International Advance Journal of Engineering, Science and Management (IAJESM)
ISSN -2393-8048, January-June 2018, Submitted in January 2018, iajesm2014@gmail.com

Study On Systematic Literature Review Related to Software Process Simulation Modeling

Basavaraj U, Assistant Professor, Department of Computer Science, Government First Grade College and PG Centre Thenkinidiyur Udupi, Karnataka India, Email- ubkottur@gmail.com

Abstract

Changes and continuous progress in logistics and productive systems make the realization of improvements in decision making necessary. Simulation is a good support tool for this type of decisions because it allows reproducing processes virtually to study their behavior, to analyze the impact of possible changes or to compare different design alternatives without the high cost of scale experiments. Although process simulation is usually focused on industrial processes, over the last two decades, new proposals have emerged to bring simulation techniques into software engineering. This paper describes a Systematic Literature Review (SLR) which returned 8070 papers (published from 2013 to 2017) by a systematic search in 4 digital libraries. After conducting this SLR, 36 Software Process Simulation Modeling (SPSM) works were selected as primary studies and were documented following a specific characterization scheme. This scheme allows characterizing each proposal according to the paradigm used and its technology base as well as its future line of work. Our purpose is to identify trends and directions for future research on SPSM after identifying and studying which proposals in this topic have been defined and the relationships and dependencies between these proposals in the last five years. After finishing this review, it is possible to conclude that SPSM continues to be a topic that is very much addressed by the scientific community, but each contribution has been proposed with particular goals. This review also concludes that Agent-Based Simulation and System Dynamics paradigm is increasing and decreasing, respectively, its trend among SPSM proposals in the last five years. Regarding Discrete-Event Simulation paradigm, it seems that it is strengthening its position among research community in recent years to design new approaches.

Introduction

Software testing is the process of executing software to determine whether it matches its specification and executes in its target environment (Whittaker, 2000). This process is not complete. Researchers have acknowledged the limitations of testing for a long time as vindicated by following famous statement from Edsger Dijkstra's Turing Lecture (Dijkstra, 1972). The famous statement is "Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence". This means that all software at the end of the day may be released with some faults. This makes some questions crop up on our minds which include questions like Why do we test software, if it is released with faults? How much testing needs to be done before its release? When do we decide to stop testing? When do we release the software? Will the lab-tested software faithfully run in the field environment? Have we covered the randomness and uncertainty of operational environment while testing the software under controlled settings, so on and so forth? Thus it necessitates the importance of attention that needs to be lent to the software development process. In (Jeske & Zhang, 2005) the authors present a detailed account of some of the formidable obstacles that beset the practice of the theory.

During the progress of testing the management is expected to systematically allocate resources so as to maximize reliability and minimize potential operational failure penalties (Lin & C.Y.Huang, 2008). This can be done by optimizing the resource allocations and consumptions, improving test. Effort, introducing state-of-art techniques and tools, upgrading staff skills and re-staffing etc. However, making these decisions is not easy. Scheduling deadline pressures or other ongoing projects may affect the manner in which testing resources could be allocated. All these factors in turn show up in terms of variations on testing effort consumption rate (Lin & C.Y.Huang, 2008). Thus testing effort could change and needs to be studied for minimization of costs and not missing deadlines. Use of SRGMs provides an opportunity to understand and improve the testing phase. Fault detection rate (FDR) functions have also been studied to understand the variations in failure rate per error. According to (Zhao, et al., 2006) fault detection is used for the effectiveness of fault detection capability of

