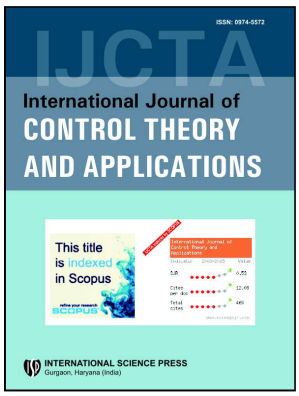

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 19 • 2017

Use of people traits and software tools in software development

B. Sunil Kumar[1] and Orsu. Naga Raju[2]

[1] *Department of Computer Science and Engineering, Narayana Engineering College Nellore, Andhra Pradesh, India.*
[2] *Department of Computer science,* Government College, Macherla , Andhra Pradesh, India.

Abstract: Software development is a very important field. Various approaches to software development has been proposed. Diverse people traits and tools and processes are need to support the software development ecosystem. This paper is a systematic literature review of people traits and tools needed. In this paper people traits needed at every stage of the software lifecycle are disused. Also various tools needed at each stage of the software development lifecycle are explained briefly. These two aspects are explained by detailing at each stage of requirement of the software lifecycle process.

Keywords: Software development, testing, design, deployment, tools, people, traits

1. INTRODUCTION

Software development is a very important activity which also contributes to the society at large. There are different types of software used at different stages of life. Software in the form of video games are used by kids and adults alike. Students use educational and e-learning soft5ware to get better grades in any college or university. Youngsters also use software in the form of event management portals to book movies and social events online. Youngsters also use the software to look for prospective matrimony portals. Married people use software to book holidays and also to book train and air tickets online. These days even grocery, furniture and every other goods required at home can be bought online with the help of ecommerce sites[1-3].

Software development is not an easy thing though. A lot of effort goes into building a software. Various software development models like lifecycle model, prototype model, spiral model and agile model have been proposed and used to build a particular software. This take care of the process part of software development. There are other factors that contribute to software development. Software is build by people using manual or automated tools which enable faster building of software[4].

When people are involved in building software there personality traits also contributes in building any software. People interact in various levels while building software. Software engineers are involved in writing basic programming and are at the bottom rung of the ladder of the corporate world. Team leads are involved in integrating the code written by software engineers and assisting the team under their mentorship to deliver within timelines. Project managers are involved in budgeting and solving any infrastructure issues related to any

project. Group and senior project managers are involved in interacting with customers and updating them about the developments and also to take their feedback to develop the software in conformity to their requirements. Delivery managers and the senior management are involved in facilitating the ecosystem better[5-6].

At each of these levels traits of these people involved in building a software plays a primordial role. Software engineers should be good team players and gel well with each other. Team leaders should possess enough leadership skills to solve any behavioral conflict among various team members who could have come from various backgrounds. Project managers should have skills to interact with customers on a weekly or a monthly basis. In these interactions communication of failure to deliver any task within timelines to a customer is extremely difficult and requires special skills. Senior managements responsibility lies in recruiting right person with right skills for a particular job who could carry out the above mentioned responsibilities without any hindrance[7-9].

The other important contributor to software development is the software used to facilitate every stage in the model being used. Right from the requirement analysis stage to the deployment stage there are various software tools used to enable and facilitate software development.

Prominent software's that are used today range from dial, together, rational rose software for design stage to selenium at the testing stage. Without the aid of these software's it would be extremely difficult to manually test or design any software. Even for planning and scheduling a particular project MS project is used which facilitates the same.

Another concept that is the buzzword today in the software world is that of automation. As per the famous jargon coined as fourth industrial revolution. It involves automating repetitive tasks so that once could focus on innovation in software development. Driverless cars, smart homes, smart city, smart traffic management, disaster prediction etc.. are some of the newer areas in which software could be built.

Scripting languages like python and Angular.js are widely being used to aid the process of automation. Robotics are going to be the ones which shall disrupt the software industry in the near future. Big data analytics is another area which might also lead to widespread disruptions that could shake up the software development ecosystem.

In this paper systematic literature review of the people and tools aspect of software development is carried out. Various traits of software developers along with in depth analysis of various tools used in building a software is taken into consideration.

It also throws some light on the probable disruptions that might change the software development landscape in the near future. How people traits and software tools gel together shall also be explained in this paper.

Our next section describes people traits that enable software development. In section 3 tools used to build a software are explained in details. Section 5 concludes this work and later acknowledgement is given to the data source followed by references.

