral need high skilled project manager whereas this new methodology needs less experienced project team and project manager (Gottesdiener, 2005; O'Connor *et al*., 2009).

Agile methodology: As mentioned, success of agile depends on knowledgeable customers and that not an easy task to find such persons especially for complex system. Whereas this methodology doesn't suppose that, it deals with all kinds of customer and agile relies on team work, as opposed to individual role assignment that characterizes this methodology (Schulmeyer, 2008; Pressman, 2005; Lewis, 2005).

RUP methodology: Rup does not state clearly how to deal with non functional requirement whereas this methodology states clearly functioned and non functional requirement (James, 2009; Abrahamson and Baddoo, 2007; Karl, 2005).

MATERIALS AND METHODS

A new proposed model: As the prior section has shown, each system development methodologies from linear waterfall to agility methods have strength and weakness (Pressman, 2005; Pine *et al*., 2008). As a result the situations project needs and constraints that you have to work with, determine the best methodology for the project. But it should keep in mind that there is no methodology is universally superior (Chandrasekhar, 2005; Timothy and Laganière, 2005).

This new methodology has many different phases, it starts with communication between technologist and users, then visualize model which can be mentioned in details anon, so as to start the analysis phase that will take into consideration a discovery of new requirements with its suitable refinements, in order to keep the mandatory part which will be considered in the next

Corresponding Author: Ahmad A. Al-Rababah, Department of Computer Science and Software Engineering, Ha'il University, KSA
This work is licensed under a Creative Commons Attribution 4.0 International License (URL: http://creativecommons.org/licenses/by/4.0/).

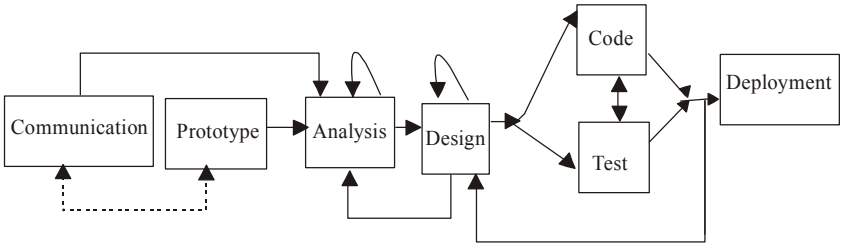

Fig. 1: The proposed model for system quality improvement

phase (design) (Fig. 1), followed by parallel code and check for errors. The latest phase is a deployment (Karl, 2005).

Communication: During the most of the history of software engineering requirements gathering has been considered to be a relatively easy part of the process. However, within the last decade close to, it's become more and more recognized as being the foremost very important a part of the method, as long as the failure to properly determine necessities mixes it just about not possible for the finished piece of computer code to satisfy the wants of the consumer or be finished on time.

This phase provides an overview of entire requirements document. This document describes all Data, functional and behavioral requirements for software.

Goals and objectives: Overall goals and software objectives are described by interviewing stakeholders and users.

Statement of scope: A description of software is presented major input, processing functionality and outputs are described without regard to implementation detail.

Software context: The software is placed in a business or product line context, strategic issues relevant to context are discussed. The intent is for the reader to understand the big picture. Any business or product line constraint that will impact the marver in which the software is to be specified, designed, implemented or tested are noted here.

Visualize prototype: In this methodology it suggests a new type of prototype called visualized prototype which is based on using the concept of multimedia.

According multimedia issue of several different media to Wikipedia, encyclopedia. Convey information by (text, audio, graphics, animation, video and interactivity), "Multimedia technology has played an important role in modern computing because it offers more natural and user friendly interaction with an automated system. This is particularly turn for systems utilizing graphical, icon or window-based input output. Multimedia technology also facilitates "reuse" more naturally, since the basic component and functions of presentation and animation can be reused for several different animation scenario. The visual requirement representation prototype enable the user to view software requirement in addition reading textual representation of the requirement also using up saves development time and money by improving communication and collaboration between customers and software engineer.

"As previous results were: improvement the accuracy and quality of requirement gathering, dramatically reduces project risks, improves developer effectiveness and increases end user satisfaction for delivered application. "Dr. Bony Boehm experiments showed that prototyping reduces program size and programmer effort by 40% and simplify software design". Fully 30 to 40% of system requirement will change without prototyping demonstration to the customer what is functionally feasible and stretches their imagination, leading to more creative input and a more forward looking system.

But in this methodology if the requirements are clear, stable and will not change, there is no need for prototyping, so the coordination rule will be transferred directly from communication to analysis.

Analysis phase: Analysis modeling uses a combination of test and diagrammatic form to depict requirements for dates, function and behavior in a way that is relatively easy to understand and more important straight forward to review for correctness, completeness and consistency. A software engineer (sometimes called and analyst) builds the model using requirements elicited from the customer. To the validate software requirements, it needs to examine these requests from a number of different point of view analysis modeling represents requirements in multiple "dimension", thereby increasing the probability that errors will be found, that inconsistency will surface and that omissions will be uncovered.