testing techniques and tools. Moreover, FDR is dependent on factors like efficiency of fault discovery, inspection rate, fault density and testing–effort, in the beginning of testing. In middle of testing, it depends on factors which include program size, code expansion factor, failure to fault relationship, skill of testers, and reliability of software. Thus, FDR is changeable (Huang, 2005b). Different types of FDRs have been studied in reliability modeling. For example a bath-tub shaped FRD (Nadarajah, 2009), Vtub-shaped FRD (Pham, 2013), Loglog fault-detection rate (Pham, 2014). It may be noted that bathtub shaped FDR; we have the infant mortality period followed by useful life of system where system has a constant rate failure, followed by wear out period where system slows down more and more as more of wear out sets in. Bathtub shaped failure rate is obtained by using Weibull distribution (Pham, 2006) (Nadarajah, 2009), (Pham, 2013), (Pham, 2014).The other Vtub shaped FRD is obtained by using a log-log distribution and a Vtub shaped curve has the usual infant mortality period followed by useful life of system where system experiences failures at a relatively low increasing rate which is not constant rate failure, followed by an increased failure rate leads to aging of system (Pham, 2006) (Pham, 2013), (Pham, 2014). These papers present these models under NHPP framework.

Review of literature:

The contemporary computing professional works in an environment where programs are thousands or millions of lines long, are often extensively modified and maintained rather than constructed, are manipulated in a tool-rich environment, and where work is usually a team effort (Mulder, Haines, Prey & Lidtke, 1995). Computer scientists are not well prepared for this contemporary environment according to Prey (1996) because their preparatory training usually focuses on the construction of small programs (programming-in-the-small) and provides little experience in complex software development. In contrast, the development of large systems in an efficient and timely manner requires a team effort, and the more complicated the problem, the larger the team needed to solve it. Another contributing factor to the need for team development is that domain-specific expertise tends to be localized and geographically distributed. Studies have shown that, particularly when such developers are dispersed, their success depends critically on their ability to use effective groupware (Nunamaker, 1999). Such factors have made collaboration in systems development a necessity, not merely a technically feasible option. The emergence of the World Wide Web has fortunately made geographically distributed collaborative systems technologically feasible in a way that was difficult or impossible until recently. We shall use the term *groupware* to refer to the kind of software environments needed to support such a team, whose members collaborate over a network (Zwass, 1998). Groupware systems are intended to provide a team a shared workspace, despite being separated spatially and temporarily. Groupware or collaborative systems can be instrumental in alleviating the logistical difficulties that are associated with the application of distributed expertise. Indeed, the next generation of development processes is expected to focus on the effective integration of distributed expertise.

Experimental studies of both experienced programmers and novices have established the positive impact of collaboration. Wilson, Hoskin, and Nosek (1993) conducted a study to determine if experience with collaboration could benefit beginning programmers performing problem-solving/programming tasks. The experimental results provided positive support for the hypothesis that collaborative efforts could improve the problem-solving required in programming tasks. The experiment compared a control group of novice programmers, solving a software problem individually, with another group that allowed partners to communicate freely. The results demonstrated that even such simple collaboration enhanced the problem-solving performance of the novice programmer. The study also found evidence that an individual's ability had little overall effect on team performance, a phenomenon they claim occurred because the collaborative effort counterbalances individual deficiencies. The study also showed evidence that the collaboration provided the programmers confidence in the solution and enhanced their enjoyment of the problem solving process. Collaborative interactions appear to help beginning programmers analyze and model problems, and may

also help them master the analytical skills required by such tasks (Wilson, Hoskin & Nosek, 1993). Other controlled experimental studies indicate it is worthwhile to integrate collaborative activities even at the early stages of problem solving and programming training (Sabin & Sabin, 1994). Experiments with experienced software engineers (Nosek, 1998) also demonstrate that collaboration improves the problem solving process. Indeed, all team projects evaluated in the study outperformed comparable individually implemented projects, while at the same time team members were more personally satisfied with their work and had greater confidence in their solutions.

The overall objective of this review is to identify ways in which collaboration can facilitate the software development process. The review will examine collaborative problem solving and groupware in the software development domain, focusing on four areas: group problem solving, individual problem solving, groupware, and group psychology/sociology, including: group and individual problem solving models and tools, groupware systems, group cognition, and team dynamics. We will highlight the contributions and outstanding issues in group problem solving and group software development, with the objective of identifying an area of research that will represent an advance in the state of the art.