2. PEOPLE TRAITS

Intellectual capital is the sum of all knowledge in possession of firms which they can use for their own competitive advantage. It involves all the resources which a company and can use while building the software. It comprises of three main subcategories[10-14]:

a) Human capital: It involves the skills required by people to execute their works in a better and productive manner. It involves knowledge about programs, the architecture, databases and the coding practices etc... It is to be possessed at the bottom rung of the ladder by people to program modules and to create and build databases. It also involves domain knowledge of employees and product knowledge. The

domain knowledge includes various alternate ways to solve various problems. It also includes creative aspects such as contributing to software development ecosystem by suggestive innovative ideas to software development ecosystem. It also includes knowledge of existing properties and concept and logic location within existing code. It also includes coding conventions, knowledge of development tools etc..

b) Social capital: This skills involves interpersonal skills needed to gel and interact with people to collaborate so as to develop software. It involves convincing skills needed at team lead or higher level to aid decision making. It also includes knowledge management skills needed to share knowledge within the team. It also includes giving constructive feedback to each other within the team so to develop the work environment. It involves distributing workload evenly within the team and also shifting workload whenever any team member is on leave. It not only involves relations within a team but also external or inter-team relations as well. This relations might include but is not restricted to collaboration with subject matter experts in the organizations, collaborating with other units, networking through various forums or social networking, collaborating with products heads and program managers and finally collaborating with customers. Another very important aspect of social capital is trust without which maintain inter personal relations is very difficult.

c) Organization capital: It includes resources regarding the software. It encompasses software code, documentation, architecture, environment, culture and infrastructure. This is a very important element as far as customer satisfaction is concerned because without organization capital satisfying a customer's needs shall be very difficult. Based on the software \architecture and software structure the entire corporate organization structure can be decided. Depending on the culture or approach followed in software development organization structure is decided.

Various levels in the organization structure different skill set from the intellectual capital so as to execute their works efficiently. At the bottom of the ladder focus is more on the human capital component and as one grows in the corporate ladder social and organization capital come into picture. It becomes imperative on the part of team and leads and above to acquire social skills without which growth in the corporate ladder is next to impossible.

Another concept in software development ecosystem is perceived intellectual capital and actual human capital. In many organizations there is a perception that gets built in the minds of senior management and this is also projected to the customers while signing deals for the company. But actually on ground and in reality the actual intellectual capital might be totally different. This gap between perceived and actual intellectual capital is what leads to customer dissatisfaction and delay in delivering an effective software within timelines with any software organization.

The only way which can bridge the gap between perceived and actual intellectual capital is by doing surveys between different teams and across the organization. The survey can be done by an external consultant so that without any fear people can give their honest opinions in the said surveys.

The senior management can then get a true picture about the gap between the perceived and actual intellectual capital and based on the same can take steps to lessen the gap by either hiring better people or by improving social skills by team engagement activities.

Action to mitigate gaps in social capital might include merging teams or shuffling of resources from one project to another and also shuffling resources at senior management level so as to get fresh ideas infused into different practice verticals.

Action in mitigating gap in human capital can be achieved by controlling attrition since as people grow within teams in experience it would obviously add to the overall human capital of the company. It can also be achieved with proper cross skilling and re skilling activities.

Action in mitigating organization capital can be accomplished by enhancing documentation within different software teams and increasing capital allocation for developing infrastructure across verticals. This call can only be taken y the senior management of any software company[15].

3. SOFTWARE ENGINEERING TOOLS

Software development has various approaches. One of the most earlier and prominent approaches is the lifecycle model. This section highlights the tools necessary to facilitate smooth functioning of each of the stages in the lifecycle model. The tools used in each of the stages of the lifecycle model are mentioned below[16-18]:

a) Requirement analysis: The first stage of the software lifecycle model is the requirement gathering or requirement analysis stage. In this stage the single point of contact from a company preferably a software project manager gathers requirements about the software to be developed from all stakeholders. In this stage two set of tools are prominently used:

i) Requirement modeling tools: Tools required to elicit, modify and validate requirement gathered from all the stakeholders are categorized as requirement modeling tools.

ii) Requirement traceability tools: Tools requirement to manage the proper flow and trace the requirement are categorized under this section.

b) Software Design tools: The second stage of the software lifecycle model is design stage. Here the algorithm and process and program flow of a particular project are finalized. Various Unified Modeling Language diagrams are drawn at this stage. Some of them are use case diagram, sequence diagram, collaboration diagram, activity diagram, state chart diagram etc.. Apart from this data flow and control flow diagrams are also finalized in this stage. Tools used in this stage include rational rose , dia etc..