Information, function and behavioral requirements are modeled using a number of different diagrammatic formats. Scenario-based modeling represents the system from wer's point of view. Flow-oriented modeling

provides an indication of how data objects are transformed by processing functions. Class-based modeling defines objects, attributes and relations ships. Behavioral modeling depicts the staffs of the system and its classes and the impact of events on these states. Once preliminary models are created, they are refined and analyzed to assess their clarity, completeness and consistency the final analysis model is then validated by stakeholders.

A wide array of diagrammatic forms may be chosen for the analysis model each of these representations provides a view of one or more of the model elements. Analysis modeling work products must be viewed for correctness, completeness and consistency, the must reflect the needs of all stakeholders and establish a foundation from which design can be conducted.

Design phase: Design is what virtually every engineer wants to do. It is the place where creativity rules- where customer requirements, business needs and technical consideration all come together in the formulation of a product or system.

Design creates a representation or model of the soft ware, but unlike the analysis model (that focuses on describing required data, function and behavior), the design model provides detail about software date structures, architecture, interfaces and components that are necessary to implement the system. Software engineers conduct each of the design tasks why is it important. Design allows a software engineer to model the system or product that is to be built. This model can be assessed for quality and improved before code is generated, tests are conducted and end-users become involves in large numbers, Design is the place where software quality is established.

Design steps: Design depicts the software in a number of different ways. First, the architecture of the system or product must be represented. Then, the interfaces that connect the software to end-users, to other systems and devices and to its own consistent components are modeled. Finally the software components that are used to construct the system are designed.

Each of these views represents a different design action, but all must conform to a set of basic design concepts that guide all software design work.

What is the work product? A design model that encompasses architectural, interface, component-level and deployment representation is the primary work product that is produces during software design.

Implementing the design: The design model is assessed by the software team in an effort to determine whether it contains errors, inconsistencies, or missions; whether better alternatives exist; and whether the model can be implemented within the constraints schedule and cost that have been established.

Parallel coding and quality testing: Coding is the process whereby the physical design specification created by the analysis team are turned into working computer code by the programming team depending on the size and complexity of the system, coding can be an involved intensive activity.

As each program module is produced it can be tested individually, then as part of the larger program and then as apart to a larger system. The general idea is that code is tested after it is written. If the code passes the test, then it is integrated into the system. If it doesn't pass, the code is reworked until it does pass.

Parallels helps to take small steps when writing software, a practice that we've promoted for years-baby steps are far more productive than headlong leaps. For example, assume that add some new functional code, compile and test it. It's likely to be tests will be broken by defects that exist in the new code. To find and fix those defects, it's much easier to hunt through two instead of 2,000 new lines of code. The faster compiler and regression test suite, the more attractive it is to proceed in smaller and smaller steps. Generally it prefers to add a few new lines of functional code, typically fewer than 10, before recompile and rerun tests.

RESULTS AND DISCUSSION

Deployment: The Deployment activity encompasses three actions: delivery, support and feedback. There are two type of coordination rule in this model:

First: Feedback from phase to another phase, which means return back from design to analysis or from code to design. The user community needs to be actively involved throughout the project. And this involvement may be a positive for the project, once victimization this sort, communication and coordination skills take center stage in project development and this what we wish to realize, the requests for improvement once every section cause a lot of qualities method, the feedback will cause "scope creep", finding and fixing software system downside once the delivery of system is usually meet a hundred times dearer than finding and fixing it throughout analysis and style.

Second: Recursive rule to the same phase, which means that, involves repeated cycles of analysis, design and test. Reanalysis redesign and recoded parallel with retest until the system meets its usability goals and is ready for release. The goal for the redesign, reanalysis