A group that develops a plan for designing a system that will solve an existing problem is by definition engaging in collaborative problem solving. Collaborative groups appear able to deal with complex tasks more effectively than individuals, partly because groups automatically have a broader range of skills and abilities than individuals (Finnegan & O'Mahony, 1996). Despite this, studies indicate that group problem solving is intrinsically more complex than individual problem solving (XXX Finnegan & O'Mahony, 1996). It can introduce difficulties that are specifically group-related, such as an interaction environment that inhibits the free expression of ideas (Hoffman, 1965), participation biases, conflicts caused by interpersonal difficulties, or complications arising from the structure of the group. Overall, however, the benefits of collaboration in problem solving far outweigh its disadvantages (Hohmann, 1997). For example, one notable benefit is the ancillary improvement of human capital effected by collaboration, because the individuals involved in a group learn from the skills and abilities of the other group members (Prey, 1996). The need to articulate designs, critiques, and arguments to other group members also hones an individual's technical, critical, and interpersonal skills (Guzdial et.al, 1996).

A collaborative problem solving *model* is an explicit methodology used to facilitate collaborative problem solving. A comprehensive such model will include not only generic problem-solving steps, domain-specific tasks, and requisite cognitive skills, but also the communication and coordination activities required by a collaborative environment. The collaborative problem solving method may be similar to an individual problem solving method. Indeed, in his important work on group software development, Hohmann (1997) observes that collaborative problem solving can be done using the very same problem solving methods that are used by individuals. Hohmann claims that while it is important for a group to explicitly choose and follow a problem solving method, and while group members should be familiar with the selected method, nonetheless, the method itself does not need to be designed specifically for group problem solving. Despite this laissez-faire approach to the chosen problem-solving method, Hohmann observes that the way in which a team will appropriate such a method in a collaborative environment, will differ substantially different from the way in which an individual will apply the same method.

A logical/qualitative method for facilitating joint decision making and alleviating conflict resolution was developed in Wong (1994). The approach is applicable to the kind of cooperation required of software engineers on a development project. Wong's model has three stages: identification, processing, and negotiation. The identification stage entails first identifying a decision agenda using priority-ordered criteria, then identifying the agents concerned with each criterion, where the term agent refers to the person or system responsible for a problem solving step. Competing alternatives are identified and the relationships among the alternatives are determined. The processing stage develops a set of so-called preference expressions for each criterion in the decision agenda. These preference

expressions are merely ordering relations for pairs of alternatives. The alternatives are then rank ordered to determine a recommended solution. A final negotiation stage then follows where the agents negotiate conflicts.

A model of group problem solving formulated by empirically observing group decision making behavior in environments which lay outside scientific/engineering/software development contexts was developed by Finnegan and O'Mahony (1996). This behaviorally-based model empirically recognized the same kind of problem-solving processes that have been systematically and explicitly articulated for engineering contexts. Groups progressed from an initial problem realization to a solution choice by a process dominated by communication of information and group collaboration, and needed significant levels of coordination and control throughout the decision making process. The initial, problem realization stage was typically initiated by a specialized group or by organization management. The next stage, planning, required coordination of subgroups. A subsequent information search stage was followed by group discussion of the information discovered about the problem. Subsequently, alternatives were identified and evaluated, and a preferred alternative selected, followed by validating, marketing or selling of the alternative to other groups, and ultimately implementation of the selected solution. The process is iterative, adapting to new requirements as they arise, reminiscent of user-centered software design in which a design is tested and redesigned through multiple iterations (Kies, Williges & Rosson, 1998).