c) Software development tools: Software development is the most important stage of the software lifecycle model. Here is very actually the software is programmed by software engineers. Various tools required at this stage include and are not restricted to program editors, interpreters, debuggers, compilers and code generators.

d) Software Testing tools: Once the software has been developed, it has to be tested for any bugs or errors and the same has to be rectified before deploying the same at the customer location. This is taken care by the software testing stage of the software lifecycle model. Tests can vary from being done at the program level by an individual. Here the individual tests the code he/she has written by using test cases and che3cks if he/she is getting desired output for any input given. At a later stage integration test is carried out to make sure that the complete software is free of errors. Finally load test is done to ensure the software is able to withstand more load once it gets deployed at the customer location. And at the last stage alpha and beta test are carried out to test the final cut of the software developed. Various tools can facilitate carrying out the above mentioned tests. These are IDE's like selenium etc..

e) Software deployment tools: Once the software is deployed and thoroughly tested for any errors/ bugs it needs to be deployed at the customer location or the end-user site. Various tools are used at this stage of the lifecycle model for the said software deployment like web server, application servers, databases etc.. Examples of web servers include Microsoft IIS, Apache web server. Examples of application servers include WebLogic application server, WebSphere application server etc..

In addition to the above mentioned tools other tools used in the software development are Microsoft Project which is used to plan and schedule a particular project[19-20].

4. PEOPLE TRAITS AND TOOLS

At different stages of software development lifecycle people with different skill sets and experience are needed. At an early stage people with 2-3 years of programming experience are needed and tools to be used are those under software development umbrella. At the next stage design tools and people are hired for projects[21-23]. prominent and people with 5-8 years of experience are hired and deployed onto projects.

5. CONCLUSION

In this paper people traits required for efficiently executing development of a software is explained. Apart from this tools required to facilitate software development is also explained. The people traits and tools are explained by comparing the requirements at every stage of the software development lifecycle model. At a future stage one can get into details of people traits and corresponding tools which can be enhanced so as to make the software development ecosystem advance further. One can demonstrate the same be making use of a pilot at any small software company and compare the results with the already existing best practices followed in Information technology industry in general and a small scale corporate in particular.

REFERENCES

[1] Shirley Cruz, Fabio Q.B. da Silva, Luiz Fernando Capretz, Forty years of research on personality in software engineering: A mapping study, Computers in Human Behavior, Volume 46, May 2015, Pages 94-113, ISSN 0747-5632, http://dx.doi.org/10.1016/j.chb.2014.12.008.

[2] Pan-Wei Ng, Integrating software engineering theory and practice using essence: A case study, Science of Computer Programming, Volume 101, 1 April 2015, Pages 66-78, ISSN 0167-6423, http://dx.doi.org/ 10.1016/j.scico. 2014.11.009.

[3] Mark van den Brand, Jan Friso Groote, Software engineering: Redundancy is key, Science of Computer Programming, Volume 97, Part 1, 1 January 2015, Pages 75-81, ISSN 0167-6423, http://dx.doi.org/ 10.1016/j. scico.2013.11.020.

[4] Richard F. Schmidt, Chapter 1 - Introduction to Software Engineering, In Software Engineering, Morgan Kaufmann, Boston, 2013, Pages 7-27, ISBN 9780124077683, http://dx.doi.org/10.1016/B978-0-12-407768-3.00001-X.

[5] Benjamin Satzger, Rostyslav Zabolotnyi, Schahram Dustdar, Stefan Wild, Martin Gaedke, Steffen Göbel and Tobias Nestler, Chapter 8 - Toward Collaborative Software Engineering Leveraging the Crowd, In Economics-Driven Software Architecture, Morgan Kaufmann, Boston, 2014, Pages 159-182, ISBN 9780124104648, http://dx. doi.org/10.1016/B978-0-12-410464-8.00008-8.

[6] Alok Chauhan, V. Vijayakumar, Rajiv Vincent, K.V. Pradeep, Towards the Development of a Framework for Socially Responsible Software by Analyzing Social Media Big Data on Cloud Through Ontological Engineering, Procedia Computer Science, Volume 50, 2015, Pages 524-530, ISSN 1877-0509, http://dx.doi.org/ 10.1016/j.procs. 2015.04.026.

[7] He Zhang, Muhammad Ali Babar, Paolo Tell, Identifying relevant studies in software engineering, Information and Software Technology, Volume 53, Issue 6, June 2011, Pages 625-637, ISSN 0950-5849, http://dx.doi.org /10.1016/ j.infsof.2010.12.010.