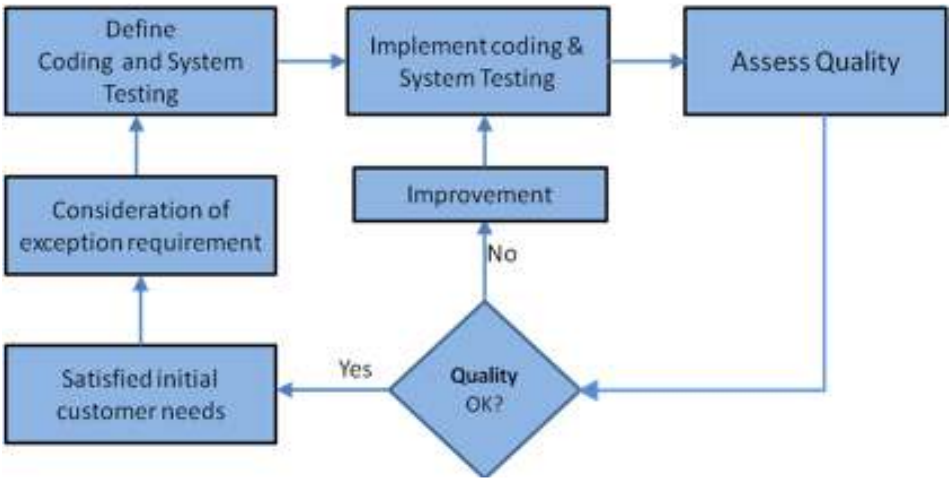

Fig. 2: Parallel coding and quality testing

and recode of any iteration is to be simple, straightforward and modular. The iteration is based upon user feedback, or error discovered or to achieve better (structure, modularity, usability, reliability, efficiency and achievement of goals). Design modifications are created and new purposeful capabilities are supplementary. Finding and fixing the error through the same phase, more cheaply and better than fixing them after leaving the phase to another phase. It allows more involvement to user who is the core of project. Reanalysis allows user validation prior to design, so the design will be better.

Sometime the iterative on the same phase needed any difficulty in design; coding and testing a modification should signal the need for redesign or re-coding. Modifications ought to match simply into isolated and easy-to-find-modules. If they are doing not, some plan is required. Modifications to tables ought to be particularly simple to create. If any table modification isn't quickly and simply done, plan is indicated. Modifications ought to become easier to create because the method progress. If they're not, there's a basic downside like a style flaw or a proliferation of patches therefore the plan is required. Notice that, it's important to stay every development iteration or feedback on the right track and practicality might have to be born to stay development inside the time box. Management plays a vital part in ensuring everything is progressing according to schedule, keeping the customer in involvement regarding changes in the functionality and keeping the team motivate. Implementation of quality ensures that the system must be completed depending on agreed specifications, the standard procedures and tasks without errors or problems probabilities. The main advantages are expected to get effectiveness services before production and deployment should find fault before the application and the impact on business. This minimizes disruptions; reduce the cost of fixing defects and errors that will lead to a qualified product Fig. 2.

CONCLUSION

This study introduced a new type of methodology that treats some weakness in previous methodologies. In this development methodology once coding has begun, the testing can begin and proceed in parallel. The prototype provided an analysis test bed and vehicle to validated and evolve system requirement. At all iterations on the same phase, the quality of software is increased and that what will be achieved in this methodology.

REFERENCES

Abrahamson, P. and N. Baddoo, 2007. Software process model. Proceeding of 14th European Conference. Euro SPI Potsdam, Germany.

Aggarwal, K.K. and Y. Singh, 2005. Software Engineering. 2nd Edn., New Age International Publishers, New Delhi.

Aranda, J., S. Easterbrook and G. Wilson, 2007. Requirements in the wild: How small companies do it. Proceeding of 15th IEEE International Requirements Engineering Conference (RE 2007), pp: 39-48.

Chandrasehakhar, K., 2005. Software Engineering & Quality Assurance. BPB, 2005.

Dan, P. and M. Russ, 2008. Head First Software Development. O'Reilly, Sebastopol, CA.

Gottesdiener, E., 2005. The Software Requirements Memory Jogger: A Desktop Guide to Help Software and Business Teams Develop and Manage Requirements. GOAL/QPC, Salem, NH.

James, A.W., 2009. Exploratory Software Testing: Tips, Tricks, Tours and Techniques to Guide Test Design. Pearson Education Publisher, ISBN: 0321647858, 9780321647856.

Janzen, D. and H. Saiedian, 2005. Test-driven development concepts, taxonomy and future directions. Computer, 38(9): 43-50.

Karl, E.W., 2005. More About Software Requirements: Thorny Issues and Practical Advice. Microsoft Press.

Lewis, W., 2005. Software Testing and Continuous Quality Improvement. 2nd Edn., CRC Press, Boca Raton.

O'Connor, R., N. Baddoo and J.C. Gallego, 2009. Software process improvement. Proceeding of 16th European Conference, Euro SPI 2009, Alcaba (Madrid), Spain.

Pine, F.J., F. García and M. Piattini, 2008. Software process improvement in small and medium software enterprises: A systematic review. Software Qual. Control, 16(2): 237-261.

Pressman, R.S., 2005. Software Engineering: A Practitioner's Approach. 6th Edn., McGraw-Hill, New York.

Schulmeyer, G., 2008. Handbook of Software Quality Assurance. Artech House, Boston.

Thayer, R.H. and M.J. Christensen, 2005. Software Engineering. 3rd Edn., The Development Process, Wiley and Sons, New York.

Timothy, C.L. and R. Laganière, 2005. Object-oriented Software Engineering: Practical Software Development using UML and Java. 2nd Edn., McGraw-Hill, New York.

Weinberg, G., 2008. Perfect Software and Other Illusions about Testing. Dorset House.

Wiegers, K., 2005. Software Requirements. 2nd Edn., Microsoft Press, 2005.