We will use the models of Simon (1997) and Hohmann (1997) as points of reference. Simon's (1960) influence in the field of problem solving has been seminal (Deek, 1997; Hohmann, 1997). Though Simon considered only collaborative decision making, the similarities with collaborative problem solving (Huitt, 1992) makes his collaborative decision making model an important point of reference for the collaborative problem solving models we have described. Hohmann's (1997) model, on the other hand, is a useful point of reference because it is relatively comprehensive and closely related to some of Simon's most influential work.

Dubey Mohit and Aarti Garg (2014)3 opine that the IT industry plays vital role among all the industries in achieving country's objective of economic development. The IT sector has been rich over the years and has emerged as a key contributor to the global economic growth. IT sector is comprised of Software Services, Enabled Services (ITES) and Hardware. The sector has seen steady rise in growth trajectory throughout past few years. According to NASSCOM, the sector has shared 7 % of GDP in Indian economy. The prime aim of this article is to analyze the growth and performance of Information Technology in India. The study concludes that the IT sector has been revolutionary by creating employment opportunities with multiple scopes in various domains. Indian IT sector is one of the parts of global village and it is an instrumental way to transform Indian people in to the social modernization.

G. V. Vijayasri (2013) observes the relationship of the Indian economy and Information Technology industry along with Government promotion policies regarding IT industry. The study shows that the IT Industry has to play major role to the industrial verticals such as railways, airways, sea- network that have smooth functioning with IT Industry. The paper also looks back in to the years 1992-2001 where the phenomenal growth of industry services was marked over 50%.With the support of IT policies IT sector has provided 2.9 million jobs directly and 8.9 million jobs indirectly to the nation. Yet, IT sector has some challenges to face like insufficient subsidy, mistargeting and Government scares resources of the Indian IT industry and the rapid growth of 50,000 graduate engineers are in the queue of seeking employment in IT Industry in India every year.

Saji T.G. et al.'s (2013) the Global Financial Crisis and Performance of the Indian Corporate sector: A firm level analysis' involves the impact of financial crisis on some selected corporate sectors e.g. Banking, Reality and Infrastructure Sector, Automobile, FMCG, Pharmaceuticals & Information Technology. The selected four large groups of companies were Infosys, Wipro, TCS and Tech Mahindra .This study includes financial

performances like sales growth, earning growth, profit margin, return on equity, solvency ratio, earning per share, dividend yield and net worth of return from the period of 2006-07 to 2008-09. Review of Literature - II Economic Analysis of Changing Dimensions of IT Sector in India. The study was mainly based on pre and post crisis of the six sectors that were analyzed on the basis of secondary data collection. The results of the study showed that the growth of banking sector during the period of crisis was up securing all financial performances whereas, in this study of IT sector, the sales and earning profile were considerably improved in the study period. Wherein, Automobile companies in India slightly showed a steep decline in their earnings. FMCG (Fast moving consumer goods) were seemed to be not much affected by crisis while the pharmaceutical selected companies' growth showed negative trends in case of EPS, Dividend yield, return on net worth and came to the conclusion that the degree of global crisis did not shock the same of the Indian corporate sector. Some sectors were hit by global crisis and some remained unaffected.

Economic Survey (2012-13) concludes that Indian IT and ITES sector have started facing many challenges due to the rising growth of other countries in software services for example Sri Lanka 28 %, Argentina 37%, Philippines 69%, Ukraine59%, Costa Rica 35% and Russian Federation 27% during the years 2005 to 2011 which was higher than the world's top ten exporters. The big issue of outsourcing in USA and UK is that both countries have just initiated a local workforce. India should take up opportunity through improving a value chain software services focusing on domestic sector, raising wages in urban BPO, moving towards rural areas for skill development and English language training with USA, European and different countries. Das Shyamanuja (2012) in "The High Momentum Verticals" analyzes the vertical-wise trends of IT industry. Today in India the demand side of the IT sector (spending) varies in segment to segment of the business. Manufacturing is a broad sector whereas telecom is focused and well regulated, construction and retail are dominated by unorganized sectors; education that is dominated by government and so on. His methodology of the study is to compare 2012 data of IT spending and forecast in 2013 in percentage growth of IT industry and compare the verticals among Banking, Manufacturing, Telecom, Construction, Education, Automotive and Retail. He concludes that IT is now interwoven with business and it is not only finding business solutions but also finding a new opportunity for business. Review of Literature - II Economic Analysis of Changing Dimensions of IT Sector in India.