[8] Makrina Viola Kosti, Robert Feldt, Lefteris Angelis, Personality, emotional intelligence and work preferences in software engineering: An empirical study, Information and Software Technology, Volume 56, Issue 8, August 2014, Pages 973-990, ISSN 0950-5849, http://dx.doi.org/ 10.1016/j.infsof.2014.03.004.

[9] Oscar Pedreira, Félix García, Nieves Brisaboa, Mario Piattini, Gamification in software engineering – A systematic mapping, Information and Software Technology, Volume 57, January 2015, Pages 157-168, ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2014.08.007.

[10] Jaschar Domann, Sindy Hartmann, Michael Burkhardt, Alexander Barge, Sahin Albayrak, An Agile Method for Multiagent Software Engineering, Procedia Computer Science, Volume 32, 2014, Pages 928-934, ISSN 1877-0509, http://dx.doi.org/10.1016/j.procs.2014.05.513.

[11] Eva-Maria Schön, Jörg Thomaschewski, María José Escalona, Agile Requirements Engineering: A Systematic Literature Review, Computer Standards & Interfaces, Available online 2 September 2016, ISSN 0920-5489, http://dx.doi.org/10.1016/j.csi.2016.08.011.

[12] Marko Gasparic, Andrea Janes, What recommendation systems for software engineering recommend: A systematic literature review, Journal of Systems and Software, Volume 113, March 2016, Pages 101-113, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2015.11.036.

[13] T. Dybå, G.R. Bergersen and D.I.K. Sjøberg, Evidence-based software engineering, In Perspectives on Data Science for Software Engineering, edited by Tim Menzies, Laurie Williams and Thomas Zimmermann, Morgan Kaufmann, Boston, 2016, Pages 149-153, ISBN 9780128042069, http://dx.doi.org/10.1016/B978-0-12-804206-9.00029-5.

[14] Claes Wohlin, Darja Šmite, Nils Brede Moe, A general theory of software engineering: Balancing human, social and organizational capitals, Journal of Systems and Software, Volume 109, November 2015, Pages 229-242, ISSN 0164-1212, http://dx.doi.org/ 10.1016/ j.jss.2015.08.009.

[15] Per Lenberg, Robert Feldt, Lars Göran Wallgren, Behavioral software engineering: A definition and systematic literature review, Journal of Systems and Software, Volume 107, September 2015, Pages 15-37, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2015.04.084.

[16] Vahid Garousi, Ahmet Coskunçay, Aysu Betin-Can, Onur Demirörs, A survey of software engineering practices in Turkey, Journal of Systems and Software, Volume 108, October 2015, Pages 148-177, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2015.06.036.

[17] Anas Shatnawi, Abdelhak-Djamel Seriai, Houari Sahraoui, Zakarea Alshara, Reverse engineering reusable software components from object-oriented APIs, Journal of Systems and Software, Available online 5 July 2016, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2016.06.101.

[18] Vahid Garousi, Ahmet Coskunçay, Onur Demirörs, Ali Yazici, Cross-factor analysis of software engineering practices versus practitioner demographics: An exploratory study in Turkey, Journal of Systems and Software, Volume 111, January 2016, Pages 49-73, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2015.09.013.

[19] Ganesh Ram Santhanam, Qualitative optimization in software engineering: A short survey, Journal of Systems and Software, Volume 111, January 2016, Pages 149-156, ISSN 0164-1212, http://dx.doi.org/10.1016/ j.jss.2015.09.001.

[20] Tassio Vale, Ivica Crnkovic, Eduardo Santana de Almeida, Paulo Anselmo da Mota Silveira Neto, Yguaratã Cerqueira Cavalcanti, Silvio Romero de Lemos Meira, Twenty-eight years of component-based software engineering, Journal of Systems and Software, Volume 111, January 2016, Pages 128-148, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2015.09.019.

[21] Vahid Garousi, Kai Petersen, Baris Ozkan, Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review, Information and Software Technology, Volume 79, November 2016, Pages 106-127, ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2016.07.006.

[22] Parag C. Pendharkar, Ensemble based point and confidence interval forecasting in software engineering, Expert Systems with Applications, Volume 42, Issue 24, 30 December 2015, Pages 9441-9448, ISSN 0957-4174, http://dx.doi.org/10.1016/j.eswa.2015.08.002.

[23] Arjumand Bano Soomro, Norsaremah Salleh, Emilia Mendes, John Grundy, Giles Burch, Azlin Nordin, The effect of software engineers' personality traits on team climate and performance: A Systematic Literature Review, Information and Software Technology, Volume 73, May 2016, Pages 52-65, ISSN 0950-5849, http://dx.doi.org/ 10.1016/j.infsof.2016.01.006.