Kathuria Rajat and Mansi Kedia (2012) trace upon the growth phase of Telecom industry. The growth of teledensity increased with more than 75% driven by the growth of mobile telephony. In the case of mobile telecom network, India has become the second largest in the world after China. The research further points out the fact that, though the process o f telecom liberalization began in the 1980s, but the real reformation transcends in 1994 with the enactment the National Telecom Policy (NTP).Not only that Government created its corporatized departmental monopoly like MTNL and VSNL. The rest of monopoly managed by DoT (Department of Telecommunication) The Telecom Regulatory Authority was eventually set up in 1997The new telecom act created the TDSAT (Telecom Dispute Settlement and Appellate Tribunal) it was fast track disputes settlement process .DoT was also created BSNL in 2000 for provisions for services separately. Even at this point the country is obvious regarding the absence of strong policy of Electronic System Design and Manufacturing (ESDM) and critical to develop in the country. It is examined that combination of IT and telecommunications with their convergence technology has brought numerous opportunities from different domains. Thus, IT has come up as a sunrise sector for the country.

Natarajan Ganesh (2012) carries out that the all economies are passing through difficult times but IT industry has clear opportunity of the three million people by their reinforce skills and deliver maximum value of the industry. The financial services of JP Morgan and Barclays companies had bad impact on worlds companies but Indian IT industry has shown its good prospects in the area of Cloud computing, mobility, enterprises, social

media and big data.. Foundation leadership (NASSCOM) Provide strong new initiatives in the field of IT industry in India.

Reference:

Singpurwalla, N. D., 1991. Determining an Optimal Time Interval for Testing and Debugging Software. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, APRIL.VOL. 17(NO. 4).

Sommerville, I., Software Engineering, Fifth Edition, Addison-Wesley Publishers, 1996.

Sonnentag, S., Brodbeck, F., Heinbokel, T., Stolte, W., "Stressor-burnout relationship in software development teams", Journal of Occupational and Organizational Psychology, Volume 67, Pg. 327-344, December 1994.

Sourabh, Pushap C.(2012)" Embracing New Trends, „Dataquest , August 2012Vol. XXX, PP- 40-41.

Stacy, W., Macmillian, J., "Cognitive Bias in Software Engineering", Communications of the ACM, Volume 39, Number 6, pp. 57-63, June 1995.

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., Suchman, L., "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings", Communications of the ACM, Volume 30, Number 1, pp. 32 –47, January 1987.

Subbiah, A. and Selva Kumar M. (2008). „IT Sector: A Robust Growth", Facts for You, March2008 Vol. XXVII, PP.-31-33.

Swigger, K., Brazile, R., Shin, D., "Teaching Computer Science Students How to Work Together", CSCL Conference Proceedings, October 1995, Available [Online]: http://www-cscl95.indiana.edu/cscl95/swigger.html [26 November 2000].

T. J. Biggerstaff, Design recovery for maintenance and reuse, Computer, 22 (7), pp. 36-49, 1989 16. Binkely, D. The Application of Program Slicing to Regression Testing, Technical Report, Loyola College in Maryland, pp. 1-24, 1998.

Teng, X. & Pham, H., 2006. A new methodology for predicting software reliability in the random field environments. Reliability, IEEE Transactions on, Volume 55(3), pp. 458-468.

Thomas, J., Chapter 2 of Human Factors and Interactive Computer Systems, Edited by Yannis Vassiliou, Ablex Publishing Corporation, Norwood, NJ, 1984.